

**Spin-Spin Couplings in Two Limits:
Experimental, Theoretical, and Computational
Studies of Dipole-Coupled Nuclear Spins in Solids**

by

Daniel Kevin Sodickson

MIT LIBRARIES

JUL 18 1994

Submitted to the Harvard-MIT Division of Health Sciences and
Technology

Medical Engineering and Medical Physics Program
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Medical Physics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1994

© Massachusetts Institute of Technology 1994. All rights reserved.

Author
Harvard-MIT Division of Health Sciences and Technology
Medical Engineering and Medical Physics Program
^
May 9, 1994

Certified by
John S. Waugh, Ph.D.
Institute Professor, Department of Chemistry
Thesis Supervisor

Accepted by
Roger G. Mark, M.D., Ph.D.
Co-Director, Harvard-MIT Division of Health Sciences and
Technology

SCHERING

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

MAY 11 1994

LIBRARIES

**Spin-Spin Couplings in Two Limits:
Experimental, Theoretical, and Computational Studies of
Dipole-Coupled Nuclear Spins in Solids**

by

Daniel Kevin Sodickson

Submitted to the Harvard-MIT Division of Health Sciences and Technology
Medical Engineering and Medical Physics Program
on May 9, 1994, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Medical Physics

Abstract

For isolated sets of magnetically active nuclei, the dipole coupling serves as a sensitive gauge of interspin distances. A solid-state NMR technique has been developed to reintroduce homonuclear dipole couplings into magic angle spinning spectra which are normally free of dipolar effects. One- and two-dimensional experiments have been designed in which repetitive radiofrequency pulses combine with mechanical sample rotation to promote exchanges of magnetization among dipole-coupled spins over a wide range of chemical shift differences. A theoretical description is constructed based on the dynamics of a fictitious spin $1/2$, and this theoretical framework serves both as a basis for quantitative simulations and as a geometrical aid for visualization of the dipolar exchange process. Magnetization exchange experiments may be fit to simulations to extract precise internuclear distances, or else they may be used, in the style of liquid-state NOESY, to derive a qualitative map of atomic proximities for the determination of molecular structure in the solid state.

In large lattices of nuclear spins, the dipole coupling is responsible for relaxation processes such as spin diffusion, which can play a pivotal role in solid-state NMR spectra. Classical spin dynamics simulations have been used to explore certain of the microscopic underpinnings of the spin diffusion process. Conservation principles required for adherence to a traditional diffusion equation are identified, and the breakdown of diffusive behavior for magnetization in zero applied field is tied to non-conservation of spin angular momentum by the dipole-dipole interaction. The effects of dilution upon the spin diffusion constant are also studied. At low concentrations, the results are suggestive of percolation. Finally, the classical simulations are linked to quantum theories using the interpolating properties of spin coherent states.

Thesis Supervisor: John S. Waugh, Ph.D.

Title: Institute Professor, Department of Chemistry

Acknowledgments

A thumb placed strategically over a suggestive data point can alter radically the outcome of an experiment. For this quantum principle and classic lesson, I thank my research supervisor, Professor John Waugh, under whose discerning eye the work described in Part II of this thesis was done. At the same time, I thank Professor Waugh for supporting some of my more ambitious extrapolations in the course of the past two years of theoretical and computational exploration. I would also like to acknowledge his student and my erstwhile advisor, Professor Bob Griffin, in whose laboratory the first part of the thesis was engendered. To complete the triad of MIT physical chemists to whom this medical physicist turned for consultation, I thank Professor Bob Silbey for our periodic conversations.

As regards matters monetary, the work reported here on broadband dipolar recoupling was supported by the National Institutes of Health. A graduate fellowship from the National Science Foundation paved the way for my first three fiscal years of graduate school, and the NSF is also to be acknowledged for support of the work on spin diffusion.

My introduction to rotational resonance, and to many of the subtleties of the vigorous discipline known as spin gymnastics, came in large part through the papers and personal tutelage of Dr. Malcolm Levitt. Had he not departed to enlighten the Scandinavian peninsula, I would have looked forward to many an animated inquiry into the evolution of multiple-quantum coherences, the trajectory of squash balls, and the tumultuous array of influences in Middle-East politics.

To Emma, I owe many hours of cooperation and company – both in the lab and in the world at large. I look forward with excitement to sharing with her the exploration of that larger world, at times when the laboratory does not loom so large.

The chalk-dust hieroglyphics scratched on the small blackboard in MIT's Room 6-133 are the surest reminder of my last acknowledgements. It is a rare researcher who can stand with confidence and comfort before his father and his brother, and

unburden himself of hours' worth of arcane physical or mathematical speculations. Dad has been my scientific sounding board since first I had occasion to be curious. His unerring knack for asking the right questions has had a guiding influence on me for as long as I can recall, and his work is a continual reminder of the worthy ends to which one can put a physicist's training. Aaron, my brother and my friend, has been a valued colleague in these my graduate years. If he is satisfied with an argument of mine, I can rest assured that I have argued well, and have omitted no important step in my reasoning. From my friend and sister, Deborah, I learn the complexities of the lively discipline known as neuroscience, and I watch with appreciation as she outpaces her older brother in attaining the status of expert. My mother is a true master of that most complicated and interdisciplinary science of all – the understanding of what, from the remotest psychiatric principle to the most concrete practical experience – makes up a person. This space is too small to reveal the true lustre of the family with which I have been blessed. Perhaps a lifetime will be space enough.

Contents

Prologue	10
I Two-Spin and Few-Spin Systems: Broadband Dipolar Recoupling	13
1 Introduction	14
1.1 Dipole Couplings in Restricted Spin Systems	14
1.2 Broadband Dipolar Recoupling: Background and Motivations	15
2 Experiments	17
2.1 One-dimensional experiments	18
2.2 Two-dimensional experiments	22
3 Theory	28
4 Conclusions	38
II Many-Spin Systems: Spin Diffusion in Lattices	40
5 Introduction	41
5.1 Dipole Couplings in Extended Systems	41
5.2 Spin Diffusion	42
6 Classical Spin Dynamics and Spin Diffusion Calculations	44

6.1	The Diffusion Equation	45
6.2	The Initial Lattice Configuration	45
6.3	Time Evolution	47
6.4	Practicalities – a Sample Simulation	51
7	Classical Spin Diffusion: Results of Simulations	55
7.1	Spin Diffusion and the Effects of Applied Magnetic Field	56
7.2	Concentration Dependence of Spin Diffusion and the Effects of Lattice Geometry	71
8	Quantum Spin Dynamics and Spin Coherent States	87
8.1	Comparison of Classical and Quantum Spin Diffusion Using Moment Theory	88
8.2	The Quantum Dynamical Problem	89
8.3	Spin Coherent States – An Introduction	90
8.4	Spin Coherent States and Geometrical Phase	99
8.5	Spin Coherent States and Quasi-Classical Quantum Dynamics	101
	Epilogue	107
A	Evaluation of Moments in the Theory of Redfield and Yu	110
B	Matlab Code	115
B.1	Broadband Dipolar Recoupling	116
B.2	Classical Spin Diffusion	132
	A last word	200

List of Figures

2-1	Pulse sequence for one-dimensional magnetization exchange experiments	18
2-2	Magnetization exchange curves for $^{13}\text{C}_3$ -alanine	20
2-3	Magnetization exchange curves for $^{13}\text{C}_2$ -ZnAc	21
2-4	Pulse sequence for two-dimensional magnetization exchange experiments	23
2-5	Two-dimensional exchange spectra of $^{13}\text{C}_3$ -alanine in the absence of mixing pulses	25
2-6	Two-dimensional exchange spectra of $^{13}\text{C}_3$ -alanine with one π pulse applied per rotor period during the mixing period.	26
2-7	Schematic two-dimensional coupling map of $^{13}\text{C}_3$ -alanine.	27
3-1	Resonant structure of dipole coupling activity with and without π pulses	33
3-2	π pulse effects on fictitious fields.	34
3-3	Fictitious spin trajectories with and without π pulses.	35
3-4	Summary of fictitious-spin dynamics.	36
6-1	Output of a sample simulation of spin diffusion	52
6-2	Decay of the magnetization profile $M(x,t)$	53
7-1	Wavelength dependence for high-field diffusion of magnetization . . .	57
7-2	Wavelength dependence for high-field diffusion of interspin energy . .	58
7-3	Wavelength dependence for zero-field diffusion of interspin energy . .	59
7-4	Wavelength dependence for zero-field diffusion of magnetization . . .	60
7-5	Wavelength dependence for intermediate-field diffusion of Zeeman energy	66
7-6	Wavelength dependence for intermediate-field diffusion of dipole energy	67

7-7	Wavelength dependence for intermediate-field diffusion of total magnetic energy	68
7-8	Time course of average energies and disturbance amplitudes in intermediate applied field	69
7-9	Concentration dependence of the spin diffusion constant in a 3D lattice	73
7-10	Fit of $D(c)$ to the Redfield and Yu moment theory.	75
7-11	Concentration dependence of the spin diffusion constant in one, two, and three dimensions	78
7-12	Concentration dependence of normalized spin diffusion constants in one, two, and three dimensions	79
7-13	Comparison of simulated $D(c)$ curves with the predictions of moment theory and of uniform scaling arguments	81
7-14	Concentration dependence of high-field spin diffusion for a nearest-neighbor Heisenberg exchange coupling	82
7-15	Comparison of $D(c)$ for dipole couplings and Heisenberg exchange couplings.	84
7-16	Spin diffusion constant as a function of unit cell elongation.	85
8-1	Spin diffusion constant D_Z as a function of spin quantum number . .	88
8-2	Overlap of spin coherent states with the usual Zeeman states	95
8-3	Overlap of spin coherent states with each other	97

List of Tables

5.1	Comparative summary of diffusion constants for Zeeman energy. . . .	43
7.1	Diffusion constants for interspin energy in high and low applied magnetic field	71
7.2	Fitting parameters and diffusion constant values for the moment theory fit of $D(c)$	76

Prologue

*There is in this Earth no Maneuver so unnerving as the Spin.
Just when one thinks to have advanced into the Twilight,
Dawn comes round again.*

— Samuel Bowditch

The magnetic dipole-dipole interaction has long been recognized as a prominent player in the dynamics of nuclear spins. Since the early days of Nuclear Magnetic Resonance (NMR), the properties of the dipole coupling have been studied in a wide range of experimental and theoretical settings. Some of these properties have also been exploited in an equally broad range of experimental NMR techniques. This thesis describes work on dipole-coupled spin systems in two limits of complexity: first, in the few-spin limit, in which the full quantum-mechanical behavior of coupled spins may be appreciated and manipulated to advantage; and, second, in the many-spin limit, which requires approximate approaches, especially in the arena of quantum mechanics.

Dipole-dipole interactions depend strongly upon molecular geometry, and thus they may be used as a sensitive probe of molecular structure. Part I describes the theoretical foundation and experimental realization of an NMR technique for structure determination originating in the dynamics of the dipole coupling. The technique, dubbed Broadband Dipolar Recoupling (BDR) [1], uses a combination of radiofrequency pulses and mechanical sample rotation to promote dipole-mediated exchanges of spin magnetization under a broad range of experimental conditions. BDR experiments may be used to measure the through-space distances between pairs

of dipole-coupled spins, and to gain qualitative insight into coupling networks in more complicated spin systems. By providing detailed structural information in otherwise inhospitable circumstances, the BDR technique may help to elucidate the three-dimensional conformations of biomolecules in the solid state.

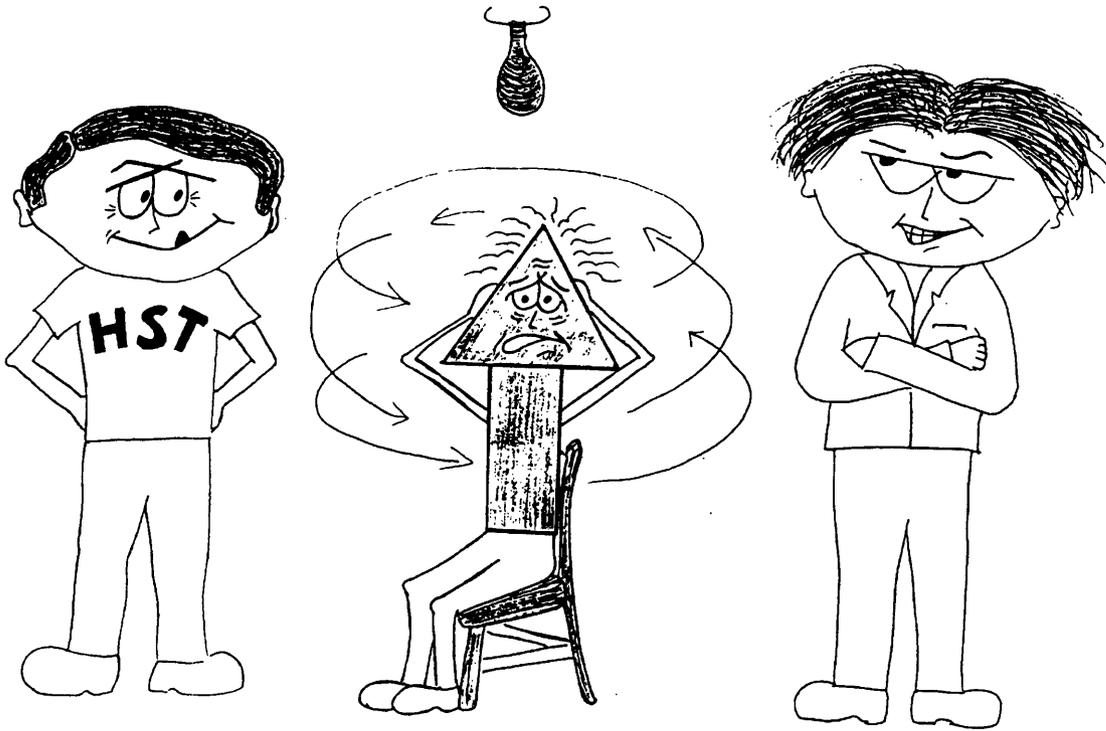
In Chapter 1, the motivations for Broadband Dipolar Recoupling are introduced, and the technological and conceptual groundwork for the technique are laid out. Chapter 2 contains the results of one- and two-dimensional BDR experiments on single amino acids and other test molecules. Chapter 3 presents a theoretical description of the broadband dipolar recoupling effect. Possible areas of application of the BDR technique, and its potential advantages over and interactions with other available recoupling techniques, are assessed in Chapter 4.

In Part II, the discussion turns to systems containing many dipole coupled spins, and in particular to the dynamics of spin diffusion in lattices. The spin diffusion process is founded upon dipolar mechanisms similar to those driving magnetization exchange in the BDR experiments, but it boasts a greater complexity owing to the extended nature of the spin system. For this reason, spin diffusion is of interest as a model of many-body dynamics under the influence of long-range interactions. In practice, spin diffusion can also play a pivotal role in solid-state NMR spectra.

Chapter 5 introduces prior work on spin diffusion, in the realms of theory, experiment, and numerical simulation. Chapter 6 outlines the methodology of classical simulations which are used in Chapter 7 to explore new facets of the spin diffusion process. In Chapter 8, quantum theories and classical simulations are juxtaposed and connected, using the mathematical apparatus of spin coherent states as well as an earlier theory of moments.

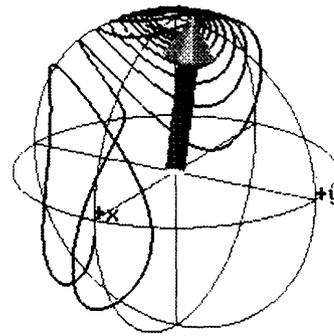
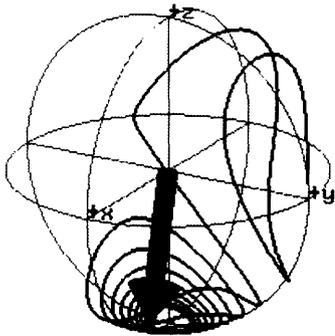
The endeavors of both Parts of this thesis, and the tasks of the NMR theorist and spectroscopist more generally, may be viewed as efforts in spin interrogation, probing nuclear spins for information about their neighbors and their material surroundings. Whether the day-to-day relations between spin and scientist are amicable or no, the dipole coupling may be counted upon to provide an effective mode of communication.

SPIN INTERROGATION



Part I

Two-Spin and Few-Spin Systems: Broadband Dipolar Recoupling



Chapter 1

Introduction

1.1 Dipole Couplings in Restricted Spin Systems

The dipole coupling drives mutual “spin-flips” between neighboring spins. Classically, this behavior corresponds to the continuous coordinated motion of each spin in the dipole field of the others. Quantum mechanically, one may trace the exchange of spin magnetization to the flip-flop terms in the dipolar Hamiltonian, $I_{\pm}S_{\mp}$, which cause transfer of population and coherence among states in an appropriate subspace of the full Hilbert space. When the spins are arranged in isolated clusters, the spin system may be treated as an ensemble of relatively simple two-state or few-state systems, and the quantum dynamical problem is tractable. In such cases, one may achieve a fine level of control over the precise pathways of dynamical evolution. By tailoring the influences governing the spin system, one may, in essence, transmute the Hamiltonian to a form amenable to study. Such is the alchemistic art of NMR spectroscopy. In the chapters to come, a particular combination of experimental influences will be used to construct a microscope of sorts for scrutinizing the dipole coupling. By juggling sample rotation, radiofrequency pulsing, and free spin precession, one may achieve well-resolved solid-state NMR spectra in which the activity of the dipole coupling is manifest.

1.2 Broadband Dipolar Recoupling: Background and Motivations

The strong distance dependence of the magnetic dipole-dipole interaction makes it an attractive tool for the measurement of internuclear distances. This fact has been effectively exploited in liquid-state NMR, where nuclear Overhauser effect spectroscopy (NOESY) [2] is used to determine three-dimensional molecular structures at the atomic level. In liquids, molecular motions modulate dipole-dipole interactions, and the fluctuating dipole fields can be used to drive magnetization exchange (or cross-relaxation) between spins over a wide range of chemical shift differences and Zeeman energies. This is more difficult in solids, where more restricted molecular motions often cannot supply the energy differences necessary for magnetization exchange (and where limitations in resolution and sensitivity make the exchange more difficult to discern). In some cases, heteronuclear dipolar couplings fluctuating under the influence of strong homonuclear couplings may provide an exchange mechanism [3][4]. Usually, however, more specific experimental intervention is necessary.

Several solid-state techniques have been developed to reintroduce dipole couplings under magic-angle-spinning (MAS) conditions in order to derive selected structural information. Rotational resonance (R^2) [5][6] experiments and Dipolar Recovery at the Magic Angle (DRAMA) [7] both allow homonuclear dipole couplings to be measured in the presence of rapid sample rotation. Both techniques, however, suffer from various restrictions in the range of chemical shift differences accessible to study. In the DRAMA experiment, $\pi/2$ radio-frequency (RF) pulses interrupt MAS averaging of the dipole coupling, leading to a nonvanishing dipolar average Hamiltonian. The average Hamiltonian analysis implies that resonance offsets and chemical shifts must be relatively small compared to the rotor frequency. In rotational resonance experiments, on the other hand, dipolar effects are reintroduced in a resonant fashion when the sample rotation rate is matched to the chemical shift separation of two coupled spins[5][6]. The R^2 phenomenon, while not restricted to small offsets or anisotropies, is highly frequency-selective: the MAS sample rotation rate must be carefully ad-

justed to an integral submultiple of the isotropic chemical shift separation of a given spin pair. Barring fortuitous accidents in the distribution of chemical shifts, all other magnetically distinct spin pairs lie off rotational resonance and show no significant dipole-mediated effects. Although the spectral selectivity of R^2 is often advantageous in quantitative studies, since multiple and relayed couplings are eliminated, a technique that placed less stringent constraints upon the sample rotation speed would be of great utility. With such a technique, spin pairs whose chemical shift separations lie outside the usual limits of stable sample spinning could be studied more easily, and systems of more than two coupled spins could be investigated efficiently.

Chapters 2 to 4 describe a dipolar recoupling technique which is significantly less constrained by chemical shifts than DRAMA or rotational resonance – a technique which allows dipole couplings between multiple spin pairs with different chemical shift separations to be measured simultaneously [1]. Not long ago, Gullion and Vega [8][9] reported that π pulse trains promote the dephasing of rotational echoes in dipole-coupled homonuclear spin pairs over a wide range of sample rotation speeds and chemical shift differences. This dephasing effect has been attributed to a revival of dipole couplings, and in fact, many of the effects of homonuclear dipole couplings may be reintroduced in a broadband fashion into MAS spectra by judiciously spaced π pulses. Chapter 2 presents the results of magnetization exchange experiments exploiting the broadband dipolar recoupling (BDR) effect, and Chapter 3 provides a theoretical description in which pulse-assisted dipolar recoupling may be understood as a form of compensated rotational resonance. As well as allowing for convenient quantitative simulations of the BDR effect, the theory also suggests a geometrical picture for understanding the mechanism of BDR. Potential applications for molecular structure determination are discussed in Chapter 4.

Chapter 2

Experiments

From the rate of magnetization exchange between the spins in a coupled pair, one may deduce the strength of the coupling between them. The coupling strength, in turn, is inversely proportional to the cube of the interspin distance.¹ Thus, measurements of magnetization exchange allow precise determination of the distances between coupled spins. Two particular classes of magnetization exchange experiments will be useful in characterizing couplings and extracting distances. The first class involves selective inversion of one of the spins and direct observation of the subsequent dipole-mediated exchange. Even though multiple spectral data sets must be acquired to follow the resonances of the exchanging spins over time, these experiments generally make reference to only one frequency domain, and they will be considered “one-dimensional”. The other class of magnetization exchange experiments is genuinely two-dimensional, yielding traditional 2D spectra whose structure of diagonal and off-diagonal peaks gives information about the geometry of coupling networks.

¹In liquids, where the spins are subject to rapid and incoherent tumbling motions, the effective distance dependence of the small residual dipole coupling is closer to $1/r^6$.

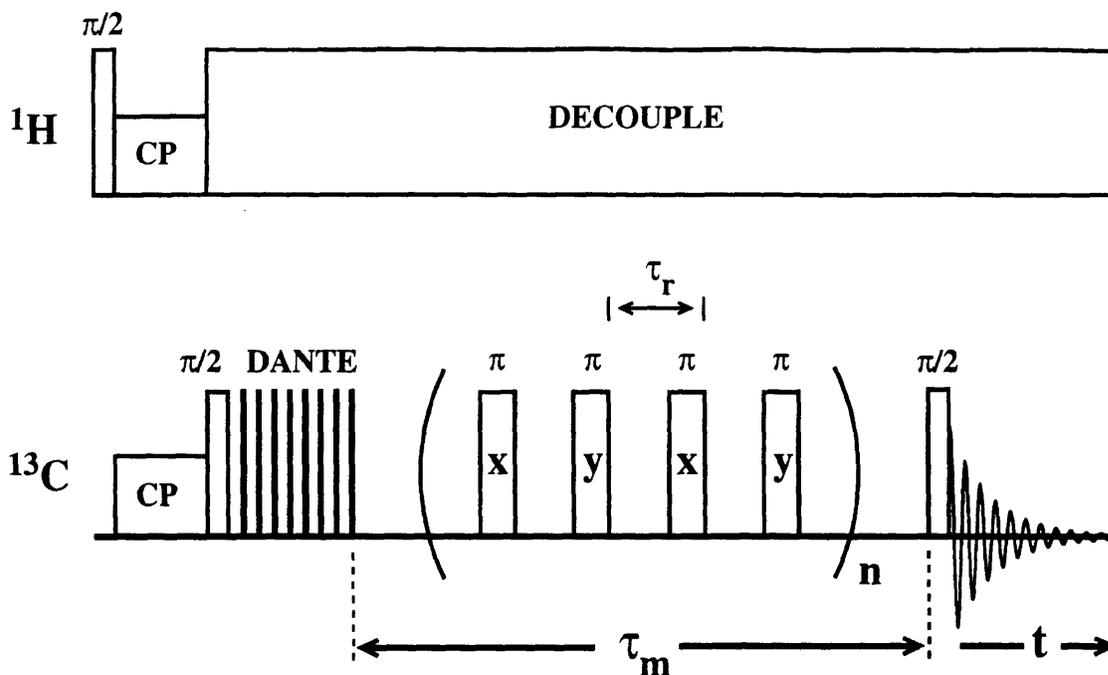


Figure 2-1: The ^1H - ^{13}C double-resonance experiment used to study longitudinal magnetization exchange.

2.1 One-dimensional experiments

The pulse sequence used for one-dimensional magnetization exchange experiments is shown in Figure 2-1.² The presence of π pulses during the mixing time is the only difference between this sequence and the one generally used for R^2 magnetization exchange experiments [6]. Following cross-polarization from abundant protons and storage along the z axis, the magnetization of one ^{13}C resonance is selectively inverted. Longitudinal magnetization exchange occurs for a variable mixing time τ_m , during

²The 1D experiments were performed on 99% ^{13}C -labelled model compounds (1,2- $^{13}\text{C}_2$ -Zinc Acetate Dihydrate and 1,2,3- $^{13}\text{C}_3$ -DL Alanine) at 79.9 MHz ^{13}C frequency on a home-built spectrometer. Selective inversion was performed using an eight-pulse DANTE [10] sequence $((\pi/8) \times 8)$, and CW proton decoupling was applied throughout the subsequent mixing period, so as to suppress proton-driven magnetization exchange. The 16-step phase cycle included 2-step cycling of the ^{13}C storage and flipback pulses to remove transverse magnetization components during the mixing time, plus a conventional 4-step CYCLOPS and ^1H $\pi/2$ pulse phase alternation to remove pulse ringdown artifacts. The ^{13}C RF carrier was placed roughly in the center of the carbon spectrum to minimize differential resonance offset effects. ^{13}C RF field strengths were set at 54 kHz, and ^1H field strengths at 81 kHz, yielding a Hartmann-Hahn mismatch of 3.5 dB between the ^{13}C and the ^1H channels.

which nonselective π pulses are applied once per rotor period. At the end of the mixing period, a nonselective $\pi/2$ pulse returns the magnetization to the transverse plane for observation. Following Fourier transformation of the resulting free induction decays, the integrated intensities of the spectral lines for each spin site (or suitable sum and difference intensities) are plotted against τ_m . Magnetization exchange, and hence exchange in spectral intensities following τ_m , occurs only between spins with an active dipole coupling.

Figure 2-2 shows the results of experiments performed on uniformly ^{13}C -labelled alanine. In the presence of strong ^1H decoupling, this constitutes a three-spin system with three distinct chemical shifts. The alpha carbon magnetization was selectively inverted, and integrated intensities for all three carbon spectral lines were then followed as a function of τ_m . In the absence of mixing pulses, exchange is expected only at rotational resonance. Indeed, in Figure 2-2(a) (left-hand column), no exchange occurs, since no rotational resonance condition is satisfied, whilst in Figure 2-2(b) (left-hand column), rotational resonance drives magnetization transfer between the $^{13}\text{C}_\alpha$ and $^{13}\text{CO}_2$ sites, as shown by the rapid equilibration of these two magnetization components. Note that the $^{13}\text{CH}_3$ site does not participate in the exchange. When π pulses are applied during the mixing time, on the other hand, exchange occurs throughout the three-spin network, at both spinning speeds considered (Figure 2-2, right-hand column).

In Figure 2-3, experimental results are compared with simulations for the two-spin system of doubly ^{13}C -labelled zinc acetate ($\text{Zn}^{2+}(\text{C}^*\text{H}_3\text{-C}^*\text{OO}^-)_2 \cdot 2\text{H}_2\text{O}$) in the presence of strong ^1H decoupling. In this case, the methyl carbon magnetization was selectively inverted, and the average difference magnetization $\langle I_z - S_z \rangle$ (I=carboxyl carbon, S=methyl carbon) was plotted against τ_m for several rotation frequencies. The decay of $\langle I_z - S_z \rangle$ to zero indicates the equilibration of magnetization. The plots also show trajectories of $\langle I_z + S_z \rangle$ in the absence of selective inversion, as a check for the effect of pulse imperfections (see below). Without mixing pulses (left column), magnetization exchange occurs only on rotational resonance ($\nu_r=4.376\text{kHz}$), $\langle I_z - S_z \rangle$ remaining essentially flat under off-resonant conditions ($\nu_r=5.000\text{kHz}$ and

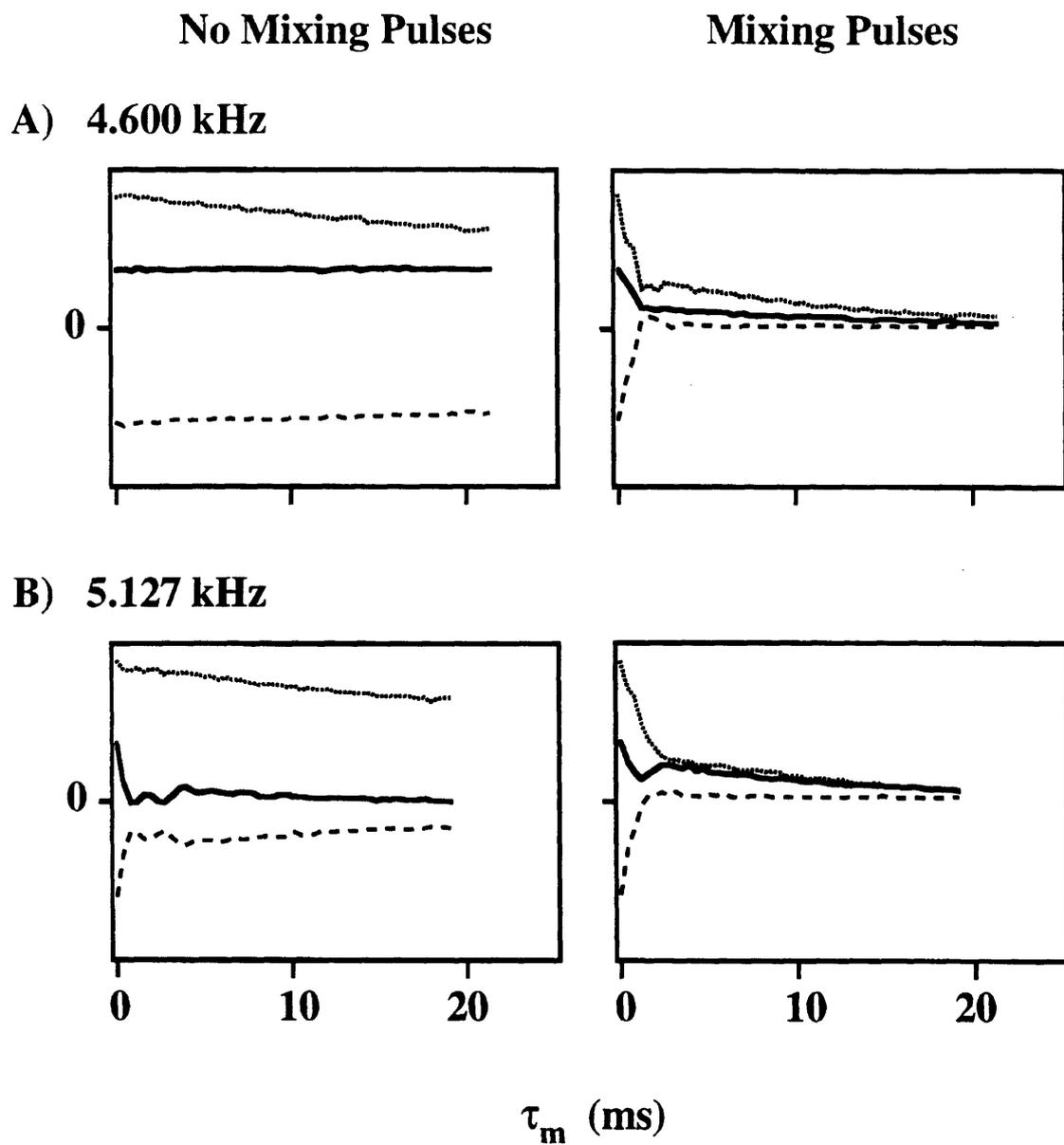


Figure 2-2: Longitudinal magnetization (arbitrary units) plotted against τ_m for $^{13}\text{C}_3$ -alanine. Solid lines represent carboxyl carbon magnetization, dashed lines the selectively inverted alpha carbon magnetization, and dotted lines the methyl carbon magnetization. (A) Off rotational resonance. (B) On an $n = 2$ rotational resonance between the carboxyl and the alpha carbons.

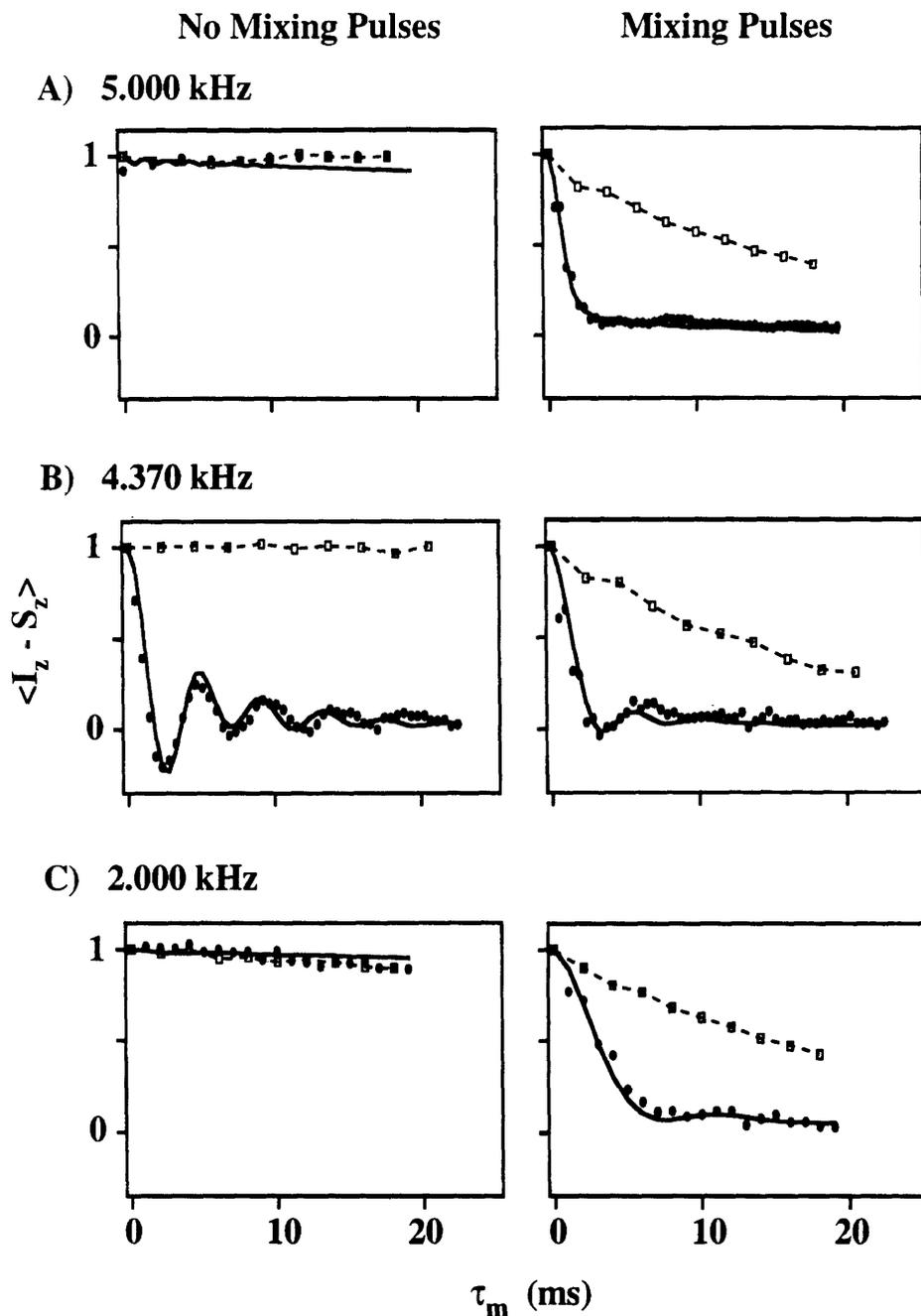


Figure 2-3: Experimental (filled circles) and simulated (solid line) magnetization exchange curves for $^{13}\text{C}_2\text{-ZnAc}$. Difference magnetization $\langle I_z - S_z \rangle$ (normalized to 1 at the outset) is plotted against mixing time τ_m . Results for pulse-free mixing periods are shown on the left. The corresponding curves with one π pulse per rotor period applied during mixing are shown on the right. Control curves of $\langle I_z + S_z \rangle$ for experiments lacking a selective inversion are also included (dashed lines with open squares). The rotor speed in (B) corresponds to an $n = 3$ rotational resonance. (A) and (C) are off rotational resonance.

$\nu_r=2.000\text{kHz}$). When a π pulse train is applied during τ_m , however, significant exchange is observed at all rotor speeds (right column).

The dashed lines in Figure 2-3 indicate the evolution of the total longitudinal magnetization $\langle I_z + S_z \rangle$ in experiments where the selective inversion was omitted. These control experiments make it possible to distinguish the cumulative signal losses due to imperfect mixing pulses from actual magnetization exchange. To minimize the losses, RF fields were made as strong as possible, and Hartmann-Hahn cross-polarization effects during the pulses were avoided by maintaining a substantial Hartmann-Hahn mismatch between the ^{13}C and the ^1H channels.³ In addition, 90° phase-alternated mixing pulse sequences $\{(\pi)_x - \tau - (\pi)_y - \tau\}_n$, as described by Gullion *et al* [11], were used to compensate for pulse imperfections. Although these schemes were initially designed for the case of transverse magnetization, they also perform well in preserving longitudinal magnetization.

The solid lines in Figure 2-3 are theoretical curves based on calculations to be described in Chapter 3.

2.2 Two-dimensional experiments

The same mechanism of longitudinal magnetization exchange driving the 1D experiments is also active in the two-dimensional experiments. Rather than varying the mixing time to plot out the time course of exchange, however, one now uses a fixed mixing time τ_m interposed between two separated evolution times, t_1 and t_2 . Figure 2-4 shows the pulse sequence for the two-dimensional experiments.⁴ This sequence is markedly similar in its outlines to a NOESY sequence, once again with the notable difference that π pulses are present during τ_m to promote broadband exchange.

³Composite mixing pulses may also be used to minimize cumulative pulse errors, but the advantages of composite pulse compensation are generally offset by the added length of the composite pulses, which affords more time for Hartmann-Hahn depolarization.

⁴99% 1,2,3- ^{13}C -DL Alanine was used for these experiments. The Larmor frequencies, carrier position, and RF field strengths were the same as for the one-dimensional experiments. A 64-step phase cycle was used to restrict the coherence pathway to -1 quantum coherence during t_1 and t_2 and zero-quantum coherence during τ_m . A spectrum of 200 t_1 slices took roughly 7 hours to acquire.

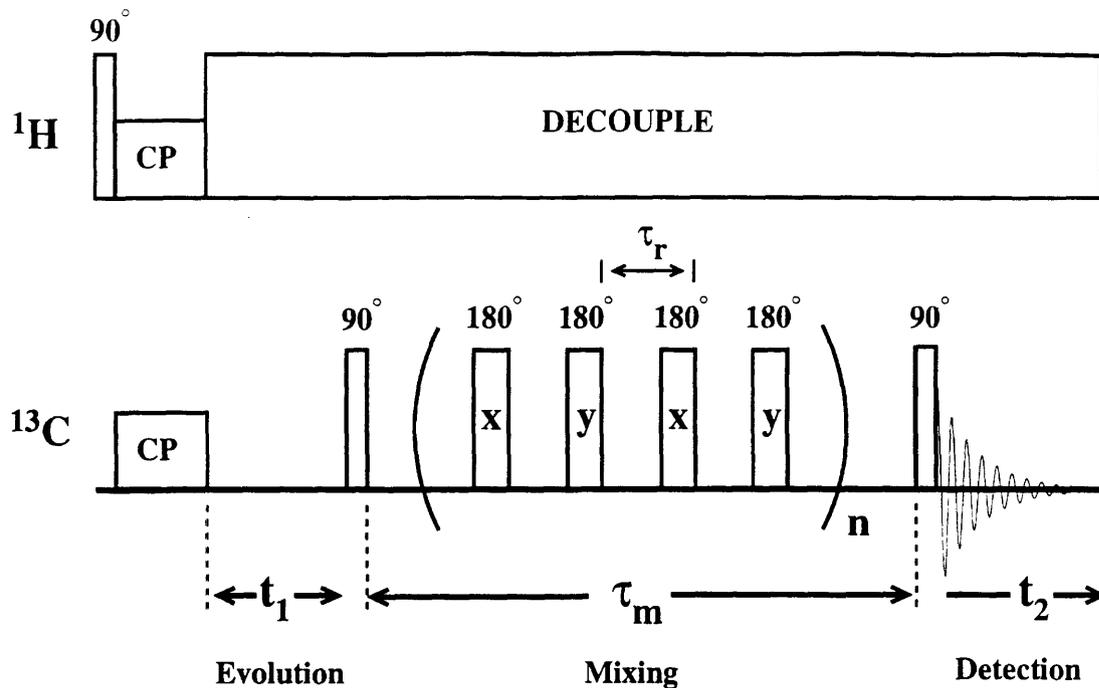


Figure 2-4: The ^1H - ^{13}C double-resonance experiment used to generate two-dimensional magnetization exchange spectra.

Evolution of spin magnetization in the transverse plane occurs during t_1 , followed by longitudinal storage during τ_m . Phase cycling is used to remove any transverse magnetization that may survive T_2 dephasing during τ_m . To the extent that a dipole coupling is present, any differential z magnetization created by the $\pi/2$ storage pulse will exchange during the mixing time, until the next $\pi/2$ pulse initiates signal acquisition for a time t_2 . Dipole-driven exchange is visible as cross-peaks connecting spectral sites along the diagonal in the doubly Fourier transformed data set. Since each 2D experiment employs only a single mixing time, corresponding to a single time point in the 1D magnetization exchange curves of the previous section, the 2D experiments are not as sensitive a measure of the precise dipole coupling strength as the 1D experiments (whose oscillatory features are a convenient foothold for fitting to simulations). Nevertheless, the 2D spectra provide a clear graphical demonstration of coupling configurations, and they are well suited to the simultaneous study of multiple couplings.

No cross-peaks appear in the pulse-free experiment when the sample rotation rate does not match a rotational resonance condition (Figure 2-5A). On rotational resonance, exchange is visible only between spins whose chemical shift difference matches the resonance condition (Figure 2-5B). The presence of a π pulse train during τ_m (one π pulse per rotor period) regenerates the expected coupling map for the full three carbon alanine molecule (Figure 2-6). A schematic coupling diagram for alanine is included in Figure 2-7 for comparison with Figure 2-6. Strong cross-peaks are visible connecting neighboring carbons (carboxyl and alpha, alpha and methyl), while relatively little exchange is observed between the more distant carboxyl and methyl sites.⁵ Thus, as in liquid-state NOESY experiments, the two-dimensional BDR spectra may be interpreted as maps of the through-space distances between spins.

⁵If the duration of the mixing time is increased significantly, strong cross-peaks will appear between the carboxyl and methyl sites as well. These peaks may be attributed to a combination of direct and relayed couplings, and may complicate the estimation of interspin distances. The choice of mixing time, therefore, is of some importance, and should be tied to the base coupling strength one expects to find between neighboring spins. Also note that the diagonal peaks on either side of the carboxyl resonance in Figures 2-5A and 2-6 are magic angle spinning sidebands of that resonance. In Figure 2-5B, the spinning sidebands are too small to be seen at the contour level of the plot, since they are broadened by the dipolar effects of rotational resonance. One may infer from the $n = 2$ resonance condition that one sideband is located midway between the carboxyl and the alpha resonances.

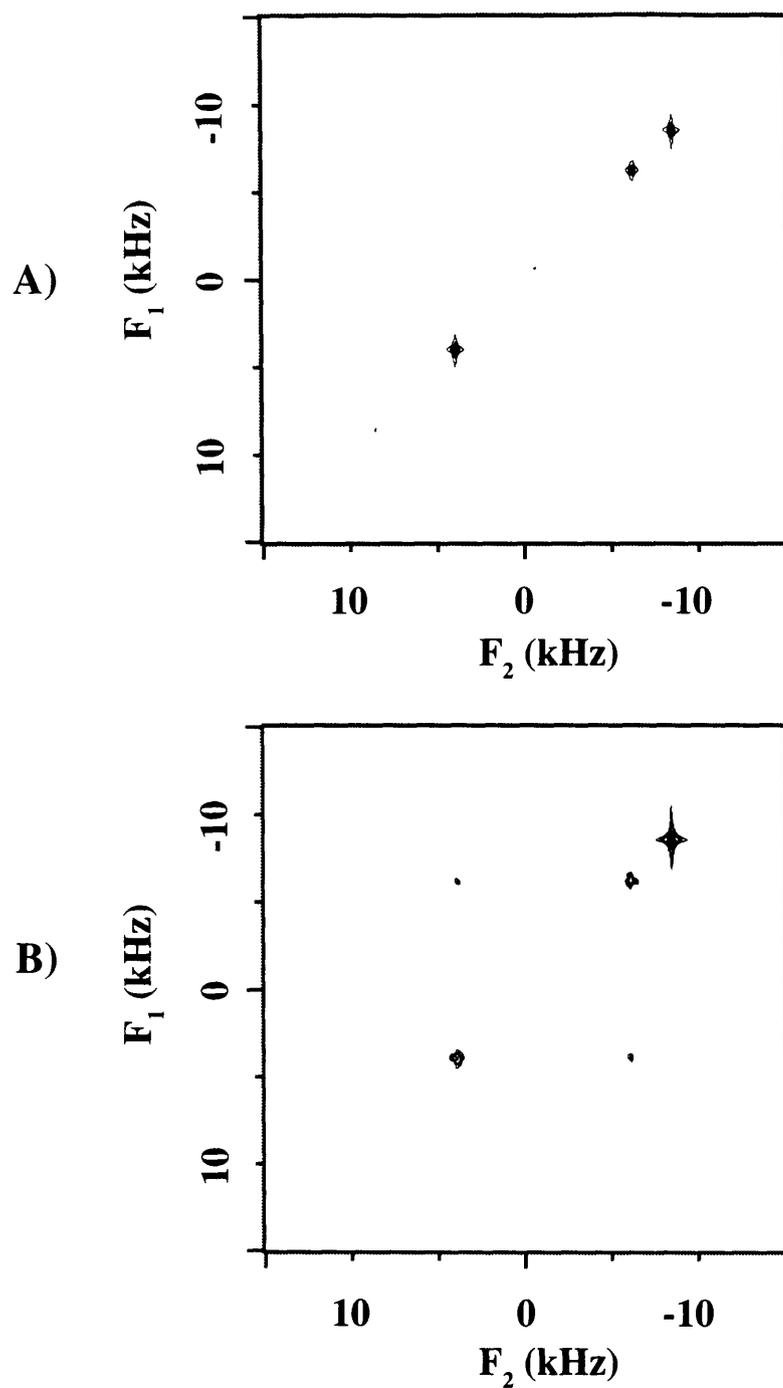


Figure 2-5: Two-dimensional exchange spectra of $^{13}\text{C}_3$ -alanine in the absence of mixing pulses. (A) Off rotational resonance. MAS rotor speed $\nu_r = 4600\text{Hz}$. (B) On rotational resonance ($n = 2$) between carboxyl and alpha carbons. $\nu_r = 5130\text{Hz}$. In both (A) and (B), the mixing time $\tau_m = 5.2\text{ms}$. Spectra are displayed in absolute value contour mode.

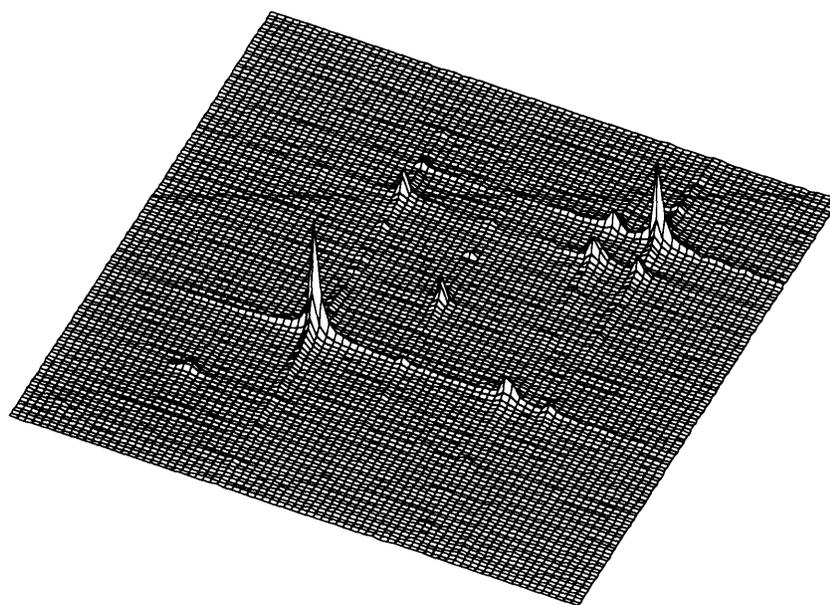
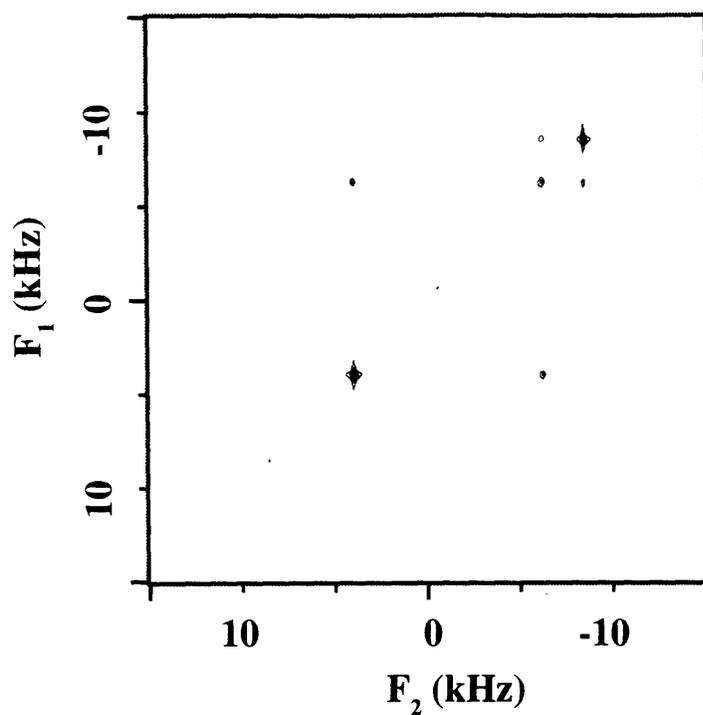


Figure 2-6: Two-dimensional exchange spectra of $^{13}\text{C}_3$ -alanine with one π pulse applied per rotor period during the mixing period. $\nu_r = 4600\text{Hz}$, $\tau_m = 1.3\text{ms}$. A contour plot and the corresponding stack plot of the absolute-value spectrum are shown.

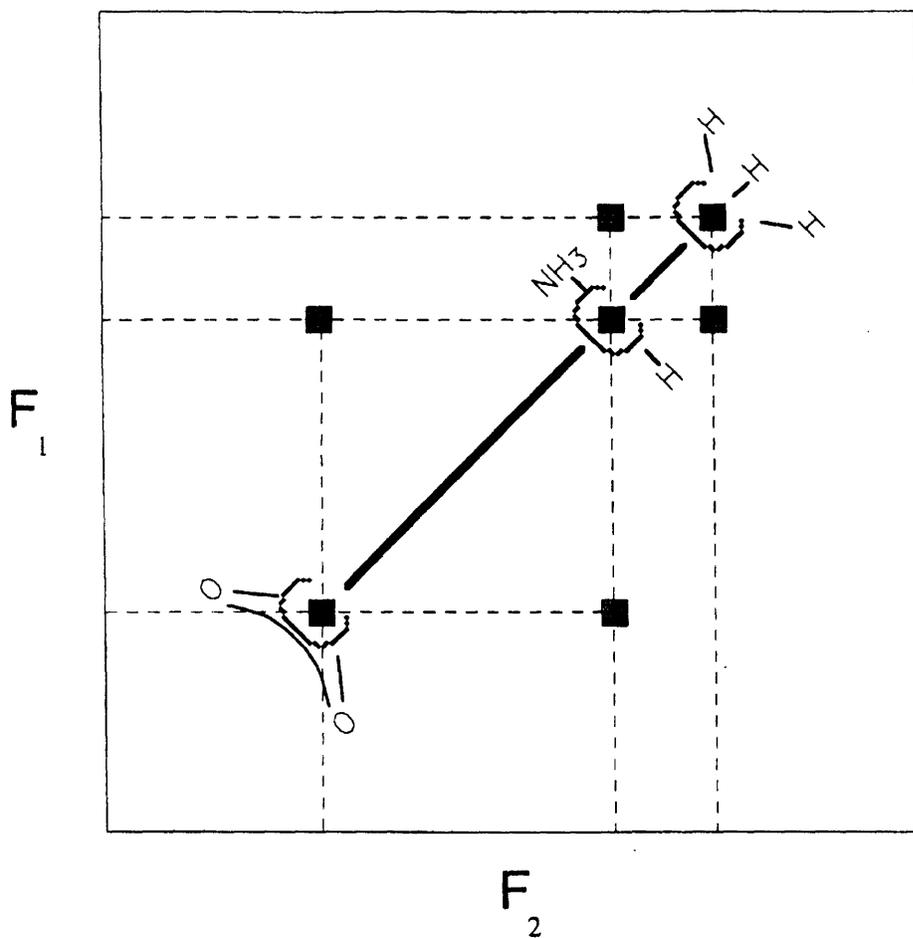


Figure 2-7: Schematic two-dimensional coupling map of $^{13}\text{C}_3$ -alanine.

Chapter 3

Theory

The recoupling effects of π pulses have been treated with Floquet theory [8][9] and with coherent averaging theory [12][13]. Here, we shall take a more pictorial or geometric approach, concentrating on the two central levels of the four-level system formed by two spins-1/2. The exchange of populations and coherences between these two levels, driven by the dipole flip-flop term, may be described in terms of a fictitious spin-1/2. The extent of dipole-mediated magnetization exchange may then be traced geometrically, much as one traces spin magnetization trajectories in other magnetic resonance experiments. The fictitious spin formalism allows immediate comparison with the narrowband recoupling phenomenon of rotational resonance. In this framework, broadband dipolar recoupling can be understood as a mismatch-compensated rotational resonance effect, precisely analogous to mismatch-compensated Hartmann-Hahn cross-polarization.

Consider an isolated pair of homonuclear spins-1/2, I and S , in a rotating powdered solid. The MAS Hamiltonian for this spin system may be separated into two commuting parts,

$$\begin{aligned} H &= H_0 + H_1 \\ H_0 &= \omega_\Sigma \frac{1}{2} (I_z + S_z) + \omega_A 2I_z S_z \\ H_1 &= \omega_\Delta \frac{1}{2} (I_z - S_z) + \omega_B \frac{1}{2} (I_+ S_- + I_- S_+) \end{aligned} \quad (3.1)$$

Here we follow the notational conventions of Levitt *et al.* [5], with $\omega_\Sigma = \omega_I + \omega_S$ and $\omega_\Delta = \omega_I - \omega_S$ (ω_I and ω_S are the Larmor precession frequencies of spins I and S). ω_A and ω_B describe the I-S couplings, including in ω_B the MAS-modulated flip-flop term of the dipole field. Labelling the four spin states $|m_I, m_S\rangle$ as in Ref. [5],

$$\begin{aligned}
|1\rangle &= \left| +\frac{1}{2}, +\frac{1}{2} \right\rangle \\
|2\rangle &= \left| +\frac{1}{2}, -\frac{1}{2} \right\rangle \\
|3\rangle &= \left| -\frac{1}{2}, +\frac{1}{2} \right\rangle \\
|4\rangle &= \left| -\frac{1}{2}, -\frac{1}{2} \right\rangle
\end{aligned} \tag{3.2}$$

and using the standard definitions of single-transition operators [14][15],

$$\begin{aligned}
I_z^{14} &= \frac{1}{2} (|1\rangle\langle 1| - |4\rangle\langle 4|) = \frac{1}{2} (I_z + S_z) \\
I_z^{23} &= \frac{1}{2} (|2\rangle\langle 2| - |3\rangle\langle 3|) = \frac{1}{2} (I_z - S_z) \\
I_x^{23} &= \frac{1}{2} (|2\rangle\langle 3| + |3\rangle\langle 2|) = \frac{1}{2} (I_+ S_- + I_- S_+)
\end{aligned} \tag{3.3}$$

as well as the unit operator

$$\mathbb{1} = \mathbb{1}^{14} + \mathbb{1}^{23} = |1\rangle\langle 1| + |4\rangle\langle 4| + |2\rangle\langle 2| + |3\rangle\langle 3| \tag{3.4}$$

we may rewrite the two parts of the Hamiltonian as follows:

$$\begin{aligned}
H_0 &= \omega_\Sigma I_z^{14} + \omega_A \frac{1}{2} (\mathbb{1}^{14} - \mathbb{1}^{23}) \\
H_1 &= \omega_\Delta I_z^{23} + \omega_B I_x^{23} = (\omega_\Delta \hat{\mathbf{z}} + \omega_B \hat{\mathbf{x}}) \cdot \mathbf{I}^{23}
\end{aligned} \tag{3.5}$$

From Equation (3.3), one can see that an initial state of pure I -spin Zeeman magnetization I_z corresponds to a superposition of $I_z^{14} + I_z^{23}$, whereas S_z corresponds to the difference $I_z^{14} - I_z^{23}$. Magnetization exchange ($I_z \rightarrow S_z$) takes place if I_z^{23} can be inverted while I_z^{14} is left unperturbed. In the absence of RF fields, I_z^{14} commutes with H and is therefore constant, so that all the significant dynamics takes place in the

$\{|2\rangle, |3\rangle\}$ space as described by the Hamiltonian H_1 .

H_1 can be identified as the Hamiltonian of a “fictitious spin-1/2” \mathbf{I}^{23} in the presence of a “fictitious field” with a z component given by $-\omega_\Delta/\gamma$ and an x component given by $-\omega_B/\gamma$. In the conventional nomenclature of NMR, the “constant”¹ z -direction fictitious field component may be said to bear an analogy to B_0 , while the time-varying x component (which contains the MAS-modulated dipole coupling) plays the role of an oscillating irradiation field B_1 . Rotational resonance occurs when the characteristic frequency of one of the Fourier components of $\omega_B(t)$ matches ω_Δ , the difference precession frequency of the I and S spins. In the rotating frame of this resonant Fourier component (and making suitable assumptions [5]), ω_Δ disappears, and the fictitious spin is free to precess about the x axis, passing from $+z$ to $-z$. This constitutes R^2 magnetization exchange, since the populations of states $|2\rangle$ and $|3\rangle$ are exchanged (I_z^{23} is inverted) whilst the populations of states $|1\rangle$ and $|4\rangle$ are unchanged (I_z^{14} is preserved). Off R^2 , the fictitious spin precesses about a tilted effective field, and exchange is always incomplete (see Figures 3-4(a) and 3-4(b)).

Now let us consider the effects of π pulses upon the spin-pair system. In the presence of RF fields, I_z^{14} does not always commute with H , and we must consider the dynamics in the $\{|1\rangle, |4\rangle\}$ space as well as the $\{|2\rangle, |3\rangle\}$ space. Ideal, nonselective, delta-function π pulses of phase ϕ lead to the following transformations of the spin states:

$$\begin{aligned}
 |1\rangle &\Rightarrow -e^{+2i\phi}|4\rangle \\
 |2\rangle &\Rightarrow -|3\rangle \\
 |3\rangle &\Rightarrow -|2\rangle \\
 |4\rangle &\Rightarrow -e^{-2i\phi}|1\rangle
 \end{aligned} \tag{3.6}$$

¹For nonvanishing chemical shift anisotropy differences, ω_Δ may also be time-dependent, but this time-dependence may be removed by transformation to an appropriate interaction frame. See Ref. [5].

The corresponding single-transition operator transformations are

$$\begin{aligned}
I_x^{14} &\Rightarrow I_x^{14} \cos 4\phi + I_y^{14} \sin 4\phi & I_x^{23} &\Rightarrow I_x^{23} \\
I_y^{14} &\Rightarrow I_x^{14} \sin 4\phi - I_y^{14} \cos 4\phi & I_y^{23} &\Rightarrow I_y^{23} \\
I_z^{14} &\Rightarrow -I_z^{14} & I_z^{23} &\Rightarrow -I_z^{23}
\end{aligned} \tag{3.7}$$

Ideal π pulses produce π rotations about the x -axis in the $\{|2\rangle, |3\rangle\}$ subspace and π rotations about the axis $(\hat{x} \cos 2\phi + \hat{y} \sin 2\phi)$ in the $\{|1\rangle, |4\rangle\}$ subspace. Since I_z^{14} and I_z^{23} are both inverted by these rotations, a π pulse does not lead by itself to magnetization exchange between the two spins.

It is convenient to transform to the interaction frame of the π pulses, in which the pulses are taken to rotate the fields rather than the spins. In this frame, the two components of the Hamiltonian may be written

$$\begin{aligned}
H_0 &= \pm\omega_\Sigma I_z^{14} + \omega_A \frac{1}{2} (\mathbb{I}^{14} - \mathbb{I}^{23}) \\
H_1 &= \pm\omega_\Delta I_z^{23} + \omega_B I_x^{23}
\end{aligned} \tag{3.8}$$

where the negative signs apply to periods following an odd number of π pulses. This Hamiltonian bears a striking resemblance to the Hamiltonian in MOIST cross-polarization [16], with the slight difference that the I_x^{14} term from MOIST is absent and that the ω_B term is time-dependent. By transforming to the rotating frame of the nearest-resonant Fourier component of ω_B , and neglecting all other Fourier components, we obtain a time-independent Hamiltonian

$$H_1 = \pm\Delta_n I_z^{23} + \omega_B^{(n)} I_x^{23} \tag{3.9}$$

with the “resonance offset” $\Delta_n = \omega_\Delta - n\omega_r$. The correspondence with MOIST in the $\{|2\rangle, |3\rangle\}$ subspace is now exact. Thus, the mechanisms for broadband dipolar recoupling by π pulses are analogous to the mechanisms for broadband cross-polarization by MOIST. In MOIST, π phase shifts of the irradiation compensate for offset from the

Hartmann-Hahn condition; here π pulses at regular intervals compensate for offset from rotational resonance.

Adapting the results of Ref. [16], we obtain an analytic expression for the time-dependence of the average difference z -magnetization under a train of ideal π pulses with uniform spacing τ :

$$\begin{aligned} \langle I_z - S_z \rangle(\tau_m) &\cong \cos \left\{ \frac{\tau_m}{\tau} \cos^{-1} \left[\cos^2(\beta_m/2) + \sin^2(\beta_m/2) \cos(2\theta_\Delta) \right] \right\} \\ \beta_m &= \left(|\omega_B^{(n)}|^2 + |\Delta_n|^2 \right)^{1/2} \tau_m \\ \cos(2\theta_\Delta) &= \frac{|\omega_B^{(n)}|^2 - |\Delta_n|^2}{|\omega_B^{(n)}|^2 + |\Delta_n|^2} \end{aligned} \quad (3.10)$$

This expression is valid in the vicinity of a rotational resonance, where all but the nearest-resonant Fourier component of $\omega_B(t)$ may safely be ignored. The same nearest-resonant approximation may be used to derive a formula for exchange in the absence of mixing pulses:

$$\langle I_z - S_z \rangle(\tau_m) \cong \frac{|\Delta_n|^2 + |\omega_B^{(n)}|^2 \cos(\beta_m)}{|\Delta_n|^2 + |\omega_B^{(n)}|^2} \quad (3.11)$$

The results of calculations using Eq.'s (3.10) and (3.11) are compared in Figure 3-1. The dramatic broadening of rotational resonances produced by π pulses is obvious. Without pulses, a plot of the difference magnetization $\langle I_z - S_z \rangle(\tau_m \approx 20\text{ms})$ versus rotor frequency shows the characteristic resonant structure. When pulses are applied, however, exchange occurs over a wide range of rotor speeds, and $\langle I_z - S_z \rangle$ is essentially flat at the resonant value of zero after 20ms.

Study of \mathbf{I}^{23} fictitious spin trajectories suggests a physical interpretation for pulse-induced rotational resonance mismatch compensation. From Equation (3.8), it can be seen that after each π pulse, the tilted effective fictitious field is replaced by its mirror image through the transverse plane, since the z component is inverted while the x component remains unaffected. This is demonstrated in Figure 3-2. On an R^2 condition, Δ_n vanishes in the appropriate rotating frame, and the resultant transverse

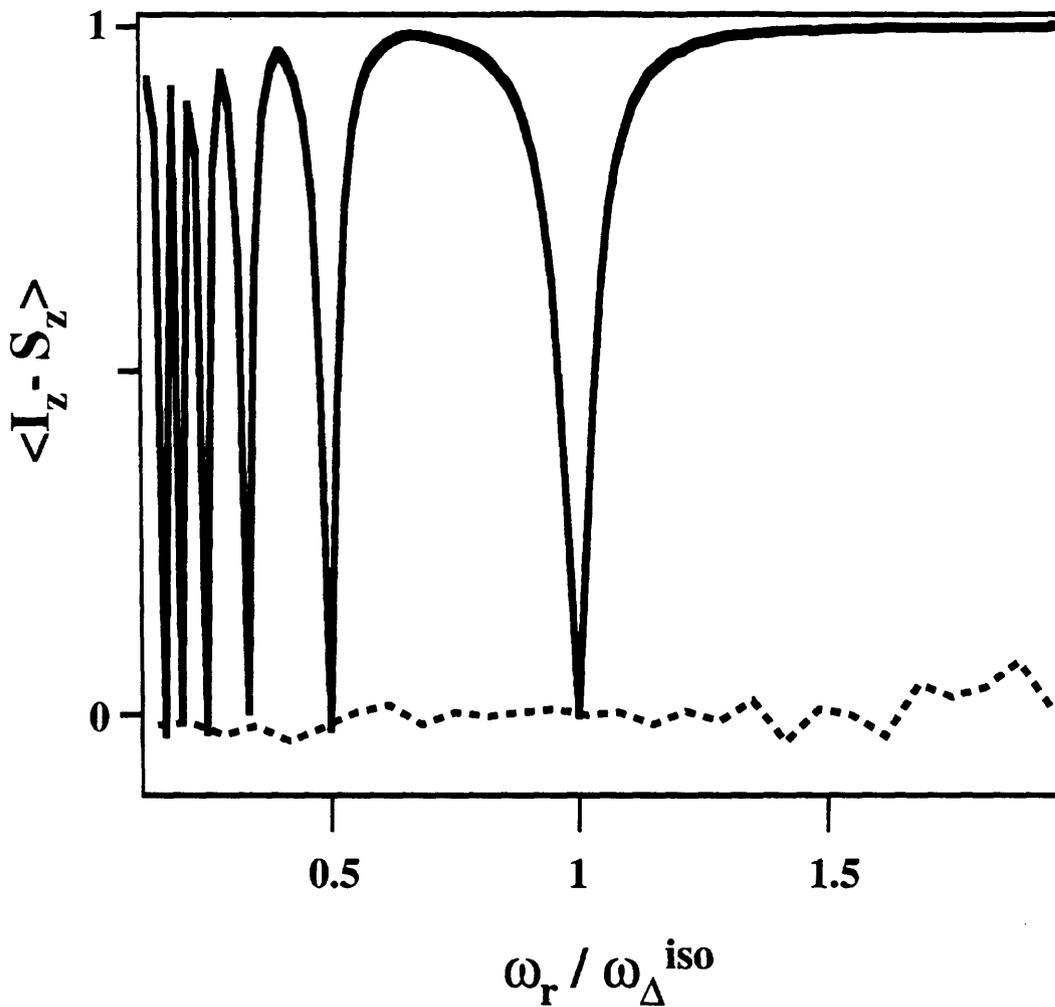


Figure 3-1: Calculated powder-averaged difference magnetization $\langle I_z - S_z \rangle$ in the vicinity of $\tau_m = 20ms$, as a function of rotor speed ω_r (measured in multiples of the isotropic chemical shift difference $\omega_{\Delta}^{iso} = \omega_I^{iso} - \omega_S^{iso}$). The solid line shows structure due to rotational resonance, calculated from Equation (3.11). The dashed line was calculated using Equation (3.10) for one π pulse per rotor period. Magnetization trajectories $\langle I_z - S_z \rangle(\tau_m)$ were computed as sums over random ensembles of 1000 crystallite orientations, using simulation parameters for zinc acetate.

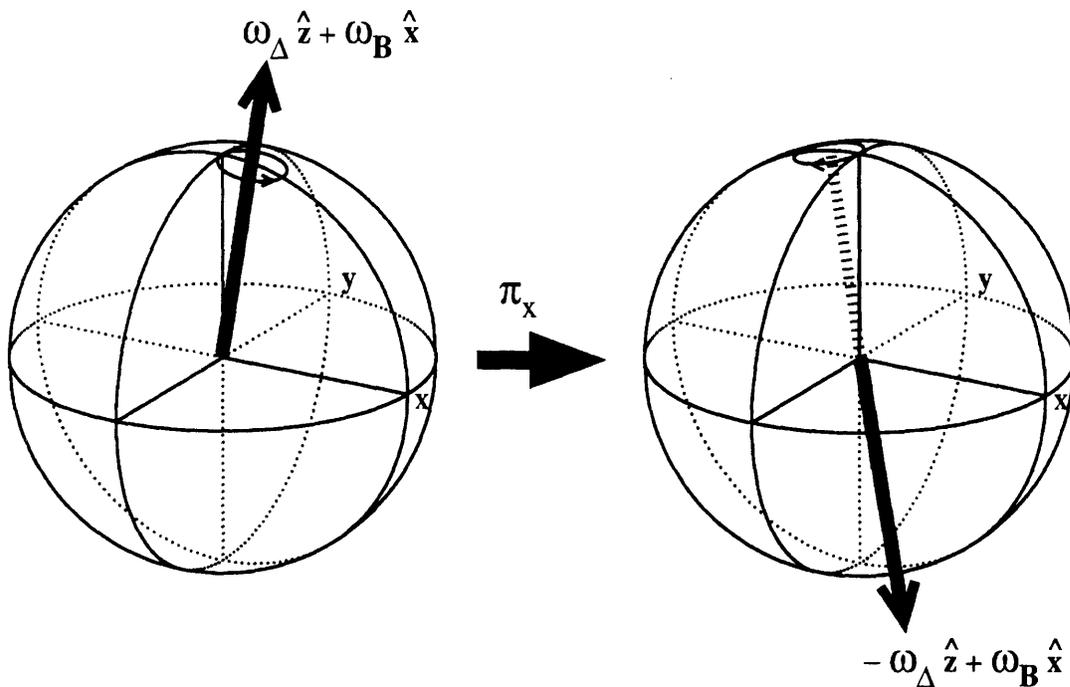


Figure 3-2: π pulse effects on $\{|2\rangle, |3\rangle\}$ fictitious fields. After each pulse, the fictitious field is inverted through the transverse plane.

fictitious field is unaffected by π pulses. Off resonance, on the other hand, a fictitious spin that begins by precessing about a tilted axis in the upper hemisphere before a π pulse will precess instead about a mirror axis in the lower hemisphere after the pulse. If this axis switching is repeated, a spin that would otherwise have maintained a tight circular off-resonant trajectory zig-zags its way from $+z$ to $-z$, and complete magnetization exchange can occur. The geometry of this situation is illustrated in Figure 3-3 and in Figure 3-4(c).

Figure 3-4 summarizes the dynamics of $\{|2\rangle, |3\rangle\}$ fictitious spins with and without π pulse trains. For comparison with experimental results, magnetization exchange profiles may be furnished by the z -projection of \mathbf{I}^{23} trajectories, since $\langle I_z^{23} \rangle = \frac{1}{2} \langle I_z - S_z \rangle$. The effects of parameters such as π pulse spacing may easily be explored with the fictitious spin picture, since it provides a direct visual measure of magnetization exchange.

For quantitative simulations of dipole-mediated magnetization exchange, one can

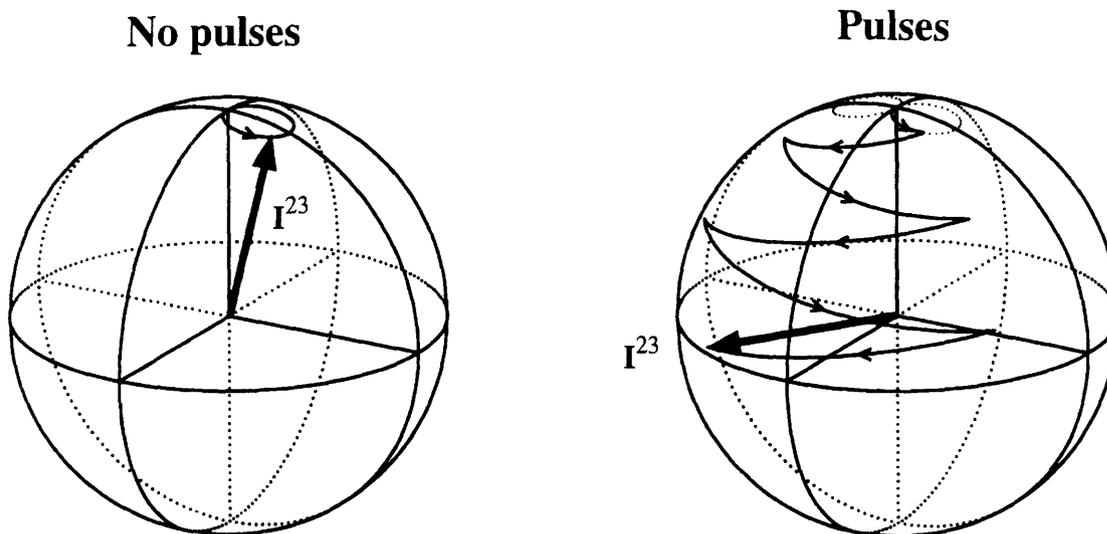


Figure 3-3: \mathbf{I}^{23} fictitious-spin trajectories with and without π pulses (for a single crystallite, in the interaction frame of the nearest-resonant Fourier component of $\omega_B(t)$). In the absence of pulses, an off-resonant fictitious spin precesses about a single tilted axis in one hemisphere. With pulses, it follows a zig-zag trajectory between hemispheres, enabling complete magnetization exchange.

abandon the nearest-resonant approximation and include the full time-dependence of ω_B . In order to fit experimental exchange curves, one must also perform a full powder average and introduce zero-quantum relaxation (equivalent to T_2 relaxation of the fictitious spin magnetization). It should be noted that the effect of π pulses upon the zero-quantum T_2 is expected to be negligible as long as the correlation times of the dominant zero-quantum dephasing processes are sufficiently shorter than the pulse repetition time, which will usually be the case.² With the addition of π rotations, the simulation strategy is essentially the same as that for pulse-free R^2 magnetization exchange. The simulated exchange curves in Figure 2-3 were generated following this strategy.³ To account for pulse imperfections, simulations were attenuated according to the signal loss curves in control experiments.

One final comment on the compensation mechanism of π pulses is in order. One

²See Ref. [5] and references therein for details on zero-quantum relaxation.

³See Appendix B for Matlab code used to execute these simulations.

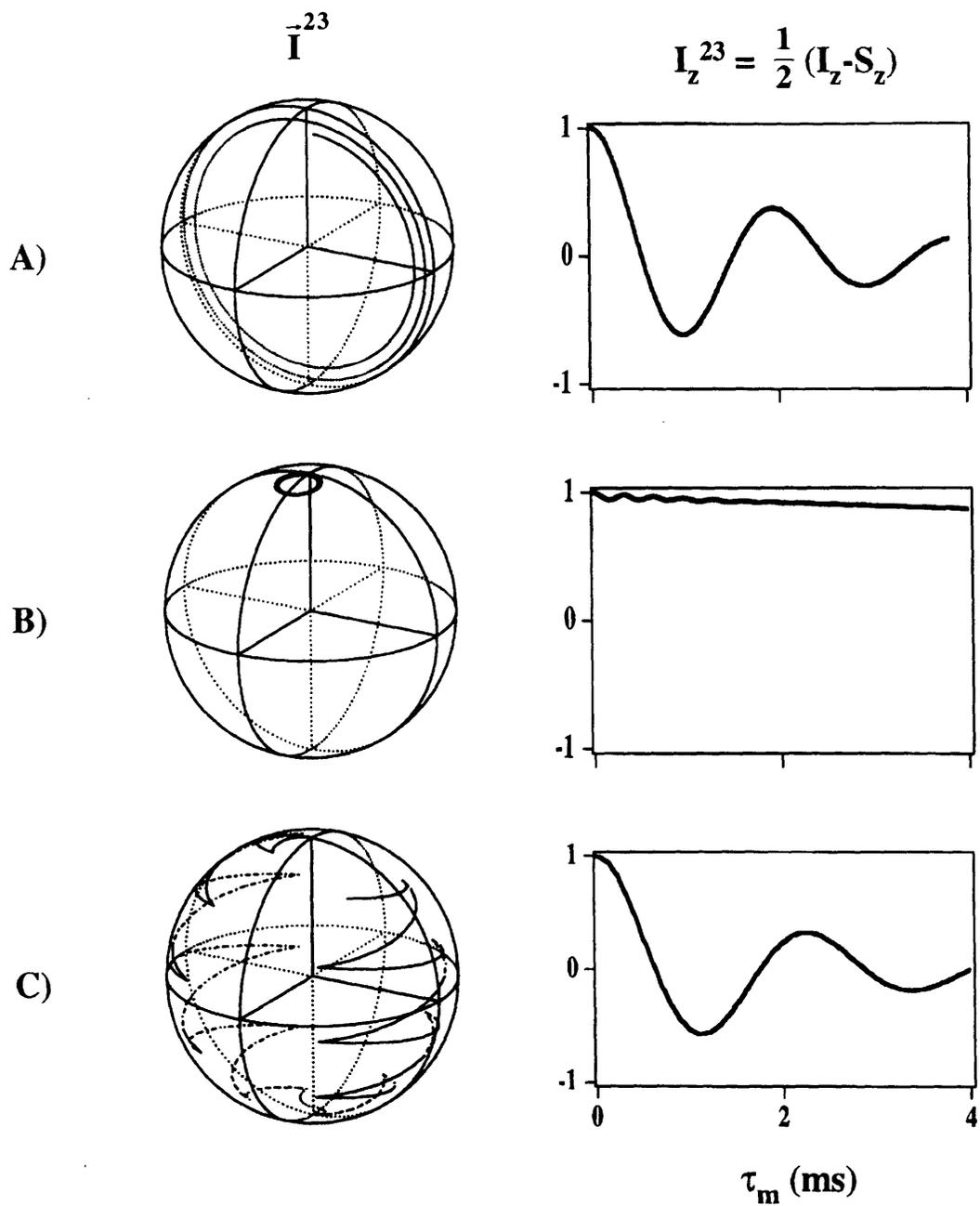


Figure 3-4: Summary of fictitious-spin dynamics. On the left, full \vec{I}^{23} trajectories; on the right, z -projections. (A) On rotational resonance, no mixing pulses. (B) Off rotational resonance, no mixing pulses. (C) Off rotational resonance, as in (B), with one π pulse applied per rotor period.

might be tempted to postulate that π pulse trains allow broadband dipolar recoupling by averaging out chemical shift differences that would otherwise frustrate magnetization exchange. This would be analogous to the mechanism of liquid-state TOCSY experiments [17]. Such a picture is incomplete for a number of reasons. First, the π pulses are repeated too slowly to suppress the chemical shift interactions. Secondly, a pure dipolar Hamiltonian would average to zero in the course of a rotor cycle, and no net exchange could accumulate from one rotor cycle to the next. It is more accurate to think of the magnetization exchange as being driven by the sample rotation, with a little assistance from the RF pulses from time to time.

Chapter 4

Conclusions

In DRAMA [7], and in the related heteronuclear REDOR [18] experiment, RF pulses are used to spoil the MAS averaging of dipole couplings. Rotational resonance experiments take advantage of rotor motion to synchronize oscillating dipole couplings with differential spin precession. When π pulses interrupt spin precession in the presence of MAS, they can serve to compensate for the offset from R^2 conditions, and broadband dipolar recoupling is possible. BDR experiments have many of the advantages of DRAMA: the effects of dipole couplings can be switched on and off at will by applying or withholding RF pulses, and multiple couplings can be measured simultaneously. The BDR phenomenon is not sensitive to inhomogeneous broadening – a common criticism of R^2 experiments – and BDR magnetization exchange experiments do not suffer from R^2 -induced broadenings and splittings of spectral lines, since the mixing pulses are absent during acquisition. The disadvantages of signal losses due to accumulated pulse errors may be allayed in part by pulse error compensation schemes.

In the theory and experiments described so far, we have concentrated upon longitudinal magnetization exchange processes. Longitudinal magnetization exchange also forms the centerpiece of liquid-state NOESY, and in fact the two-dimensional exchange experiments discussed in Section 2.2 are close analogues of NOESY, wherein the broadband modulation of dipole couplings is achieved by a combination of pulsing and sample rotation rather than by indigenous molecular tumbling. Multidimensional BDR experiments may be used, in the style of liquid state NOESY, to map out the

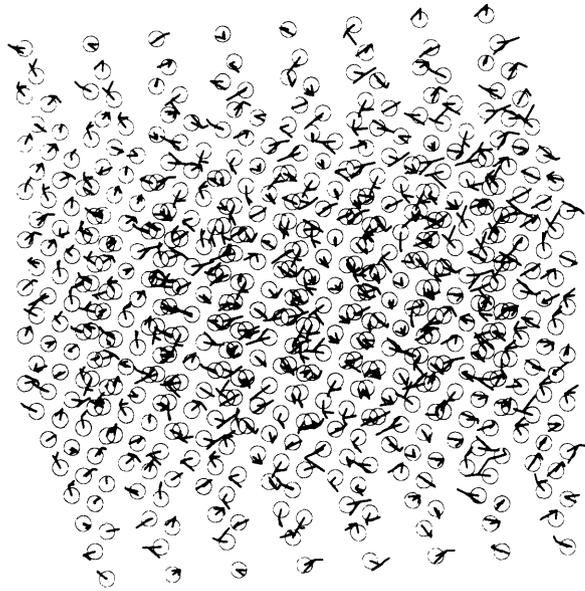
couplings of multiple spin sites in molecules of interest. Precise quantitation of these couplings, and hence determination of internuclear distances and molecular structure, will be subject to some of the same constraints as normal R^2 distance measurements. Detailed numerical simulations require some knowledge of dipolar and chemical shift tensor orientations, as well as estimates of zero-quantum relaxation rates (particularly crucial for large distances and broad resonances).¹ Nevertheless, for molecules with resolvable spin sites, one can achieve at least a qualitative NOESY-like coupling map, which can be supplemented if necessary by narrow-band R^2 experiments to place more stringent limits on the critical distances.

R^2 magnetization exchange experiments have been used to determine internuclear distances in a variety of molecules in the solid state [19][20][21][22][23], including the membrane protein bacteriorhodopsin [21][22], and a peptide fragment of the β -amyloid protein associated with Alzheimer's disease [23]. The feasibility of broadband recoupling may obviate, at least to some degree, laborious repetitions of finely-tuned R^2 measurements for one distance after another in molecules like the β -amyloid peptide. Thus, broadband dipolar recoupling adds a new degree of flexibility to the rotational resonance phenomenon, and promises to extend to solid-state NMR some of the repertoire of liquid-style structure determination.

¹For multiply labelled compounds, one can also observe the effects of relayed couplings not present in selective R^2 experiments.

Part II

Many-Spin Systems: Spin Diffusion in Lattices



Chapter 5

Introduction

5.1 Dipole Couplings in Extended Systems

When the number of spins in a coupled spin system becomes large, simple approaches such as the fictitious spin model of Chapter 3 break down. The Hamiltonian for a system of N spins $1/2$ has dimension $2^N \times 2^N$, thus unless the coupling topology is particularly simple, the quantum matrix problem becomes intractable for large N . Nevertheless, many-body effects must often be reckoned with in real systems of macroscopic extent. Consider a crystalline solid, which may be modelled for our present purposes as a regular lattice of equivalent nuclear spins.¹ We take the lattice to be rigid and stationary, so that the dipole-dipole couplings between spins are not motionally modulated. The dipole interaction then falls off as the inverse cube of the interspin distance, while in a three dimensional crystal the number of spins in a spherical shell of radius r around each spin grows approximately as r^2 . Thus, a simple integral for the resultant dipole field at each site in an infinite crystal is logarithmically divergent.² Evidently, the consideration of dipole couplings in a three-dimensional spin lattice presents us with a genuine many-body problem.

¹Calcium fluoride crystals provide a practical realization of this model, in which the magnetically active fluorine nuclei lie at the vertices of a simple cubic lattice.

²In practice, this divergence is tamed by the discrete nature of the lattice, and the problem may be treated by the techniques of Ewald summation.

5.2 Spin Diffusion

Dipole couplings in solids can drive processes like spin diffusion – the transport of magnetization and spin-spin energy by means of flip-flops between neighboring magnetic spins which are themselves fixed in space.³ A local disturbance in magnetization or energy will eventually be dispersed across the lattice through the actions of spin-spin couplings, and the rate of dissipation of the initial inhomogeneity may often be characterized by a coefficient of diffusion. Spin diffusion plays an important role in the spin-lattice relaxation of many systems of interest in solid-state NMR [24], and the spin diffusion process itself may be exploited in NMR experiments to give information about the structure of materials. As a mechanism of magnetic transport, spin diffusion stands as a many-body counterpart to the dipole-mediated exchange processes studied in Part I.⁴

Several approximate analytic theories for spatial spin diffusion exist [24]-[31] and are in rough agreement with experiment [24][32]-[39], but many of the details of the spin diffusion process remain to be explored. As a consequence, a classical simulation was recently developed to investigate the dynamics of spin diffusion in lattices [40][41]. Using a model of classical gyromagnets precessing in each others' dipole fields, Tang and Waugh derived numerical values for the spin diffusion constant in model lattices, and characterized several of the main conceptual and computational issues for classical spin diffusion. Some of their results are summarized in Table 5.1, which compares computed values of the classical spin diffusion constant with a variety of prior theoretical and experimental studies.⁵ There is good agreement in all but a few cases, which may well represent experimental difficulties rather than theoretical anomalies. The computational work of Tang and Waugh included simulations of diffusion profiles

³Spin diffusion is to be distinguished from the diffusion of spatially mobile spins. In a spin diffusion process, only magnetization is transported, rather than the spins themselves.

⁴Longitudinal magnetization exchange stands at the center of both processes, though differences arise in the details. Traditional considerations of spatial, as opposed to spectral, spin diffusion involve spins of uniform Larmor frequency. In the absence of chemical shift barriers, the dipole coupling is free to drive exchange of magnetization, with no heroics of mechanical or electromagnetic manipulation required to restore it to full potency.

⁵See Ref. [40], Chapter 1, and the introduction to Ref. [41], for a survey of the theoretical, experimental, and computational history of spin diffusion.

D_{\parallel}	D_{\perp}	$\bar{D}[001]$	$\bar{D}[111]$	Ref.
0.35	0.14	0.21	0.22	[41]*
		0.24	0.26	[28][29]
0.33	0.11	0.19	0.18	[30]
		0.20	0.036	[39]

Table 5.1: Comparative summary, taken from Ref. [41], of the diffusion of Zeeman energy on a simple cubic lattice. The diffusion always occurs along a fourfold axis of the crystal. This is either parallel (D_{\parallel}) or perpendicular (D_{\perp}) to the external field, or else the field is in the crystal [111] direction. In the last case, symmetry requires that the diffusion coefficient along any fourfold axis is the average over all directions with respect to the field. The diffusion constants shown here are measured as multiples of $|\gamma m|/r_0$, where γ is the gyromagnetic ratio, m is the magnetic moment of a single spin, and r_0 is the nearest-neighbor distance between spins in the lattice. In a CaF_2 crystal, this quantity amounts to $2.12 \times 10^{-15} m^2 s^{-1}$.

and free induction decays for dipole-coupled lattices in high applied field, simulations of diffusion in Heisenberg magnets, and diffusion studies in crystals containing two spin species. Their work also addressed the interplay of dynamics and energetics in classical spin diffusion: they found that the value of the conserved spin-spin energy had a strong influence on the diffusion of Zeeman energy. The work presented here is a continuation of the program begun in Ref.'s [40] and [41], and is designed to explore further the microscopic underpinnings of classical spin diffusion. After a brief review of computational methodology, we shall extend the classical studies to new situations of interest in solid-state NMR and condensed-matter physics. In particular, we shall study the cases 1) of spin lattices subjected to small external magnetic fields, and 2) of lattices randomly diluted with vacancies or non-magnetic impurities. One final chapter is devoted to reconciling quantum theories with classical simulations.

Chapter 6

Classical Spin Dynamics and Spin Diffusion Calculations

There are in this Earth many Maneuvers more unnerving than the simulation of classical spin dynamics. If we wish to study the behavior of a system of classical spins, we need only set up a representative model system in an initial condition of interest, then track the subsequent motion of each of the spins as they evolve under the influence of their mutual interactions. This is, in essence, how the simulations of spin diffusion are performed: after initializing a classical spin lattice to a convenient starting configuration, we calculate and store an array of individual spin orientation histories, from which we can reconstruct bulk parameters such as the spin diffusion constant. If we were interested in studying the spatial diffusion of spins engaged in translational motion (say in a liquid sample), we could simply calculate the root mean square displacement of each spin from its starting position as a function of time, and from this we could derive a diffusion constant. Since rotations rather than translations are the currency of classical spin dynamics, however, such purely microscopic methods are not available. Each spin in a fixed lattice precesses continuously about a local magnetic field generated by all of its neighbors, and it is impossible to trace the “path” of any component of individual spin orientation through the lattice. There is no “particle” of orientation. Instead, we must look to the spatial distribution of spin orientation, and study how this distribution changes with time.

6.1 The Diffusion Equation

If a certain local region in our spin system has a degree of magnetic ordering that exceeds the global equilibrium value, this local disturbance will decay at a rate that is closely related to the spin diffusion constant. Consider a lattice of N classical gyromagnets \mathbf{m}_j ($j = 1, 2, \dots, N$) in a constant magnetic field $\mathbf{B}_0 = B_0 \hat{\mathbf{z}}$ at temperature T . We may write the net z magnetization $M(\mathbf{r})$ as a function of position in the lattice, where $M(\mathbf{r}_j) = m_{jz}$ if \mathbf{r}_j is the position of spin j . The diffusion of spin magnetization M may then be described by a general diffusion equation [28][30]

$$\frac{\partial M(\mathbf{r}, t)}{\partial t} = \sum_{\alpha, \beta} D_{\alpha\beta} \frac{\partial^2 M(\mathbf{r}, t)}{\partial x^\alpha \partial x^\beta} \quad (6.1)$$

where x^α and x^β are any Cartesian coordinates. In the principal axis system of D , the diffusion equation simplifies to

$$\frac{\partial M(\mathbf{r}, t)}{\partial t} = \sum_{\mu} D_{\mu\mu} \frac{\partial^2 M(\mathbf{r}, t)}{(\partial x^\mu)^2} \quad (6.2)$$

The spatial Fourier transform of (6.2) is

$$\frac{\partial A(\mathbf{k}, t)}{\partial t} = - \sum_{\mu} (k^\mu)^2 D_{\mu\mu} A(\mathbf{k}, t) = -k^2 D_k A(\mathbf{k}, t) \quad (6.3)$$

in which A denotes the amplitude of the magnetization component with wavevector \mathbf{k} , and $D_k \equiv k^{-2} \sum_{\mu} (k^\mu)^2 D_{\mu\mu}$ is the spin diffusion constant along the direction of wavevector \mathbf{k} . To calculate D_k , it is sufficient to follow a single Fourier component of the initial disturbance. If the dynamics are truly diffusive, we expect any such Fourier component to decay exponentially over time with rate constant $k^2 D_k$.

6.2 The Initial Lattice Configuration

The first task in the classical simulations is to choose a model lattice and to set up the initial profile of spin magnetization or energy whose subsequent time evolution

we will observe. Once a template lattice geometry has been selected, a classical spin is placed at each vertex in the lattice (or, in the case of the dilution studies in Section 7.2, at a randomly selected subset of the vertices). Each spin is labelled by a set of d indices $\{l^1, l^2, \dots, l^d\}$, where d is the dimension of the lattice and the values of the indices identify the position of the spin along the unit cell directions. For almost all of the simulations to be presented here, a simple cubic lattice geometry was used, which means that the lattice indices correspond directly to cartesian coordinates $\{x^1, x^2, \dots, x^d\}$.

After the spins have been placed in the lattice, they must be oriented so as to yield the desired initial condition. Since, on the merits of Eq. (6.3), we will be tracking the decay of a Fourier component of the initial disturbance, it is simplest and most efficient to use a disturbance which consists of a single Fourier component. If the diffusion of magnetization is to be studied, a cosine wave of spin polarization is established along a given axis x^α in the crystal by choosing a cosinusoidal target magnetization $M(x^\alpha) = A_0 \cos(kx^\alpha)$ and sampling spin orientation according to a near-Boltzmann probability distribution around this target. The result is a representative classical spin lattice whose *average* polarization along the chosen axis x^α has the desired initial profile, but whose spins individually obey Boltzmann statistics. It is the average polarization amplitude that we will follow as the system evolves.

The choice of disturbance amplitude deserves some comment. Since the spins in our model lattice are oriented statistically, we expect fluctuations in the polarization profile. For Avogadro's number of spins, these fluctuations would be of no concern, but for particle numbers commensurate with computer memories (e.g. 16^3 to 64^3 for 3D lattices), the polarization "noise" may be substantial, especially at the high average temperatures required for realistic simulations of typical NMR experiments. For reasons of computational efficiency, we would like to use disturbance amplitudes that lie well above the thermal noise level without upsetting the high-temperature behavior we wish to study. It turns out to be possible to do this. Tang and Waugh [40][41] have noted that the total spin-spin energy of the initial lattice configuration is an essential determinant of the spin diffusion constant. In fact, as long as the interspin

energy is properly constrained, the value of the magnetization diffusion constant does not vary significantly with the disturbance amplitude, and large polarizations can be used without deviating from “high-temperature” diffusion behaviors. Thus, initial amplitudes on the order of 0.5 will not be uncommon in the sections to follow, but in all cases the interspin energies have been adjusted to conform to the high-temperature limit. (For details on the treatment of interspin energy, see Ref. [40]).

When it is the diffusion constant of interspin energy that we seek, we can set up a cosinusoidal energy profile analogous to the magnetization profile discussed above. The precise form the energy will take depends upon the particulars of the spin-spin interaction, which will be our next topic. As it happens, both the value and the distribution of interspin energy can be adjusted by establishing suitable correlations among the individual spin magnetization components. (Again, see Ref. [40] for details.)

6.3 Time Evolution

Now that the lattice has been suitably initialized, we must propagate it forward in time to observe the fate of the initial disturbance amplitude. Microscopically, each of the spins \mathbf{m}_j evolves according to the classical Bloch equations:

$$\frac{d\mathbf{m}_j}{dt} = \gamma \mathbf{m}_j \times \mathbf{B}_j \quad (6.4)$$

where γ is the gyromagnetic ratio. \mathbf{B}_j is the local magnetic field experienced by spin \mathbf{m}_j , and is comprised of the net dipole field from the other spins in the lattice plus any external field \mathbf{B}_0 we choose to apply:

$$\mathbf{B}_j = \mathbf{B}_0 + \sum_{k \neq j} \mathbf{B}_{jk} \quad (6.5)$$

The dipole field \mathbf{B}_{jk} produced by a spin \mathbf{m}_k and felt at spin \mathbf{m}_j has the familiar form

$$\mathbf{B}_{jk} = \frac{1}{r_{jk}^3} \{3(\mathbf{m}_k \cdot \hat{\mathbf{r}}_{jk}) \hat{\mathbf{r}}_{jk} - \mathbf{m}_k\} \quad (6.6)$$

where $\mathbf{r}_{jk} = \mathbf{r}_j - \mathbf{r}_k$ is the interspin vector. In a strong external magnetic field, applied conventionally along the $\hat{\mathbf{z}}$ direction, \mathbf{B}_{jk} may be truncated to

$$\mathbf{B}_{jk} = b(\mathbf{r}_{jk}) \{m_{k_x}\hat{\mathbf{x}} + m_{k_y}\hat{\mathbf{y}} - 2m_{k_z}\hat{\mathbf{z}}\} \quad (6.7)$$

$$b(\mathbf{r}_{jk}) = -\frac{1}{2r_{jk}^3} \{3(\hat{\mathbf{r}}_{jk} \cdot \hat{\mathbf{z}})^2 - 1\} = -\frac{1}{r_{jk}^3} P_2(\cos \theta_{jk}) \quad (6.8)$$

The remaining effects of the external field may be removed from the problem by transforming to a suitable rotating frame.¹

Equations (6.4) to (6.8) indicate that as time passes, each spin will precess about a field which is itself varying in time through its dependence upon the orientations of the other spins in the lattice. The Bloch equation (6.4) is actually a representative of a set of N coupled differential equations. Our next task, therefore, is to solve this set of equations – a numerically formidable if conceptually simple task for large N . First, the calculation is divided into small time steps. Beginning with the initial set of orientations $\{\mathbf{m}_k(0)\}$, the local field \mathbf{B}_j at each spin site j is calculated as

$$\mathbf{B}_j = \sum_{k \neq j} \mathbf{B}_{jk} \quad (6.9)$$

with \mathbf{B}_{jk} given by Eq. (6.6) or Eq. (6.7) above. Since the local field calculation is to be repeated at each time step, a full lattice summation for each spin site would be quite costly in computation time for the large lattices we wish to study. One may begin to address this problem by observing that the field in Eq. (6.9) may be written as a convolution. For the high-field case, we have

$$\begin{aligned} \mathbf{B}_j &= \sum_k b(\mathbf{r}_j - \mathbf{r}_k) (M_x(\mathbf{r}_k)\hat{\mathbf{x}} + M_y(\mathbf{r}_k)\hat{\mathbf{y}} - 2M_z(\mathbf{r}_k)\hat{\mathbf{z}}) \\ &= [b \otimes (M_x\hat{\mathbf{x}} + M_y\hat{\mathbf{y}} - 2M_z\hat{\mathbf{z}})]_j \end{aligned} \quad (6.10)$$

¹We are assuming that all the spins in the lattice have identical chemical shifts, so that in the absence of dipole interactions, they would all precess about \mathbf{B}_0 at precisely the same rate. This uniform precession will not influence the spin diffusion constant, and we may eliminate it with impunity. Thus, the external field may be taken to influence spin diffusion solely through its averaging effects on the dipole-dipole interactions.

\mathbf{M} is the lattice of spin orientations, and b an array of couplings from one origin spin to each of the others. (As usual, we have used the symbol $\mathbf{M}(\mathbf{r})$ ($\mathbf{M}(\mathbf{r}_k) = \mathbf{m}_k$) to make the spatial dependence of the magnetization clear.) The d -dimensional convolution of these two arrays produces the desired array of local fields. If we are willing to impose cyclic boundary conditions on our lattice (such that a spin at one edge of the lattice is taken to be adjacent to the corresponding spin at the opposite edge, and the coupling matrix b “wraps around” in space), then the convolution may be accomplished quite efficiently using fast fourier transforms:

$$b \otimes M_\alpha = \mathcal{F}^{-1} [\mathcal{F}[b] \cdot \mathcal{F}[M_\alpha]] \quad (\alpha = x, y, z) \quad (6.11)$$

The symbol \mathcal{F} above indicates a fourier transform along each of the unit cell directions in turn.

In zero applied field, the form of the coupling is slightly more complicated, but the same principle applies. We may write

$$\begin{aligned} \mathbf{B}_j &= \sum_k \sum_{\alpha, \beta} \hat{\alpha} b_{\alpha\beta}(\mathbf{r}_j - \mathbf{r}_k) M_\beta(\mathbf{r}_k) \\ &= \sum_{\alpha, \beta} \hat{\alpha} [b_{\alpha\beta} \otimes M_\beta]_j \quad (\alpha, \beta = x, y, z) \end{aligned} \quad (6.12)$$

with

$$\begin{aligned} b_{\alpha\alpha}(\mathbf{r}) &= 3\hat{\mathbf{r}}_\alpha^2 - 1 \\ b_{\alpha\beta}(\mathbf{r}) &= b_{\beta\alpha}(\mathbf{r}) = 3\hat{\mathbf{r}}_\alpha\hat{\mathbf{r}}_\beta \end{aligned} \quad (6.13)$$

and we may accomplish the local field computation with nine convolutions as opposed to three in the high field case.

With the net local field calculated for each spin, it remains only to cause the spins to precess in their local fields for the duration of a time step. While a straightforward rotation-matrix approach might not at first seem objectionable, computational efficiency can be further increased by one additional tactic. Taking sequential derivatives

of the Bloch equations (6.4) (and denoting the n^{th} derivative by the superscript (n)) we find

$$\begin{aligned}
\mathbf{m}_j^{(0)} &= \gamma \mathbf{m}_j \\
\mathbf{m}_j^{(1)} &= \gamma \mathbf{m}_j^{(0)} \times \mathbf{B}_j^{(0)} \\
\mathbf{m}_j^{(2)} &= \gamma \left(\mathbf{m}_j^{(0)} \times \mathbf{B}_j^{(1)} + \mathbf{m}_j^{(1)} \times \mathbf{B}_j^{(0)} \right) \\
\mathbf{m}_j^{(3)} &= \gamma \left(\mathbf{m}_j^{(0)} \times \mathbf{B}_j^{(2)} + 2\mathbf{m}_j^{(1)} \times \mathbf{B}_j^{(1)} + \mathbf{m}_j^{(2)} \times \mathbf{B}_j^{(0)} \right) \\
&\vdots \\
\mathbf{m}_j^{(n+1)} &= \gamma \sum_{m=0}^n \binom{n}{m} \mathbf{m}_j^{(m)} \times \mathbf{B}_j^{(n-m)}
\end{aligned} \tag{6.14}$$

Since the couplings do not change with time, $\mathbf{B}_j^{(n-m)}$ simply represents combinations of the $(n-m)^{\text{th}}$ derivatives of spin magnetizations according to Eq.'s (6.6) or (6.7). That is,

$$\mathbf{B}_j^{(n-m)} = \sum_k \frac{1}{r_{jk}^3} \left\{ 3 \left(\mathbf{m}_k^{(n-m)} \cdot \hat{\mathbf{r}}_{jk} \right) \hat{\mathbf{r}}_{jk} - \mathbf{m}_k^{(n-m)} \right\} \tag{6.15}$$

or

$$\mathbf{B}_j^{(n-m)} = \sum_k b(\mathbf{r}_{jk}) \left\{ m_{k_x}^{(n-m)} \hat{\mathbf{x}} + m_{k_y}^{(n-m)} \hat{\mathbf{y}} - 2m_{k_z}^{(n-m)} \hat{\mathbf{z}} \right\} \tag{6.16}$$

Thus, the time derivatives of $\mathbf{m}_j(t)$ to arbitrary order may be calculated recursively from the current spin orientations $\{\mathbf{m}_k(t)\}$. By evaluating a Taylor series with these derivatives, we may determine the motion of any spin in the lattice with arbitrary (and adjustable) accuracy. This procedure affords an appealing degree of control over the progress of the simulation [40]. Furthermore, by storing the first derivative of the magnetization trajectories as well as the trajectories themselves, we gain an additional benefit. Knowing the values of $\{d\mathbf{m}_k(t)/dt\}$, we can immediately calculate the first derivative of the disturbance amplitude $\partial A(\mathbf{k}, t)/\partial t$, and the spin diffusion

constant may be determined from

$$D_k = -\frac{\partial A(\mathbf{k}, t)/\partial t}{k^2 A(\mathbf{k}, t)} \quad (6.17)$$

6.4 Practicalities – a Sample Simulation

To summarize, the classical simulations proceed according to the following general outline:

1. Populate a lattice with spins, then orient the spins to produce an initial sinusoidal deviation from equilibrium.
2. Propagate the spins incrementally in time according to the Bloch equations, calculating the local field for each spin, then causing each spin to precess in its local field for one time step.
3. Repeat the propagation procedure for as many time steps as are needed, storing the amplitude of the disturbance and its first time derivative at each instant.
4. Calculate the diffusion constant from Eq. (6.17).

Results of a typical simulation for the diffusion of magnetization are included in Figures 6-1 and 6-2. Figure 6-1A shows the disturbance amplitude as a function of time. As expected, the decay is for the most part exponential. It is evident from Eq. (6.17) that an instantaneous spin diffusion constant $D_k(t)$ can be defined at each time t . This diffusion constant is plotted on the same time scale in Figure 6-1B. After a brief induction time, which may be taken to be the time required for realistic correlations to develop among the interacting spins, $D(t)$ fluctuates around an essentially constant average value, with the size of the fluctuations growing as the disturbance amplitude diminishes. For illustrative purposes, the decay of the actual magnetization profile as projected onto the diffusion axis $\hat{\mathbf{k}}$ is displayed in Figure 6-2. The effects of spin diffusion are apparent.

For reasons of convenience and generality, reduced units have been used in all the calculations. The nearest-neighbor distance r_0 and the magnetic moment m of

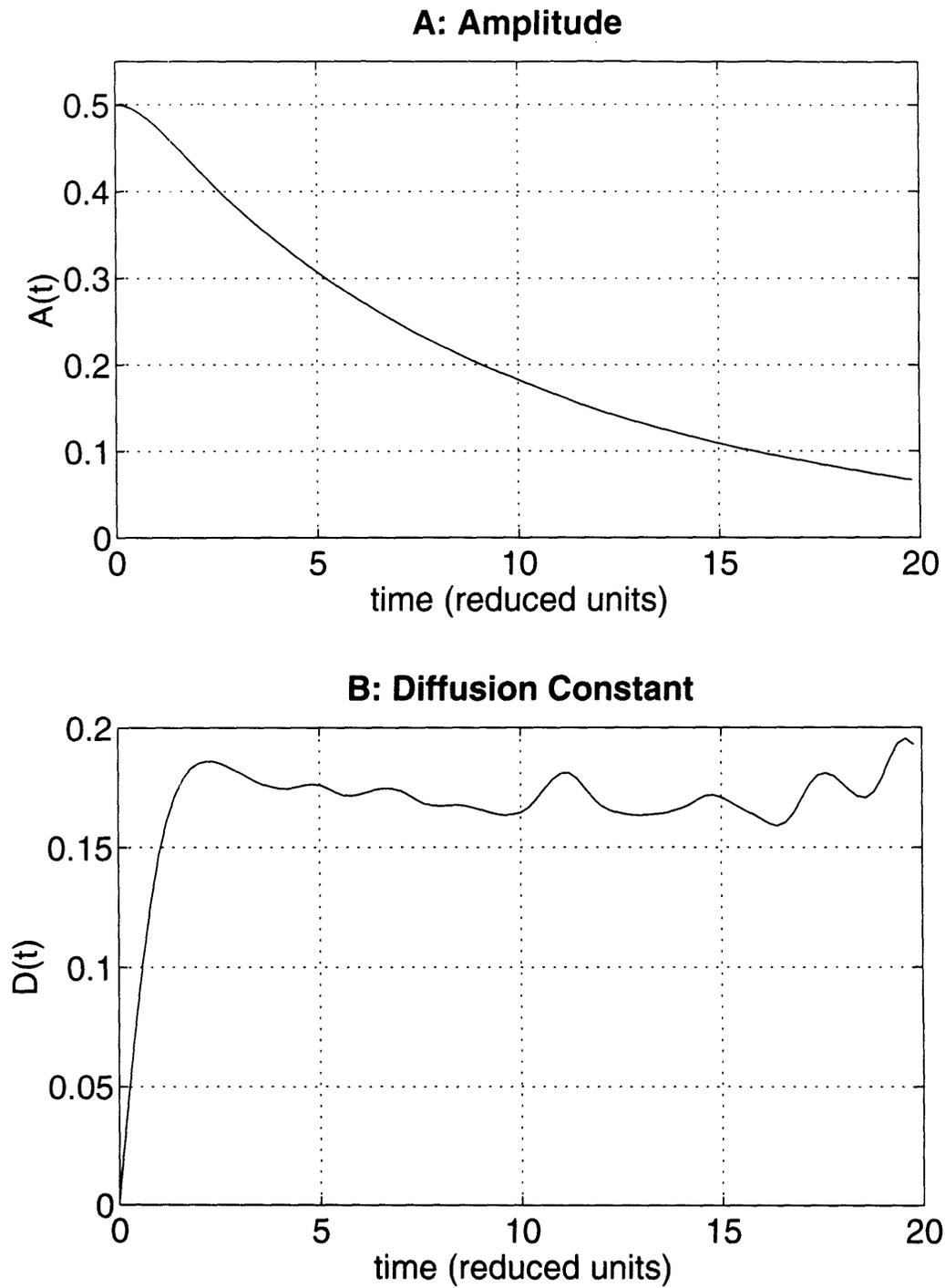


Figure 6-1: Output of a sample simulation of spin diffusion. Displayed curves are averages of 64 runs for a simple cubic lattice of $N = 16 \times 16 \times 16$ spins in high external magnetic field. The field was applied along the $[111]$ lattice direction. The initial magnetization profile was cosinusoidal with wavelength $\lambda = 8$.

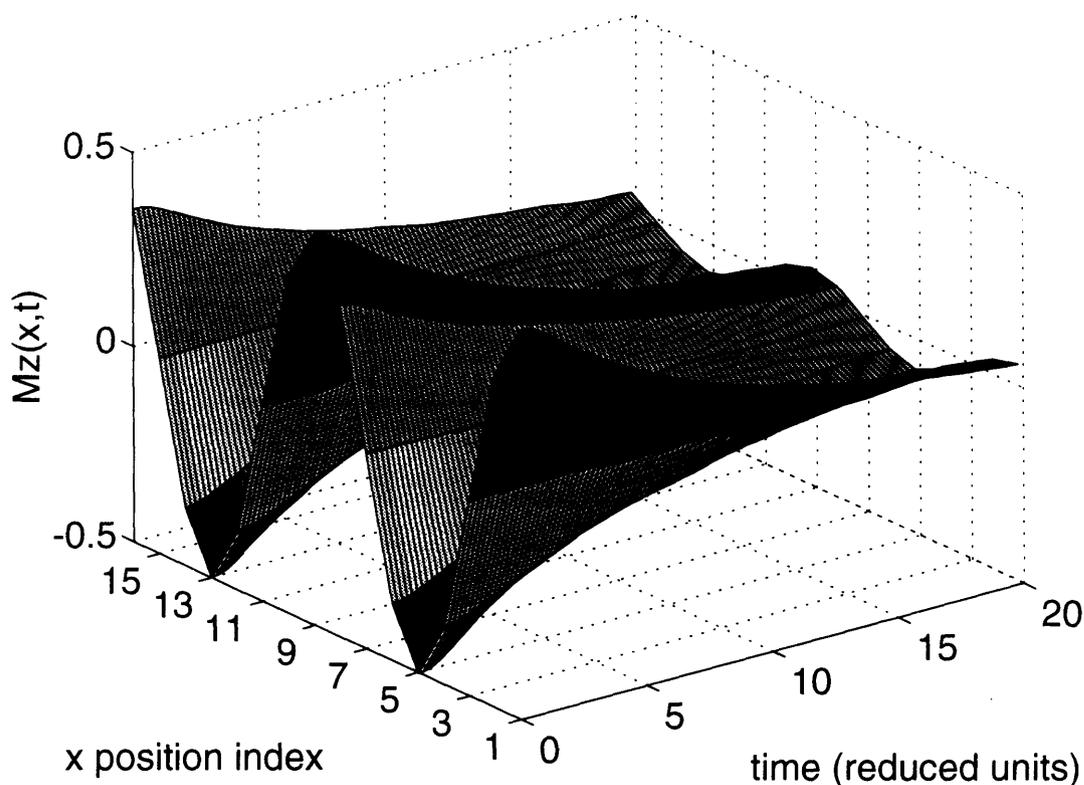


Figure 6-2: A surface plot showing the decay of the cosinusoidal magnetization profile $M(x,t)$.

a single spin are scaled to unity, as is the gyromagnetic ratio γ . Time, as displayed in Figures 6-1 and 6-2, is measured in units of $r_0^3/|\gamma m|$ (the inverse of the nearest-neighbor dipole coupling strength). Diffusion constants have units of $|\gamma m|/r_0$. In a CaF_2 crystal, these units amount to $35.1 \times 10^{-6} \text{s}$ and $2.12 \times 10^{-15} \text{m}^2 \text{s}^{-1}$ respectively.² Numerical values for D have been obtained as the time average of instantaneous $D(t)$ curves from a starting time after the initial induction to an end time before fluctuations become extreme. Where precise values are required, or where fluctuations must be suppressed, calculations have been repeated for an ensemble of statistically equivalent initial conditions, and the results have been averaged. Error bars indicated the estimated standard error of the mean (i.e. the standard deviation divided by the square root of the number of trials).

²For the ^{19}F nuclei in CaF_2 , $r_0 = 2.7 \text{\AA}$, $m = 4.55 \times \mu_N$, and $\gamma = 25.18 \times 10^7 \text{T}^{-1} \text{s}^{-1}$.

Details of the core simulation code may be found in the thesis of Changguo Tang [40], along with careful studies of error propagation and assorted parameter dependencies. Additional information on modifications of the basic simulation will be provided as needed in sections to follow. Tang's original calculations were performed on an IBM 3090 supercomputer. For the studies to be reported here, an SGI Indigo workstation was found to be sufficiently fast and versatile. Unlike Tang's original Fortran code, these simulations were implemented in Matlab 4.0, which boasts a parsimonious programming language and a convenient graphical interface. Matlab code for the simulation of classical spin diffusion is included in Appendix B.

Chapter 7

Classical Spin Diffusion: Results of Simulations

Since the classical calculations of Tang and Waugh seem to be in good agreement both with experiment and with approximate analytical theories¹, we may be emboldened to use classical simulations to explore facets of spin diffusion which are not so accessible to experiment, but which shed light upon the microscopic foundations of the phenomenon. Two classes of numerical experiments in particular serve to bring into relief the detailed mechanisms of magnetization transport in dipole-coupled spin systems. First, by comparing spin diffusion in the two extremes of high and low applied magnetic field, we may identify certain basic prerequisites for diffusive spin dynamics. We find that in the absence of a suitable conservation principle, the evolution of spin magnetization does not follow a truly diffusive profile. Second, a study of the effects of spin concentration and lattice geometry upon the spin diffusion constant yields insight into the interplay of local and global effects in the spin lattice. Lattice dilution has a marked influence on the spin diffusion constant, and this influence scales with dimension in a manner reminiscent of percolation phenomena.

¹cf. Ref.'s [40] and [41] and Section 5.2 above

7.1 Spin Diffusion and the Effects of Applied Magnetic Field

Tang and Waugh have shown that in the presence of a strong applied magnetic field, both magnetization and interspin energy display demonstrably diffusive dynamics. That this is so is perhaps most easily ascertained by comparing results for different wavelengths of the initial magnetization or energy profile. As we saw in Chapter 6, the rate of decay of the initial profile is expected to scale as $k^2 = (2\pi/\lambda)^2$ if a diffusion equation is obeyed, in which case a single diffusion constant can be defined for all wavelengths using Eq. (6.17). Figures 7-1 and 7-2 contain results from Changguo Tang's thesis showing that for sufficiently large values of λ , $A(t)$ (the magnetization amplitude, Figure 7-1) and $A_E(t)$ (the amplitude of interspin energy, Figure 7-2) scale appropriately with wavelength. A long-wavelength limit is reached in which the calculated diffusion coefficient is indeed independent of λ .

In zero applied magnetic field, similar results are obtained for the diffusion of interspin energy. Figure 7-3 shows the scaling of $A_E(t)$ and $D_E(t)$ over four octaves from $\lambda = 4$ to $\lambda = 32$. The diffusion constant (in reduced units) is on the order of unity for all the wavelengths tested. By contrast, the transport of spin magnetization is *not* diffusive in zero field, as is demonstrated by Figure 7-4. For wavelengths from $\lambda = 4$ to $\lambda = 32$, the spin polarization amplitude $A(t)$ decays at a uniform rate, and no single diffusion constant can be defined.

In attempting to explain this discrepancy between high-field and zero-field behaviors, it behooves us to look more closely at the makeup of the dipole coupling in high and low field. In high applied field, the dipole-dipole Hamiltonian is truncated to the following form (in the rotating frame of the external field):

$$\begin{aligned}
 \mathcal{H} &= -\frac{1}{2} \sum_j \mathbf{m}_j \cdot \mathbf{B}_j = -\frac{1}{2} \sum_{j,k \neq} m_{j_x} m_{k_x} + m_{j_y} m_{k_y} - 2m_{j_z} m_{k_z} \\
 &= -\frac{1}{2} \sum_{j,k \neq} \frac{1}{2} \{ m_{j_+} m_{k_-} + m_{j_-} m_{k_+} \} - 2m_{j_z} m_{k_z}
 \end{aligned} \tag{7.1}$$

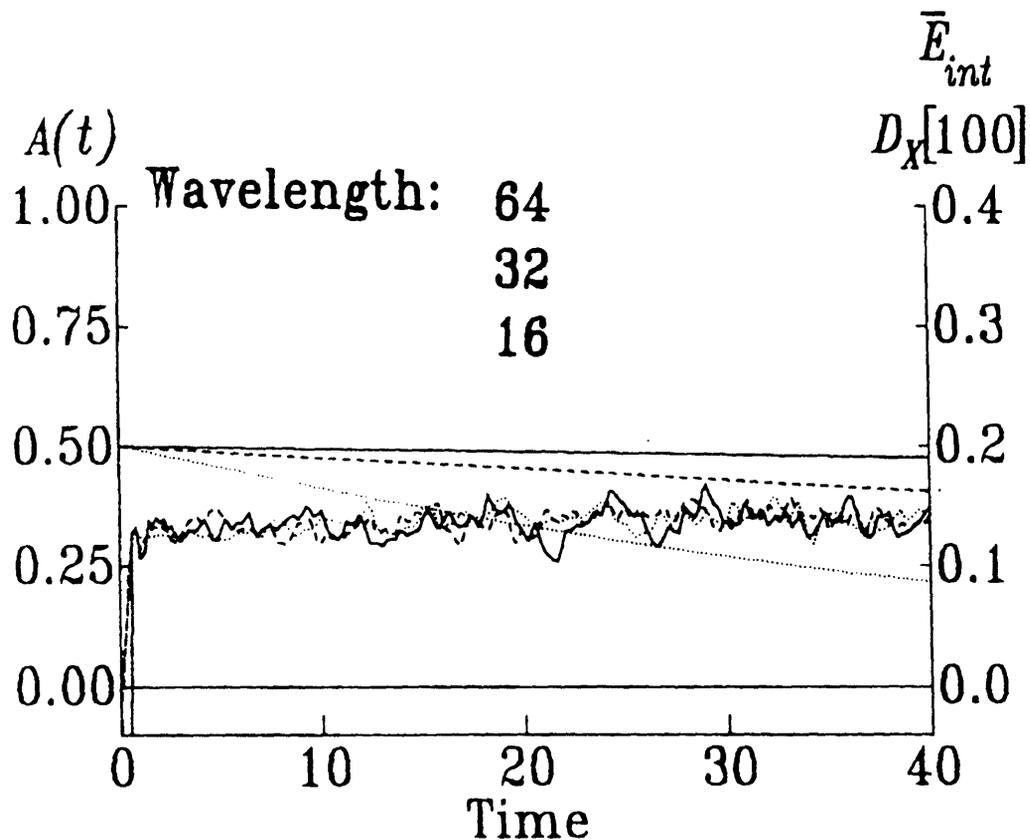


Figure 7-1: Wavelength dependence for high-field diffusion of magnetization (reproduced from Ref. [40]). Diffusion was measured along the field direction [100], for a lattice of $N = 64 \times 64 \times 64$ spins with a variety of disturbance wavelengths (solid line for $\lambda = 64$, dashed line for $\lambda = 32$, and dotted line for $\lambda = 16$). The left-hand axis measures the disturbance amplitude, which begins at $A(0) = 0.5$. The right-hand axis gives the scale for the diffusion constant and the average interspin energy (the latter remains fixed near $\bar{E}_{int} = 0$).

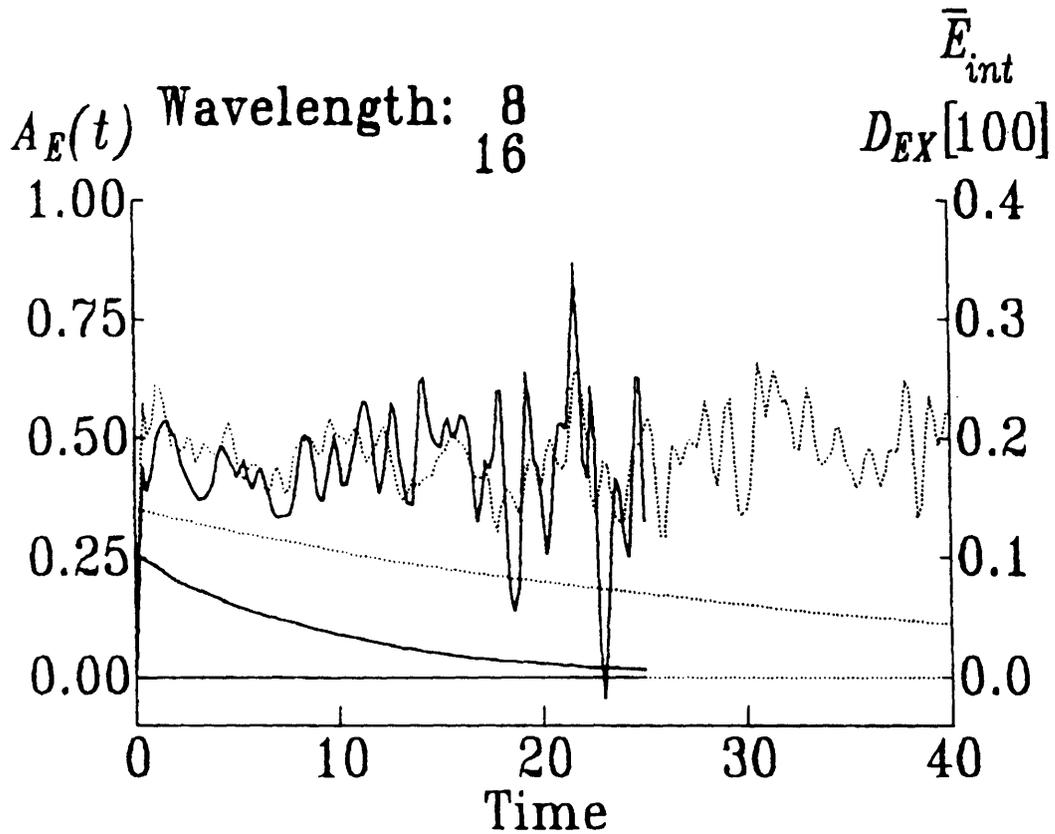


Figure 7-2: Wavelength dependence for high-field diffusion of interspin energy (reproduced from Ref. [40]). A lattice similar to that in Fig. 7-1 was used, with an initial distribution function of interspin energy characterized by wavelengths $\lambda = 8$ (solid line) and $\lambda = 16$ (dotted line).

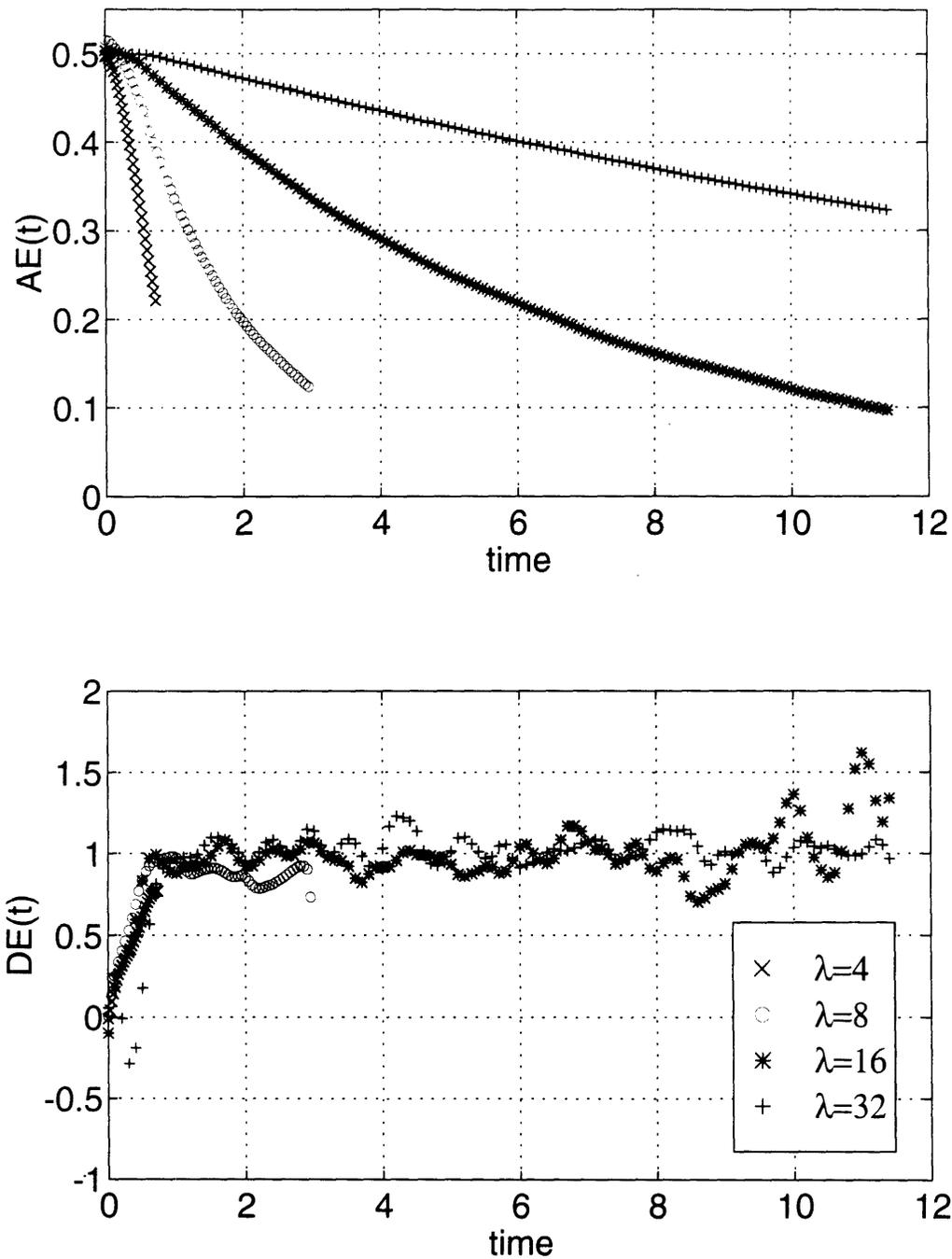


Figure 7-3: Wavelength dependence of disturbance amplitude $A_E(t)$ and diffusion constant $D_E(t)$ for interspin energy in zero applied field. The field is oriented parallel to the diffusion axis [100]. Plotted curves are averages of 64 runs for a simple cubic lattice of $N = 16 \times 16 \times 16$ spins for $\lambda = 4, 8, 16$, and $N = 32 \times 16 \times 16$ for $\lambda = 32$.

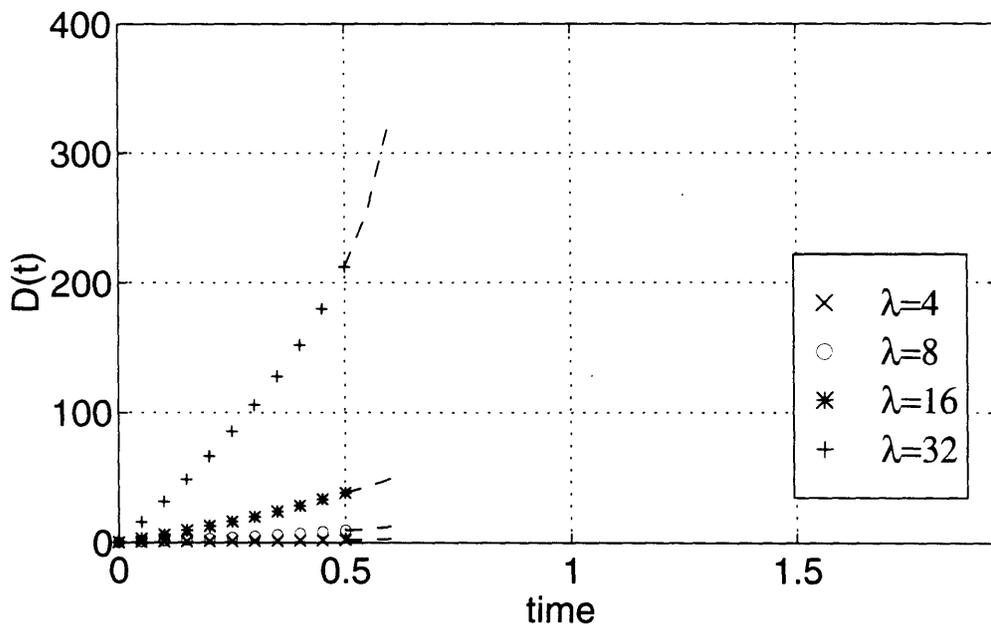
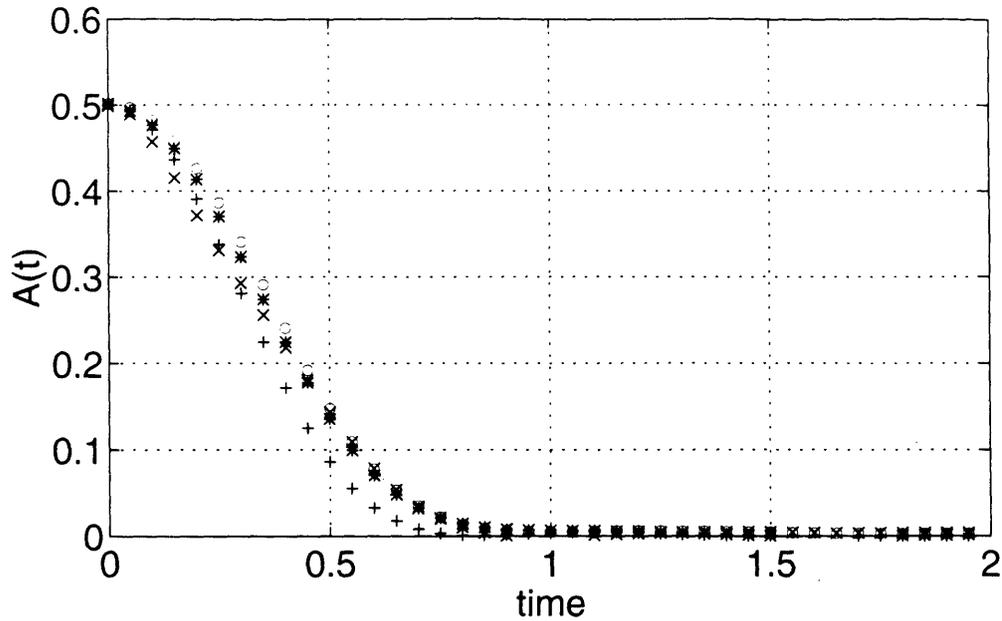


Figure 7-4: Wavelength dependence of disturbance amplitude $A(t)$ and diffusion constant $D(t)$ for magnetization in zero applied field. As in Fig. 7-3, the field is oriented along $[100]$, and the curves are averages of 64 runs using a simple cubic lattice of $N = 16 \times 16 \times 16$ spins for $\lambda = 4, 8, 16$, and $N = 32 \times 16 \times 16$ for $\lambda = 32$.

Classically, \mathcal{H} is simply the energy of the spin system (minus the Zeeman energy), and it corresponds dynamically to a precession of each spin in the local field given by Eq. (6.7). The raising and lowering operator notation familiar from quantum mechanics, $m_{\pm} \equiv m_x \pm im_y$, is used to emphasize that while the spins rotate about each other and magnetization is exchanged across the lattice, the total z component of magnetization is always preserved. Any spin j that is “flipped” upward is compensated for by downward flips in the set of spins $\{k\}$.² Indeed, the presence of a strong applied field serves to identify the z component of magnetization with a large energy – the Zeeman energy – and the high-field truncation is designed such that only motions conserving this energy are allowed. In high field, then, the z component of spin angular momentum is effectively conserved.³ Although it is customary to consider spin systems from the standpoint of energy conservation alone, we shall see that it will be advantageous to attend as well to the conservation of angular momentum when investigating the action of the dipole coupling in zero field.

In the absence of an applied magnetic field, the dipole Hamiltonian takes its full untruncated form

$$\begin{aligned} \mathcal{H} &= -\frac{1}{2} \sum_j \mathbf{m}_j \cdot \mathbf{B}_j = -\frac{1}{2} \sum_{j,k} \frac{1}{r_{jk}^3} (3(\mathbf{m}_j \cdot \hat{\mathbf{r}}_{jk})(\mathbf{m}_k \cdot \hat{\mathbf{r}}_{jk}) - \mathbf{m}_j \cdot \mathbf{m}_k) \\ &= -\frac{1}{2} \sum_{j,k} \sum_q R_{jk}^{(q)} T_{jk}^{(q)} \end{aligned} \quad (7.2)$$

²Equivalently, we may observe that, by symmetry,

$$\begin{aligned} \frac{d}{dt} \sum_j m_{jz} &= \sum_j \frac{dm_{jz}}{dt} = \gamma \sum_j (\mathbf{m}_j \times \mathbf{B}_j)_z \\ &= \gamma \sum_j \sum_k b(\mathbf{r}_{jk}) (m_{jx} m_{ky} - m_{jy} m_{kx}) = \gamma \sum_j \sum_k b(\mathbf{r}_{kj}) (m_{kx} m_{jy} - m_{ky} m_{jx}) = 0 \end{aligned}$$

³Due to the torque provided by the external field, we do not expect conservation of the transverse components of angular momentum.

with

$$\begin{aligned}
R_{jk}^{(0)} &= \frac{1}{r_{jk}^3} \left\{ m_{j_z} m_{k_z} - \frac{1}{4} (m_{j_+} m_{k_-} + m_{j_-} m_{k_+}) \right\}, & T_{jk}^{(0)} &= 1 - 3 \cos^2 \theta_{jk}, \\
R_{jk}^{(1)} &= -\frac{3}{2r_{jk}^3} (m_{j_z} m_{k_+} + m_{j_+} m_{k_z}), & T_{jk}^{(1)} &= \sin \theta_{jk} \cos \theta_{jk} \exp \{-i\phi_{jk}\}, \\
R_{jk}^{(-1)} &= -\frac{3}{2r_{jk}^3} (m_{j_z} m_{k_-} + m_{j_-} m_{k_z}), & T_{jk}^{(-1)} &= \sin \theta_{jk} \cos \theta_{jk} \exp \{+i\phi_{jk}\}, \\
R_{jk}^{(2)} &= -\frac{3}{4r_{jk}^3} m_{j_+} m_{k_+}, & T_{jk}^{(2)} &= \sin^2 \theta_{jk} \exp \{-2i\phi_{jk}\}, \\
R_{jk}^{(-2)} &= -\frac{3}{4r_{jk}^3} m_{j_-} m_{k_-}, & T_{jk}^{(-2)} &= \sin^2 \theta_{jk} \exp \{+2i\phi_{jk}\}
\end{aligned} \tag{7.3}$$

Now \mathcal{H} is the entire energy of the system: there is no Zeeman energy and no preferred direction of spin alignment. Along with the high-field “flip-flop” terms $m_{j_{\pm}} m_{k_{\mp}}$, the Hamiltonian now contains terms like $m_{j_{\pm}} m_{k_z}$ and $m_{j_z} m_{k_{\pm}}$ which represent uncompensated single or double spin “flips.” There would seem to be no intrinsic reason that any component of spin magnetization should be conserved under the action of this Hamiltonian. One might argue, however, that the zero-field dipole coupling is a purely internal interaction, which can exert no net torque on the system as a whole. The total angular momentum, and the magnitude of each of its components, should therefore be conserved. Let us see if this is truly the case. Consider for the moment a simplified two-spin system, \mathbf{m}_1 and \mathbf{m}_2 , with

$$\begin{aligned}
\frac{d\mathbf{J}_{spin}}{dt} &= \frac{d}{dt} (\mathbf{m}_1 + \mathbf{m}_2) = \gamma (\mathbf{m}_1 \times \mathbf{B}_1 + \mathbf{m}_2 \times \mathbf{B}_2) \\
&= \frac{\gamma}{r_{12}^3} (\mathbf{m}_1 \times (3\hat{\mathbf{r}}_{12} (\mathbf{m}_2 \cdot \hat{\mathbf{r}}_{12}) - \mathbf{m}_2) + \mathbf{m}_2 \times (3\hat{\mathbf{r}}_{12} (\mathbf{m}_1 \cdot \hat{\mathbf{r}}_{12}) - \mathbf{m}_1)) \\
&= \frac{3\gamma}{r_{12}^3} ((\mathbf{m}_1 \times \hat{\mathbf{r}}_{12}) (\mathbf{m}_2 \cdot \hat{\mathbf{r}}_{12}) + (\mathbf{m}_2 \times \hat{\mathbf{r}}_{12}) (\mathbf{m}_1 \cdot \hat{\mathbf{r}}_{12})) \\
&\neq 0
\end{aligned} \tag{7.4}$$

For a general orientation of the two dipoles \mathbf{m}_1 and \mathbf{m}_2 , it appears that angular momentum is not conserved! Perhaps there is something unnerving to be found in classical spin dynamics after all.

We can get no assistance here from quantum mechanics, for the quantum mechanical commutator of two coupled spin operators \mathbf{I}_1 and \mathbf{I}_2 gives the same result:

$$\begin{aligned}\frac{d\mathbf{J}_{spin}}{dt} &= \frac{d}{dt} (\mathbf{I}_1 + \mathbf{I}_2) = \frac{i}{\hbar} [\mathcal{H}, \mathbf{I}_1 + \mathbf{I}_2] \\ &= \frac{3\gamma}{r_{12}^3} ((\mathbf{I}_1 \times \hat{\mathbf{r}}_{12}) (\mathbf{I}_2 \cdot \hat{\mathbf{r}}_{12}) + (\mathbf{I}_2 \times \hat{\mathbf{r}}_{12}) (\mathbf{I}_1 \cdot \hat{\mathbf{r}}_{12}))\end{aligned}\quad (7.5)$$

Of course, if the conservation of angular momentum were truly to fail, a great deal more than the diffusion equation for spin magnetization would be at stake. Entire cottage industries might arise to advertise the universe's most fashionable directions. To this point we have been focusing solely upon the rotational motion of spins, but our rescue ultimately comes at the hands of coordinate space. The net torque generated by the dipolar interaction between the two spins serves to rotate their interspin vector \mathbf{r}_{12} such that the orbital angular momentum $\mathbf{L}_{coord} = \mathbf{r}_{12} \times \mathbf{p}_{12}$ compensates for the change in spin angular momentum. It is straightforward to derive this result quantum mechanically, using the commutation relations of \mathbf{r}_{12} and \mathbf{p}_{12} :

$$\begin{aligned}\frac{d\mathbf{L}_{coord}}{dt} &= \frac{i}{\hbar} [\mathcal{H}, \mathbf{r}_{12} \times \mathbf{p}_{12}] \\ &= -\frac{3\gamma}{r_{12}^3} ((\mathbf{I}_1 \times \hat{\mathbf{r}}_{12}) (\mathbf{I}_2 \cdot \hat{\mathbf{r}}_{12}) + (\mathbf{I}_2 \times \hat{\mathbf{r}}_{12}) (\mathbf{I}_1 \cdot \hat{\mathbf{r}}_{12})) = -\frac{d\mathbf{J}_{spin}}{dt}\end{aligned}\quad (7.6)$$

$$\frac{d\mathbf{J}_{tot}}{dt} = \frac{d}{dt} (\mathbf{J}_{spin} + \mathbf{L}_{coord}) = 0\quad (7.7)$$

These arguments can be generalized without complication to our many-spin lattice, for which we conclude that spin angular momentum is *not* conserved in zero field, but rather exchanges continually with the collective orbital angular momentum of the lattice. An interesting consequence of this dipole-mediated coupling of spin space and coordinate space is that an ideal rigid crystal of dipole-coupled spins suspended in free space would be expected to undergo microscopic rotations to offset its changes in internal spin angular momentum.

What relation do these elementary, if possibly intriguing, speculations on angular momentum conservation bear to the fate of spin diffusion in zero field? Diffusion,

at least in the strict sense embodied by the diffusion equation, is in fact founded upon conservation principles. Diffusive transport is always driven by a gradient in some globally conserved quantity. The canonical example is the diffusion of gas along a concentration gradient. There, the diffusion equation may be derived directly from the continuity equation for mass flow through a given region of space. If the constraints of mass conservation (or conservation of total particle number) are lifted, the gas transport is no longer strictly diffusive. Similar arguments can be adduced for spin diffusion. In high field, Zeeman energy and interspin energy are independently conserved. Both therefore diffuse according to a classical diffusion equation. In zero field, however, while interspin energy is still conserved, there is no conserved Zeeman energy, and hence no conserved component of spin angular momentum. Not only can net magnetization in a given direction flow from regions in which it is abundant to other regions in which it is scarce, but it can also disappear into the “sink” of lattice rotation, or can equilibrate among the other magnetization directions.

Microscopically, we may say that magnetization transport in high field relies on a local magnetization gradient: a given spin is driven to “flip” only if there are other opposing spins nearby (cf. the $(1/r_{jk}^3)m_{j\pm}m_{k\mp}$ term in \mathcal{H}).⁴ This local gradient-dependence is what gives rise ultimately to the wavelength dependence of Eq. (6.3) and Figures 7-1 to 7-3. For an initial sinusoidal disturbance, magnetization will be transferred from peak to trough. If the wavelength of the disturbance is short, the peaks and troughs are closely spaced, the internal magnetization gradients are steep, and the equilibration will be fast, whereas if the disturbance wavelength is long, the internal gradients are gentle and the equilibration will proceed at a more leisurely pace. In zero field, by contrast, a spin may be driven to flip without such careful regard for the alignment of its neighbors. In fact, any spin polarization at all constitutes a “gradient” of sorts, and each component of this polarization will decay

⁴In what follows, we appeal to arguments about the microscopic relations of “neighboring” spins. We may use the language of near neighbors with impunity even in the presence of a borderline long-range interaction like the dipole interaction in part because our initial disturbance is applied along only one direction of the three-dimensional lattice, thus the “gradient-weighted” coupling distribution has some degree of one-dimensional character.

locally by equilibration among the three cartesian directions more rapidly than it will spread across the different neighborhoods of the lattice. It is for this reason that in Figure 7-4 we see a rapid decay of the initial magnetization amplitude, essentially independent of the shape of the disturbance.⁵

We may test our convictions about the interconnectedness of conservation and diffusion by applying to our spin lattice a field of moderate strength, intermediate between the high-field and zero-field extremes. In this case, neither Zeeman energy nor dipole-dipole energy is independently conserved. Only their sum, the total magnetic energy, remains constant over time. The transport behavior of each of these three energies – Zeeman, dipole, and total – is summarized in Figures 7-5, 7-6, and 7-7 respectively. In all cases, the untruncated dipole coupling was supplemented by an external field in the z direction of five times the magnitude of the nearest-neighbor dipole coupling.⁶ At short times (shorter than the induction time of roughly one reduced unit), the amplitudes of both Zeeman and dipole energy decay at a rate nearly independent of wavelength, reminiscent of the decay of Zeeman amplitude in Figure 7-4. At longer times, the curves for different wavelengths do disperse, but not enough to define a single diffusion constant. The amplitude of total magnetic energy, on the other hand, adheres closely to a diffusive profile. For reference, the time-dependent amplitudes of Zeeman, dipole, and total energy are juxtaposed with their lattice-averaged values in Figure 7-8. Figure 7-8 corresponds to an initial condition of sinusoidal spin polarization ($A_z(0) > 0$) with no net disturbance in dipole energy ($A_{dip}(0) = 0$).⁷ Zeeman and dipole energy are exchanged in the course of the evolution, while their sum is of necessity preserved. It is interesting to note that the “least diffusive” portions of the amplitude decay curves correspond to the times of

⁵Of course, the equilibration of $x, y,$ and z components of magnetization at any given spot in the lattice is driven by fluctuating fields from the remainder of the lattice, so that we might expect some quasi-diffusive transport effects due to global correlations between magnetization and local field. Nevertheless, in a thermally populated lattice, these effects should be relatively weak compared with direct local effects.

⁶The total local dipole field felt by any spin is generally a few times the magnitude of the nearest-neighbor field, and the applied field was chosen to match this value. The field was oriented parallel to the diffusion axis.

⁷This is also the initial condition that was used to generate the data in Figure 7-5.

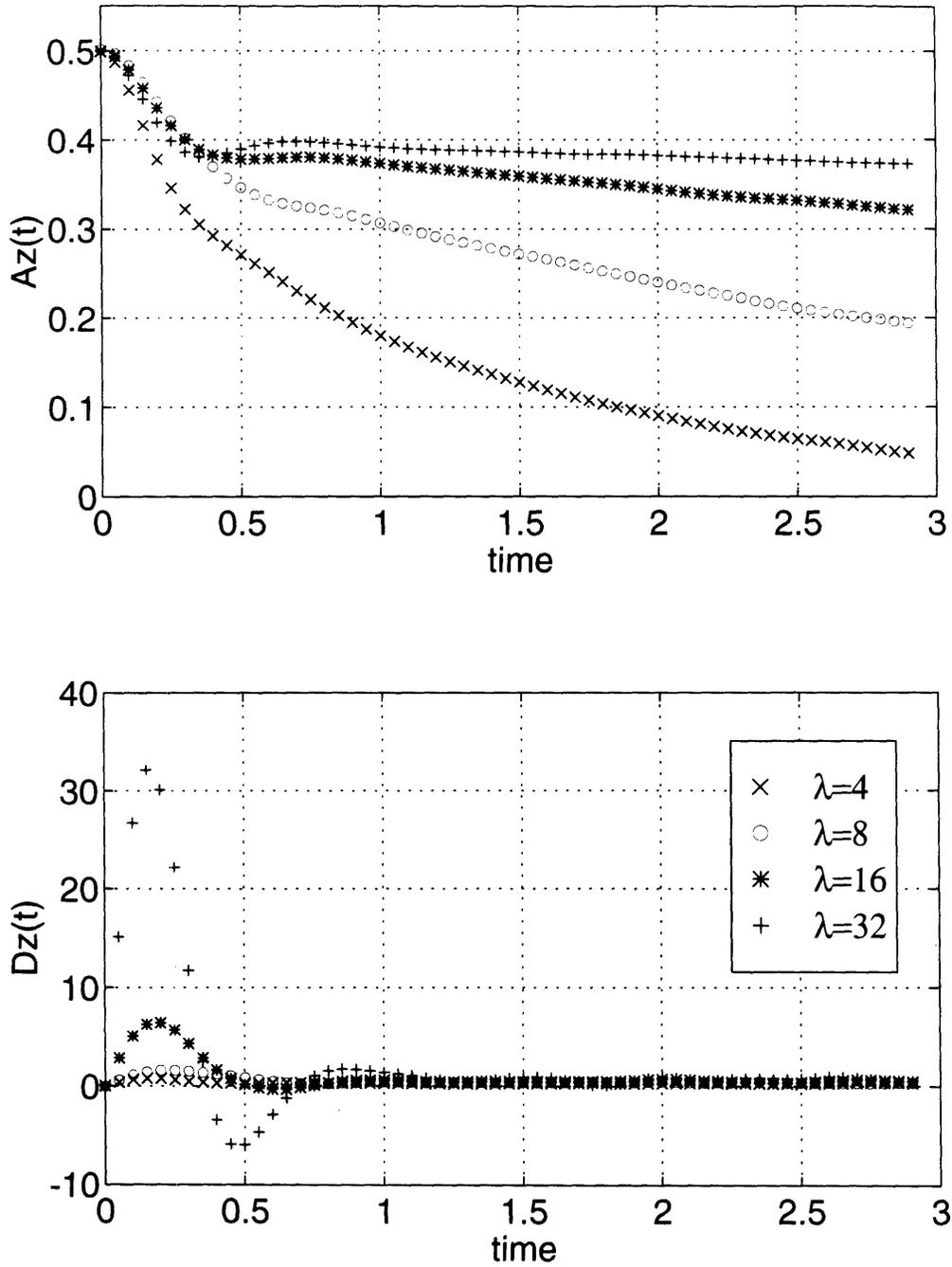


Figure 7-5: Wavelength dependence of disturbance amplitude $Az(t)$ and diffusion constant $Dz(t)$ for Zeeman energy in intermediate applied field, $\omega_0 = 5\omega_d(\mathbf{r}_0)$. Other simulation parameters as in Fig. 7-3.

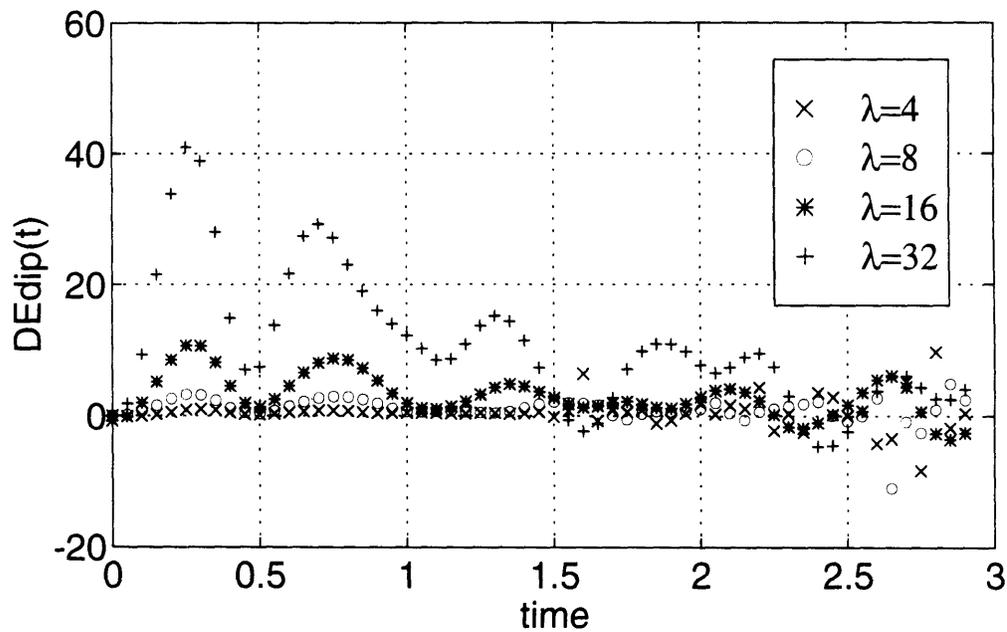
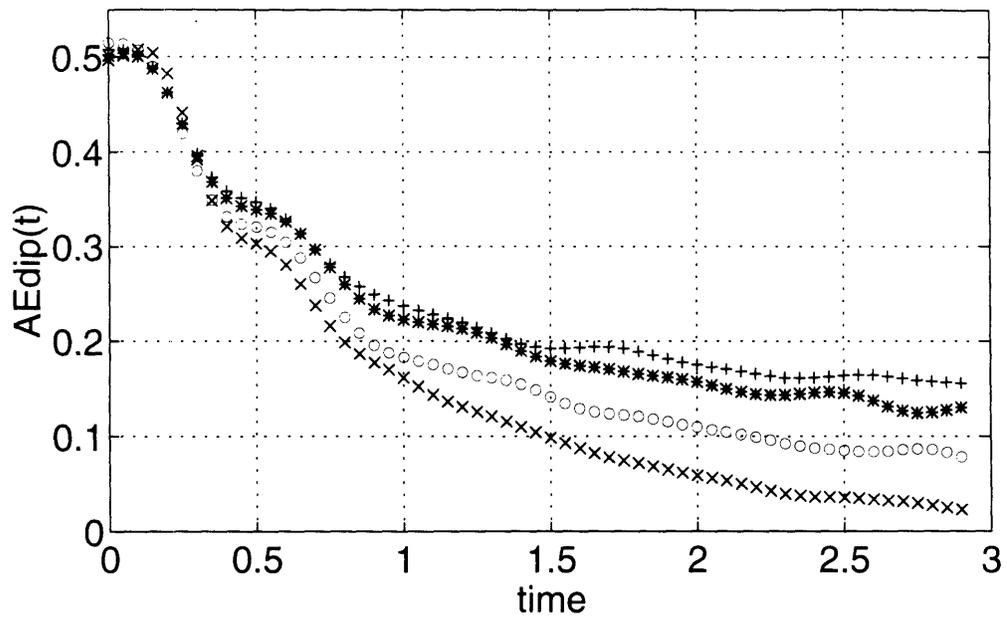


Figure 7-6: Wavelength dependence of disturbance amplitude $A_{Edip}(t)$ and diffusion constant $D_{Edip}(t)$ for dipole energy in intermediate applied field.

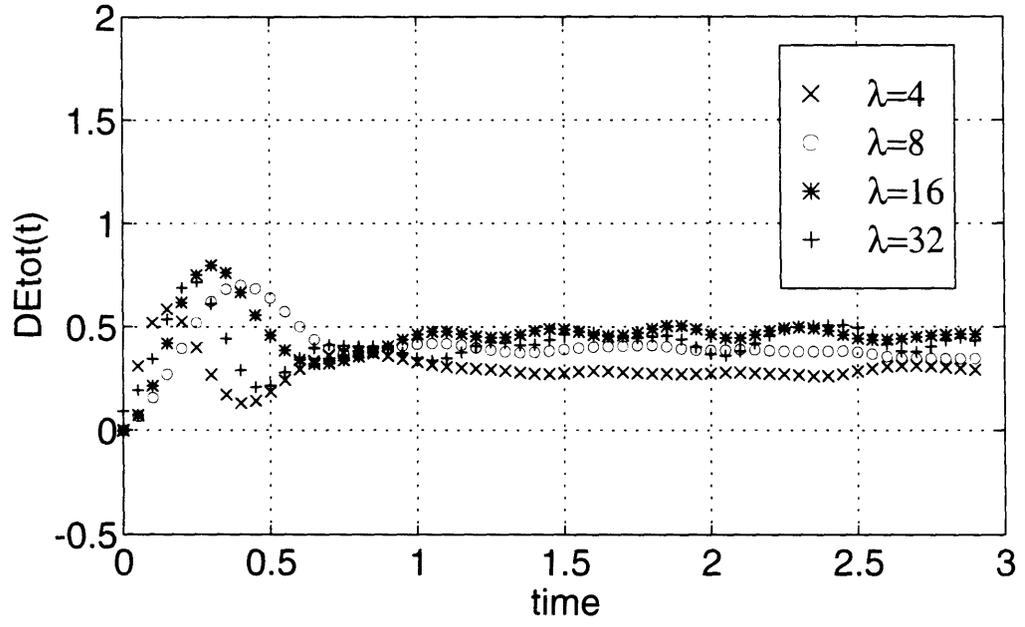
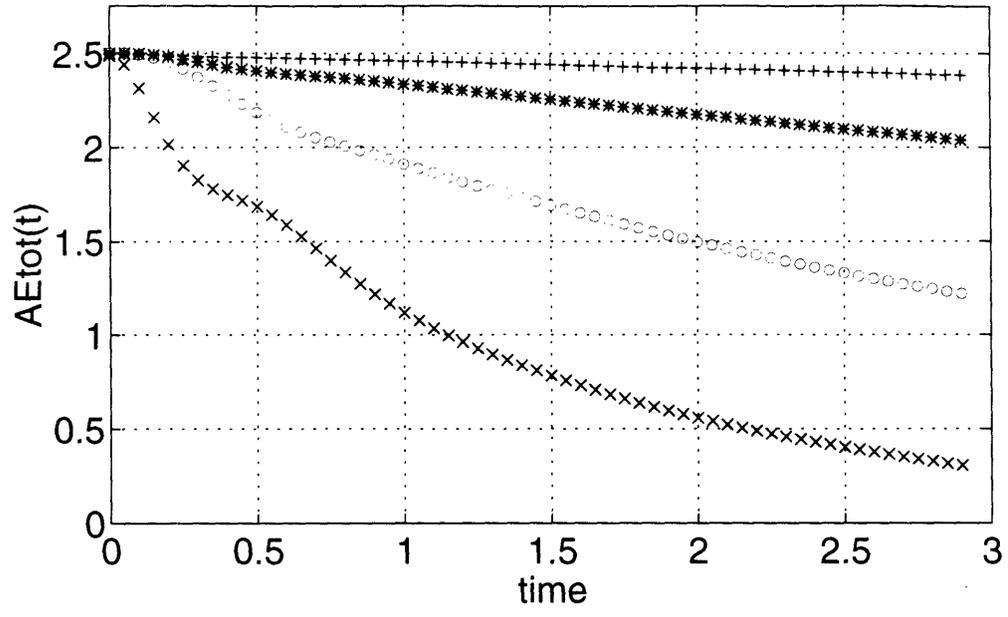


Figure 7-7: Wavelength dependence of disturbance amplitude $A_{E_{tot}}(t)$ and diffusion constant $D_{E_{tot}}(t)$ for total magnetic energy in intermediate applied field.

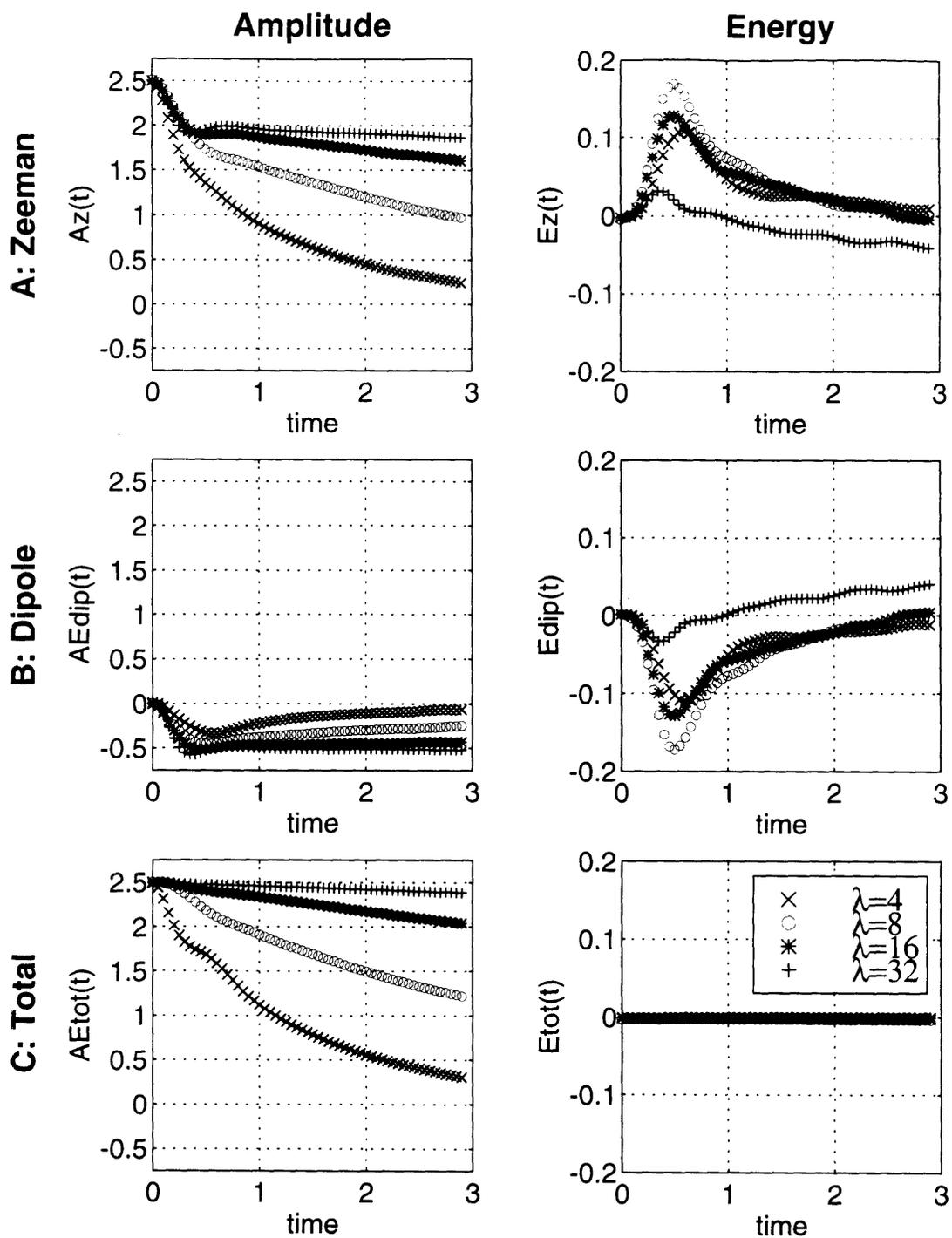


Figure 7-8: Time course of average energies and disturbance amplitudes in intermediate applied field, following establishment of an initial cosinusoidal magnetization profile without any disturbance in dipole energy. A) Zeeman energy. B) Dipole energy. C) Total magnetic energy.

most rapid change in the quantity whose transport is being measured. Apparently, even approximate conservation (i.e. slow variation) of the quantity in question will suffice to ensure approximately diffusive behavior.

Results at other intermediate field values show that diffusion behavior continues to track conservation. At very low fields, it is magnetization which “misbehaves,” while dipole energy, which constitutes the bulk of the total energy, is nearly conserved and diffuses accordingly. As the applied field grows, the Zeeman energy becomes the principal component of the total, and the dipole energy begins to grow non-diffusively relative. Of course at very high fields, truncation of the dipole interaction takes effect, and the Zeeman and dipole energies each behave diffusively. In summary, then, the study of applied-field effects upon spin diffusion is largely a study of the fate of angular momentum and energy at the hands of the local magnetic field.

A few brief comments are in order before we proceed to discuss the effects of concentration in Section 7.2. In many of the Figures presented so far, much of the non-diffusive “action” takes place during the initial induction period. What is this induction time? In the case of genuine diffusion, it is the time required for the lattice to achieve a more realistic starting configuration than the one artificially generated on the computer. For all our algorithmic maneuvering, we do not initialize the lattice with correlations appropriate to the full dipolar interactions among the spins, and an interval of “natural” evolution is necessary to correct our computational biases. In decidedly non-diffusive cases, such as the decay of spin polarization in zero field (cf. Figure 7-4), all visible information from our initial condition disappears within the span of the induction period. This decay is roughly gaussian, as one might expect from the nearly gaussian distribution of local fields in the sample. In any case, the “equilibration” of dipolar correlations occurs within about one reduced time unit, which is on the order of the time required for each spin to rotate once in the dipole field of its neighbors. This time is also commensurate with the dipolar T_2 of the sample. Thus, we may think of the induction time, like the characteristic dephasing time of an FID, as the time required to establish new correlations or to destroy old ones.

Zero Field (this work)	High Field (Ref. [40])		
D_E	$D_{E\parallel}$	$D_{E\perp}$	\bar{D}_E
1.00	0.63	0.19	0.34

Table 7.1: Diffusion constants for interspin energy in zero field and in high field. In zero field, the diffusion is essentially isotropic. The high field diffusion constants are taken from Ref. [40], where they were calculated for diffusion parallel ($D_{E\parallel}$) and perpendicular ($D_{E\perp}$) to an applied field along the [100] lattice direction. The average diffusion constant $\bar{D}_E = \frac{2}{3}D_{E\perp} + \frac{1}{3}D_{E\parallel}$ is also included.

To complement the qualitative insights gleaned in this Section, we conclude with a quantitative comparison of spin diffusion in high and low applied field. In Table 7.1 may be found the calculated diffusion constant for interspin energy in zero field, as well as the corresponding high-field values determined by Tang and Waugh. (We use interspin energy for our comparison because, once again, the magnetization does not diffuse properly in the absence of an applied field.) The larger value of D in zero field reflects the presence of extra active terms in the untruncated dipolar Hamiltonian.

7.2 Concentration Dependence of Spin Diffusion and the Effects of Lattice Geometry

In discussing the conservative origins of diffusive spin dynamics, we have emphasized the role of microscopic energy- or magnetization-gradients and have addressed the global consequences of local relations between a spin and its neighbors. Changes in the functional form of the interaction between neighboring spins (effected by changes in the strength of the applied magnetic field) can have profound effects upon the bulk transport behavior of the lattice. What happens to spin diffusion when the form of the interaction is preserved but the distribution of neighbors changes? This has been the subject of a second set of simulations in which the population and/or the geometry of the lattice were varied.

Tang and Waugh, in their original spin diffusion simulations, studied the effects of restricting the interaction region around each spin – essentially providing an arbitrary cutoff to the range of the dipole interaction. The resulting changes in diffusion con-

stant were traced to changes in interspin energy, which itself depends upon the size of the interaction region. Still, the total interspin energy is rather a coarse indicator for our dynamical expectations: there are many ways to perturb the interspin energy of a spin lattice, and one might imagine that different microscopic configurations could have distinct dynamical consequences. What would happen if some degree of disorder were introduced into the lattice – if, for example, some random subset of lattice sites were left unpopulated, or were populated by non-magnetic impurities?

It is not difficult to modify the basic spin-diffusion simulation to study the effects of random lattice dilution. One first chooses a target lattice concentration c_{target} between 0 and 1, then one samples a random number generator for each lattice site. If the generator returns a value less than c_{target} , the site is populated with a spin. Otherwise, it is left empty. The quantity c_{target} therefore serves as a site occupation probability, and the actual fractional concentration $c = N_{spins}/N_{sites}$ approaches c_{target} as the number of spins becomes large. The convolution method of Section 6.3 may still be used to compute local fields as long as the matrix of magnetizations is padded with zeros at the vacant sites. (Only the occupied sites are updated at each time point, so that errors do not accumulate at the vacancies.) The concentration dependence of the diffusion constant for spin magnetization in high applied field is plotted in Figure 7-9. A three-dimensional simple cubic lattice ($N_{sites} = 16^3$, [111] field orientation) was used for this calculation.⁸ The diffusion constant is seen to fall monotonically as the lattice is diluted. This behavior is not surprising, since the magnitude of the average local field experienced by any of the spins falls off similarly as spins are removed from the lattice.

Can we understand the shape of the $D(c)$ curve more quantitatively? The moment theory of Redfield and Yu [26] serves nicely for this purpose, since moments can be expressed as lattice sums whose concentration dependence is easy to ascer-

⁸Irmgard Nolden [42] was the first to perform dilution studies in three-dimensional lattices using modifications of the core code created by Changguo Tang. The results reported here were generated independently, but Nolden's data served as helpful guides in choosing starting parameters such as lattice size and disturbance wavelength. In particular, information from [42] suggests that for $\lambda = 16$, the diffusion constant vs. concentration curve is independent of lattice size beyond $N = 16 \times 16 \times 16$ for lattices with a [111] field orientation.

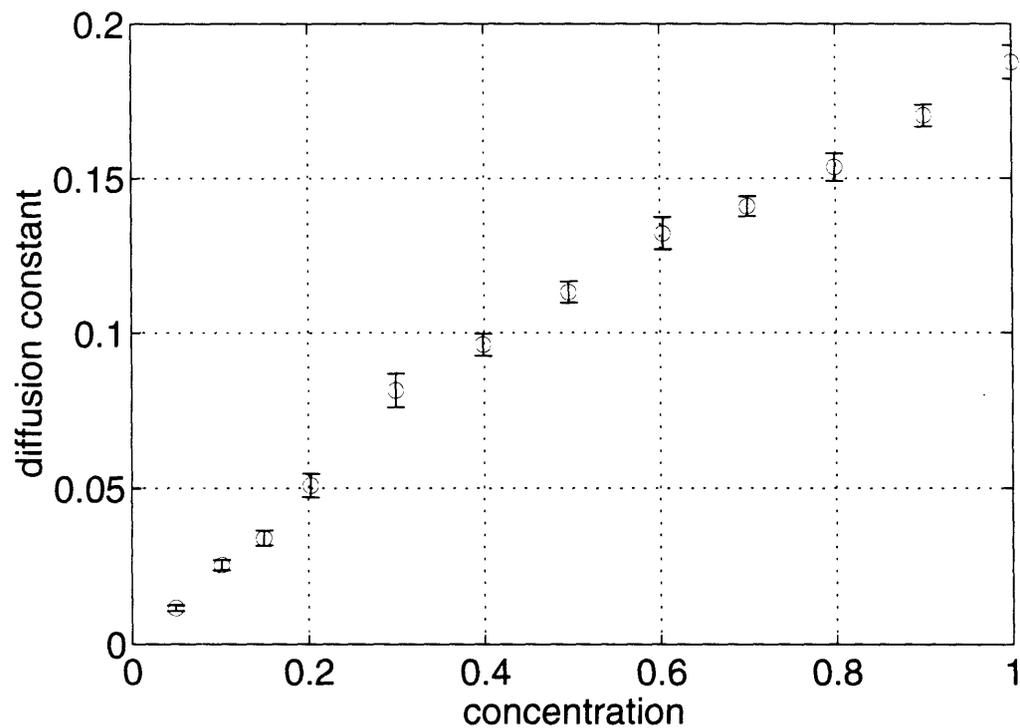


Figure 7-9: Concentration dependence of the spin diffusion constant (high-field diffusion of magnetization) in a three-dimensional simple-cubic lattice. The size of the lattice is $N_{sites} = 16 \times 16 \times 16$, and the orientation of the applied field is $[111]$. Diffusion was measured along the $[100]$ axis for an initial spin polarization with $\lambda = 16$. At each concentration, diffusion constants for 8 different runs were obtained as averages of the instantaneous values $D(t)$ over suitable time intervals, and these 8 average values were averaged to yield the value shown in the plot. The error bars indicate standard error of the mean of the 8 samples.

tain. Redfield and Yu derive the following form for the diffusion constant D_Z of spin magnetization (or Zeeman energy) in high field:

$$D_Z = \left(\frac{M_2}{k^2} \right) \left(\frac{\pi M_2}{2M_4} \right)^{\frac{1}{2}} \quad (7.8)$$

As usual, k is the wavenumber of the initial magnetization disturbance. M_2 and M_4 are the second and fourth moments of an appropriately defined absorption lineshape (cf. [26]), and are related (though not identical) to the classic Van Vleck moments [43]. Some algebraic manipulation, reproduced in Appendix A, yields the expression

$$D_Z = \left(\frac{-\frac{\pi}{2} \left(\sum_j' x_{ij}^2 \mathcal{A}_{ij} \right)^3}{\sum_j' \sum_k' x_{ij}^2 \mathcal{B}_{ijk} + \sum_j' x_{ij}^2 \mathcal{C}_{ij}} \right)^{\frac{1}{2}} \quad ijk \neq \quad (7.9)$$

where the primed sums run over the occupied lattice sites. $x_{ij}^2 = (x_i - x_j)^2$ are the squared separations of spins i and j in the direction of the initial wavevector, and

$$\begin{aligned} \mathcal{A}_{ij} &= \frac{1}{3} \left(\frac{3 \cos^2 \theta_{ij} - 1}{2r_{ij}^3} \right)^2 = \frac{1}{3} b(\mathbf{r}_{ij})^2 \equiv \frac{1}{3} b_{ij}^2 \\ \mathcal{B}_{ijk} &= \frac{2}{9} \left\{ 8b_{ij}^2 (b_{ik} - b_{jk})^2 + 4b_{ij} b_{ik} b_{jk} (b_{ik} + b_{jk}) - 3b_{ik}^2 b_{jk}^2 \right\} \\ \mathcal{C}_{ij} &= -\frac{32}{15} b_{ij}^4 \end{aligned} \quad (7.10)$$

We may now argue, following Kittel and Abrahams [44], that for a randomly diluted crystal with site occupation probability c , each spin on average will see other spins at only a fraction c of the lattice sites, and each primed sum in Eq. (7.9) will be smaller than the corresponding sum over all sites by a factor of c . That is,

$$\begin{aligned} \sum_j' &\rightarrow c \sum_j \\ \sum_j' \sum_k' &\rightarrow c^2 \sum_j \sum_k \end{aligned} \quad (7.11)$$

Thus, we may write D_Z in terms of c and the values of the (unprimed) lattice sums

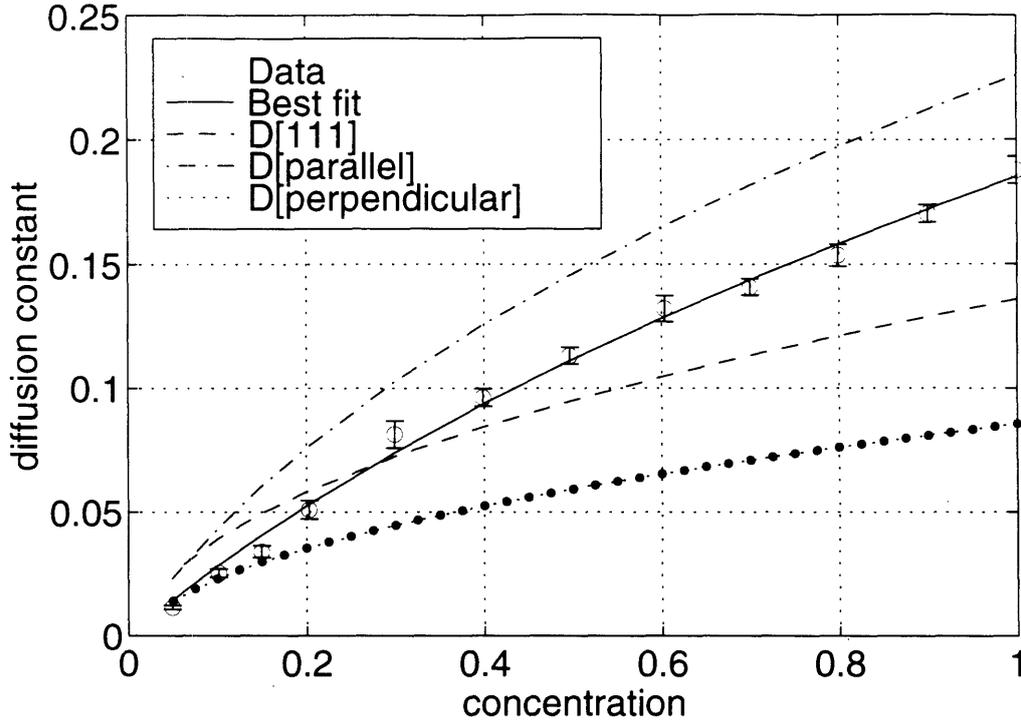


Figure 7-10: Fit of $D(c)$ from Fig. 7-9 to the Redfield and Yu moment theory [26]. The diffusion-constant data was generated with a field along the diagonal [111] direction and diffusion along [100]. Also included are moment predictions for diffusion parallel and perpendicular to an applied field along [100].

at full lattice population. The result is

$$D_Z = \frac{\chi_1 c}{(1 + \chi_2 c)^{\frac{1}{2}}} \quad (7.12)$$

with

$$\chi_1 = \left(\frac{\frac{\pi}{2} \left(\sum_j x_{ij}^2 \mathcal{A}_{ij} \right)^3}{-\sum_j x_{ij}^2 \mathcal{C}_{ij}} \right)^{\frac{1}{2}}$$

$$\chi_2 = \frac{\sum_j \sum_k x_{ij}^2 \mathcal{B}_{ijk}}{\sum_j x_{ij}^2 \mathcal{C}_{ij}} \quad ijk \neq \quad (7.13)$$

Figure 7-10 compares the data in Figure 7-9 with the predictions of the Redfield and Yu theory. A nonlinear least-squares fit of the data to the functional form of

	Best Fit to Data	Moment Calculations			
	[111]	[111]		⊥	
χ_1	0.30	0.89	0.51	0.41	This work
χ_2	1.69	42.02	4.19	21.74	
$D(c = 1)$	0.19	0.14	0.23	0.09	
$D(c = 1)$	0.22		0.35	0.14	Ref. [41]

Table 7.2: Fitting parameters χ_1 , χ_2 and diffusion constant values $D(c = 1)$ for the moment theory fits shown in Fig. 7-10. The best-fit values were obtained by minimizing rms deviation of the data from the predicted functional form of Eq. (7.12). Moment calculations derive from Eq.'s (7.12) and (7.13). The bottom row contains simulation data for various field orientations from Ref. [41].

Eq. (7.12) yields admirable agreement. The best-fit parameters χ_1 and χ_2 do not match the predicted values from Eq. (7.13), however. For a [111] field orientation, moment calculations predict a smaller diffusion constant at full concentration and a smaller initial slope than are observed in the simulations. Numerical values of χ_1 , χ_2 , and $D(c = 1)$ appear in Table 7.2, and simulation results for $D(c = 1)$ from Ref. [41] are included for comparison. One can see that the moment calculations do reproduce many of the qualitative features of the simulations, such as the expected anisotropy with respect to field orientation, though the calculated diffusion constants fall consistently below the simulated values.⁹

What accounts for the residual discrepancy between the simulated and the predicted shapes of the concentration dependence? First of all, the Redfield and Yu moment theory is by admission an approximation. There is latitude, for example, in the choice of lineshape leading to an expression like Eq. (7.8). Second, the finite lattice size used in our calculations may be a limitation: the moments of Eq. (7.8) are evaluated in the long-wavelength limit, while a lattice size of $N = 16^3$ and a wavelength of $\lambda = 16$ may fall slightly short of this limit. The cyclic boundary conditions of our model lattice may also exert an untoward effect at small enough N . Nevertheless, whether or not these factors are sufficient to explain the bulk of

⁹The value of $D(c = 1)$ from the best-fit curve of Figure 7-10 is also smaller than the corresponding value from Ref. [41]. This is because a wavelength of $\lambda = 16$ was used here, rather than $\lambda = 64$ as in Ref. [41].

the numerical discrepancy, there is on closer inspection something qualitatively unsettling about the Redfield and Yu theory. In the reasoning leading to Eq. (7.12), no reference can be found to the *dimension* of the lattice. The summation arguments of Eq. (7.11) apply equally well whether the spins are arranged in a line, a plane, or a cube. We know, however, that the dipole interaction, which is borderline long-range in three dimensions, is not so in one or two dimensions. As the effective range of the interaction is decreased, disciples of percolation theory might begin to object that diffusion should no longer scale uniformly with dilution, but should instead show signs of a percolation threshold. While the “low-resolution” statistical standpoint of moments may have merit in many situations, we may ultimately have to adjust our notions about the interplay of local and global influences in order to understand spin diffusion at low concentrations or in fewer than three dimensions.

Simulations in one, two, and three dimensions yield the results displayed in Figure 7-11. There is a marked difference in the shape of the $D(c)$ curves for the different dimensionalities. For the sake of direct comparison, similar curves normalized to unity at $c = 1$ are plotted in Figure 7-12. For all dimensions, one may observe that $\log(D)$ is nearly proportional to $\log(c)$ for an appreciable range of concentrations near $c = 1$, after which the curves seem to “turn around.” While the confidence levels are not very high for values of $\log(D)$ at low concentrations (the fractional uncertainty in D is large as D approaches zero, and the simulations are hampered by statistical noise), one may almost begin to discern a set of threshold regions in which the concentration dependence seems to change character.

Dimension-dependence may be introduced into the spin diffusion problem in a preliminary (and highly approximate) fashion by way of a simple scaling argument. If the lattice population is diluted by a factor c , then the average effective volume occupied by any spin increases by a factor of c^{-1} , and the average nearest-neighbor distance r_0 increases by a factor of $c^{-1/d}$, where d is the dimension of the lattice. If we imagine that this *average* effect on interspin distances constitutes the primary effect of the dilution, then we may model the diluted lattice as a fully populated lattice with the same geometry as the original lattice but with an effective lattice

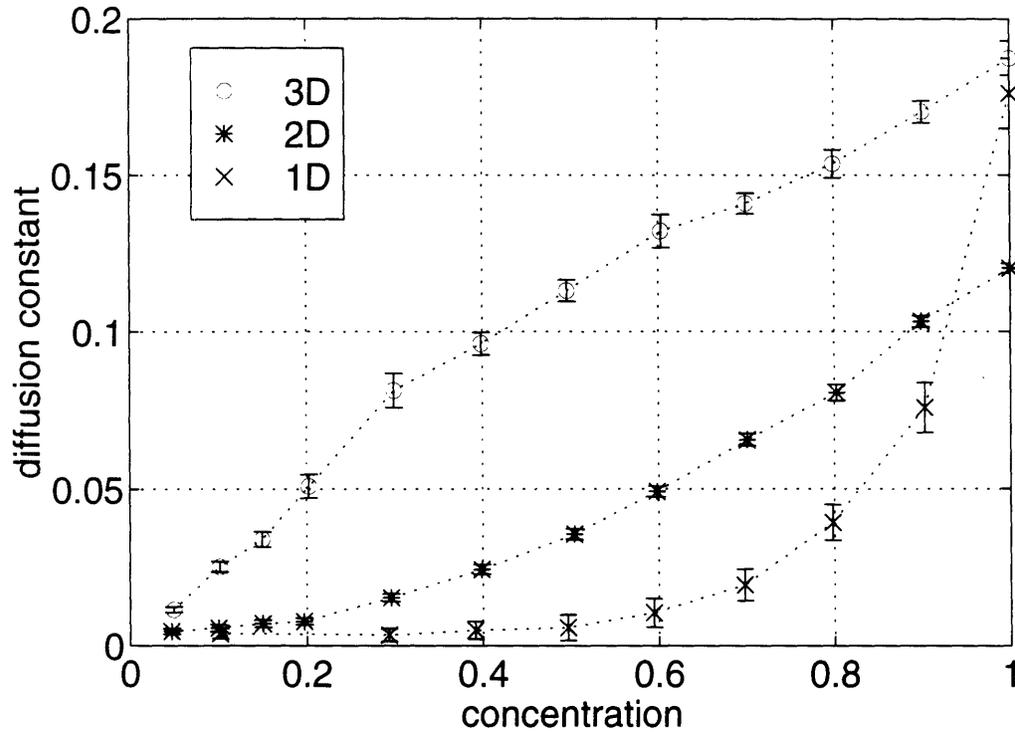


Figure 7-11: Concentration dependence of the spin diffusion constant in one, two, and three dimensions. 3D: 8 runs, $N_{sites} = 16 \times 16 \times 16$, $\lambda = 16$, field orientation [111], diffusion direction [100]. 2D: 16 runs, $N_{sites} = 64 \times 64$, $\lambda = 16$, field orientation [111], diffusion direction [100]. 1D: 32 runs, $N_{sites} = 256$, $\lambda = 16$, field orientation and diffusion direction [100]. The 1D diffusion constant is greater than its 2D counterpart at $c = 1$ because of the field orientation chosen. A [111] field orientation could not be used in the 1D case, since all 1D couplings vanish for that orientation.

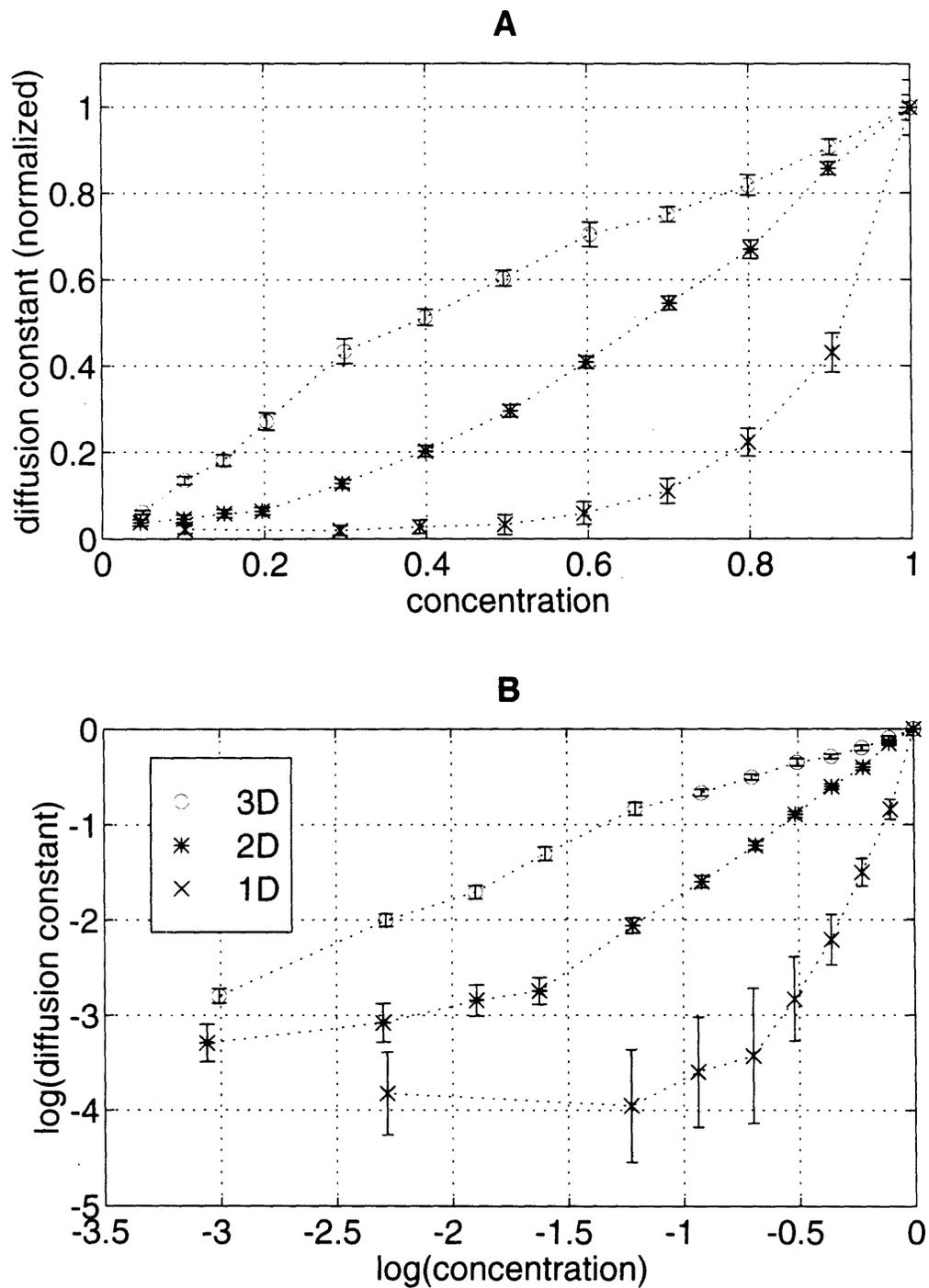


Figure 7-12: Concentration dependence of 1D, 2D, and 3D spin diffusion constants normalized to unity at $c = 1$. A) linear scale. B) log scale.

spacing of $c^{-1/d}r_0$. The base coupling strength $b(r_0) \propto r_0^{-3}$ is then scaled by $c^{3/d}$, and all characteristic times, including the decay rate $\tau_k = (k^2 D)^{-1}$, grow by the inverse factor $c^{-3/d}$. The diffusion constant D , therefore, scales with concentration as $c^{3/d}$. Figure 7-13 compares the observed $D(c)$ curves with the predictions of the uniform scaling model on the one hand and the Redfield and Yu moment theory on the other. While the moment arguments, as we have already seen, predict some of the essential features of the three-dimensional curve, scaling arguments are much more successful in one and two dimensions. Even then, however, the fit is not perfect: especially in one dimension, the simulated $D(c)$ falls off significantly faster than c^3 , and, of course, in no instance does the scaling model predict anything like a percolation threshold.

The theory of percolation [45], as applied to diffusion, addresses not just the average scaling behavior but also the distribution of interconnected clusters as a diffusing network is randomly diluted. If the dilution introduces a sufficient number of gaps such that no diffusion path connects the entire lattice, then diffusion is effectively restricted to smaller sublattices, and certain characteristic changes are expected in the behavior of the diffusion constant. The concentration at which, on average, the last fully cosmopolitan cluster disappears is known as the percolation threshold. For our diffusing spin system in particular, we expect a percolation threshold to be marked by two kinds of effects. First, for moderate dilutions, a falloff in the observed diffusion constant at long times should accompany the onset of restricted diffusion. Initially, the spins in a given region do not “know” about their membership in a restricted cluster, and diffusion proceeds at its usual rate, but after a while, the magnetization front runs up against the cluster boundaries and further transport is hindered. Some behavior of this sort is observed in the simulated diffusion constant data, though the presence of an initial induction time and of statistical fluctuations in $D(t)$ complicate these observations. Second, when cluster sizes are small enough (with an average length scale significantly smaller than the wavelength of the magnetization disturbance, for example), spin diffusion will be quenched. In this case, the restricted clusters sit on a small portion of the sinusoidal disturbance profile, and there is no significant local gradient to drive magnetization transport.

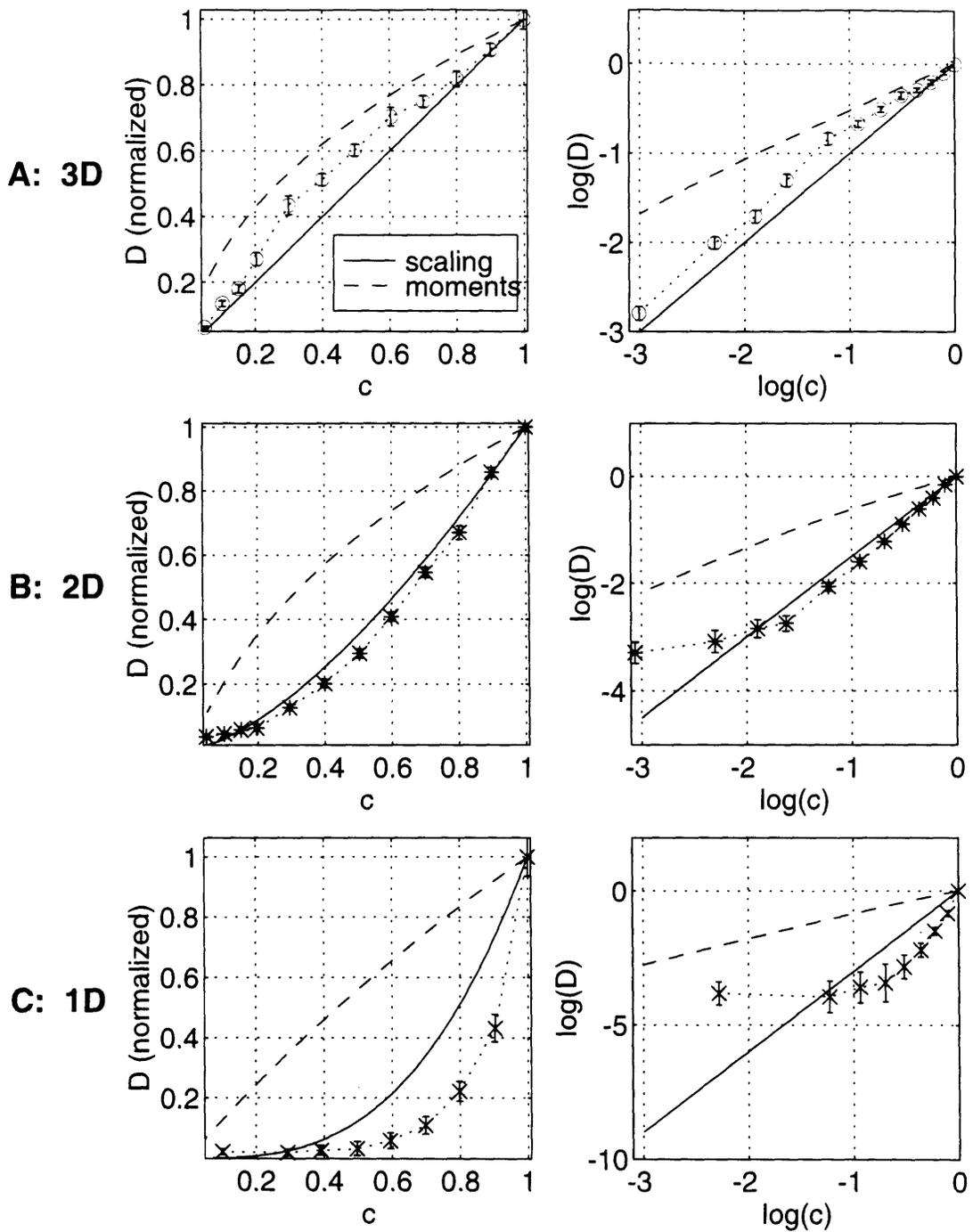


Figure 7-13: Comparison of simulated $D(c)$ curves with the predictions of moment theory and of uniform scaling arguments described in the text. A) Three dimensional lattice. B) Two-dimensional lattice. C) One-dimensional lattice. The left-hand column contains linear plots, the right-hand column the corresponding log-log plots.

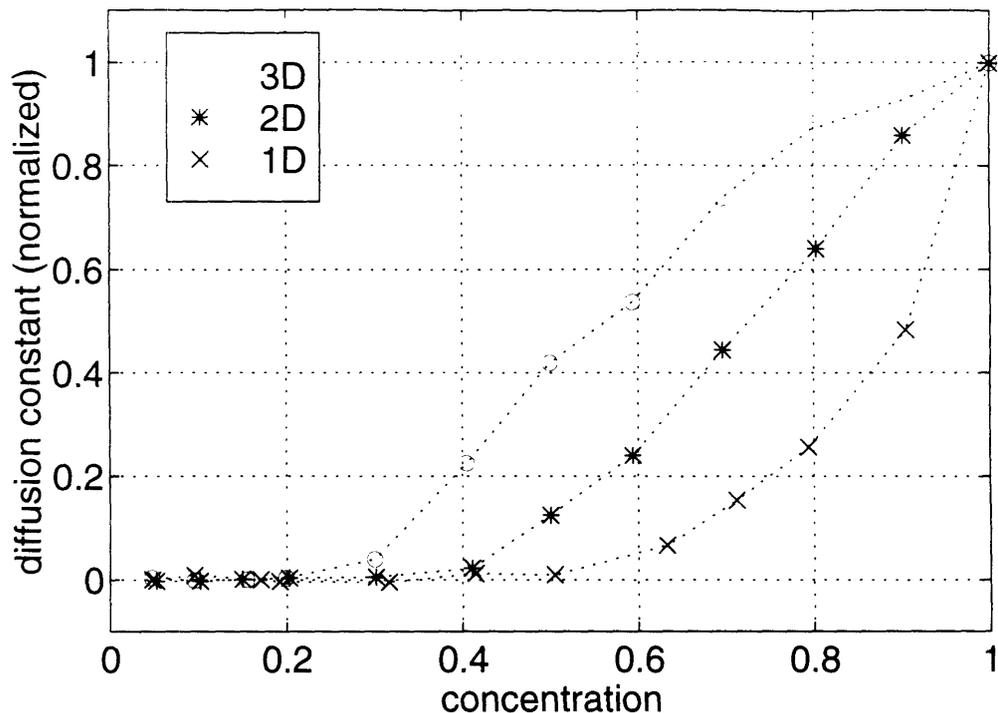


Figure 7-14: Concentration dependence of high-field spin diffusion for a nearest-neighbor Heisenberg exchange coupling.

Terms such as “cluster size” and “cluster boundary” apply only in a relative sense to a system such as ours with interactions of extended range. Even in a one-dimensional array of spins, magnetization can cross gaps, incurring only the penalty of a time delay. Furthermore, spin diffusion involves continuous rotations rather than discrete jumps, so that diffusion paths are not easy to define, not to mention to enumerate. Even if such paths could be tallied conveniently, they would have to be weighted by the particular magnetization gradients available to drive diffusion. Thus, a full quantitative theory of the percolation properties of our spin system promises to be exceedingly difficult.¹⁰ If we wish to determine with confidence whether there is such a thing as a percolation threshold for spin diffusion, we are forced to make simplifications.

Figure 7-14 shows results for one simplified spin system. In place of the dipole-

¹⁰Drastically simpler problems, involving random hops between adjacent sites only, are difficult enough in two or more dimensions. Cf. Ref. [45].

dipole coupling, a Heisenberg exchange coupling was used, with interactions occurring only between nearest neighbors.¹¹ Quenching of spin diffusion at low concentrations is now plain to see in one, two and three dimensions. When we compare the Heisenberg coupling results to the dipolar results, in Figure 7-15, we find a degree of discrepancy which varies with lattice dimension. In two and three dimensions, the $D(c)$ curve for nearest-neighbor Heisenberg coupling travels with its dipolar counterpart at high concentrations, then descends more steeply toward a distinct threshold. As expected, the quenching of diffusion occurs at higher concentrations in two than in three dimensions, and the divergence between Heisenberg and dipolar behavior is more muted in the planar lattice. The one-dimensional curves are barely distinguishable, suggesting that nearest-neighbor influences are nearly sufficient to explain the effects of dilution in a line of spins. In short, the simplified Heisenberg model reproduces many of the qualitative features of our full dipolar spin system, while emphasizing the role of, as it were, local demographics in the overall population dynamics of the lattice. The concentration-dependence of spin diffusion may then be understood as a kind of “smoothing over” of the percolative behavior of truly local interactions.

One further numerical experiment serves to highlight the effects of local and global lattice geometry upon spin diffusion. Figure 7-16 plots the change in diffusion constant as the unit cell of a three-dimensional lattice is stretched in a direction perpendicular to the diffusion axis. (The length of the unit cell is increased along the z direction, for an initial disturbance of magnetization in the x direction ($\hat{\mathbf{k}} = \hat{\mathbf{x}}$).) From Figure 7-9, one can see that the diffusion constant at $c = 1$ is roughly 0.19 for a three-dimensional lattice, and 0.12 for a two-dimensional lattice (with a field orientation of [111] in both cases). As the three-dimensional lattice of Figure 7-16 is stretched, the diffusion constant makes a smooth transition from its three-dimensional to its two-dimensional value. When the unit cell becomes so elongated that the coupling between lengthwise neighbors is many times smaller than that between spins

¹¹Direct summation rather than convolution was used for local field calculations in this case, since for a nearest-neighbor interaction, relatively few sites contribute to each local field, making summation the more efficient computation.

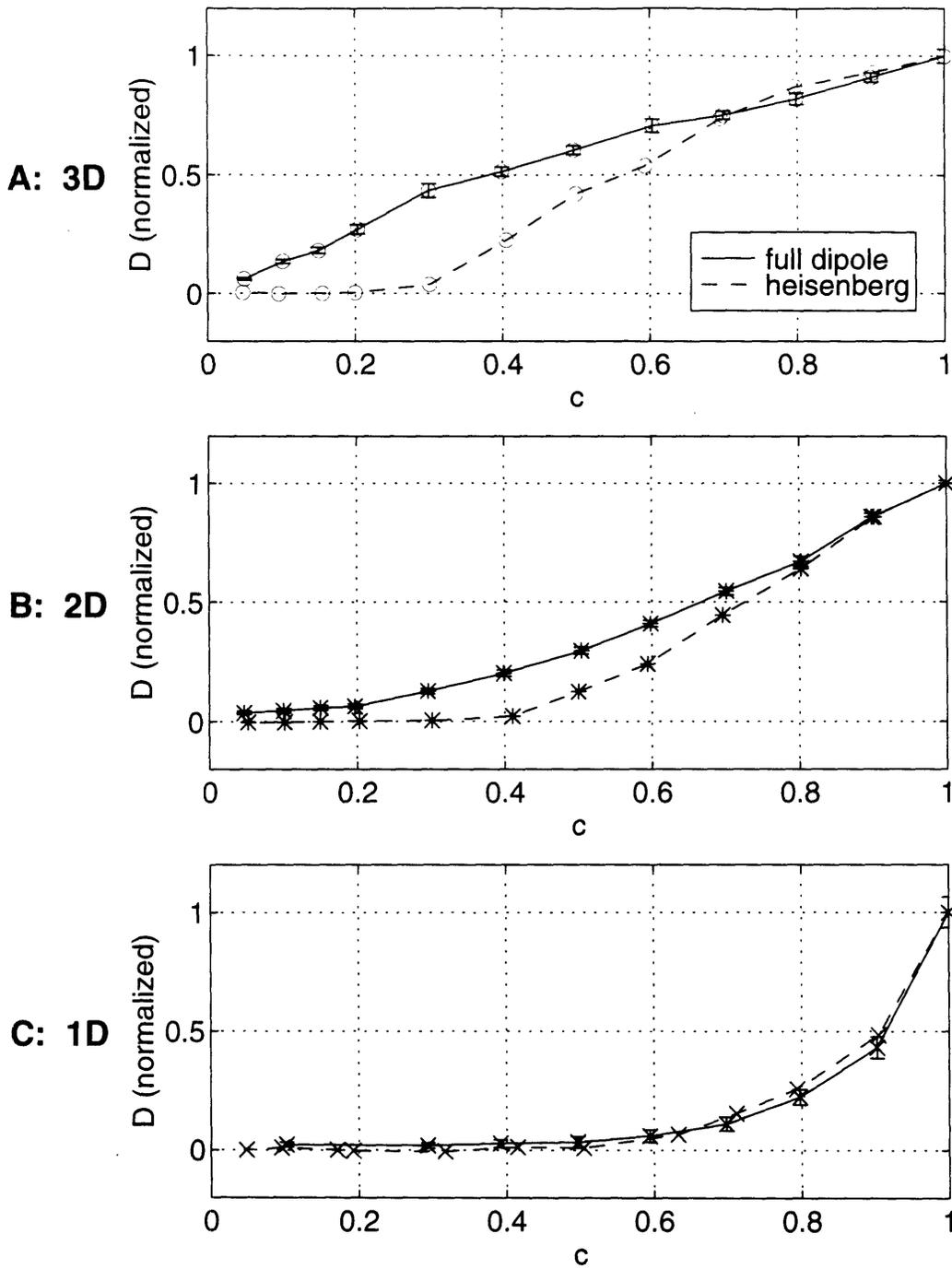


Figure 7-15: Comparison of $D(c)$ for dipole couplings and Heisenberg exchange couplings. A) Three-dimensional lattice. B) Two-dimensional lattice. C) One-dimensional lattice.

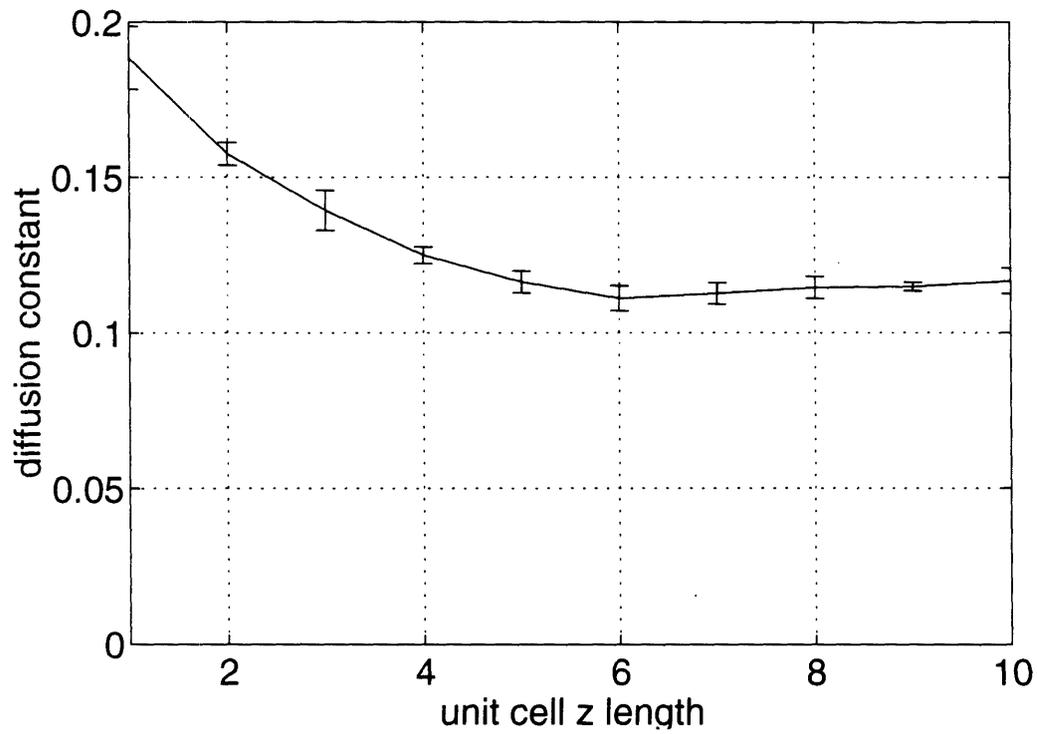


Figure 7-16: Spin diffusion constant as a function of unit cell elongation. A three dimensional lattice was used, with $N = 16 \times 16 \times 16$, $\lambda = 16$, field orientation $[111]$, and diffusion direction $[100]$, as in Fig.'s 7-9 and 7-11. The unit cell length was increased in the $[001]$ direction.

in the undisturbed transverse planes, then diffusion occurs only within those planes, and different planes serve merely as independent replicas of a two-dimensional template lattice. In the language of percolation theory, the transverse planes constitute separate clusters, with diffusion occurring within clusters but communications among clusters being effectively blocked.

Spin diffusion simulations may be repeated ad infinitum for a panoply of specific lattice geometries, but we expect the general principles from the experiments described so far to apply in most new cases. For some geometries, there will be one or several preferred directions of spin diffusion. For others, like the simple cubic geometry, the anisotropy will be determined primarily by how we choose to orient the lattice and its spins.¹² In all cases, however, the long-range character of the dipole coupling is unlikely to obscure entirely the microscopic origins of its actions in driving spin diffusion. As experimental techniques for measuring spin diffusion improve, the dilution studies presented in this Section may find some application in the study of lattices populated by spins with low natural abundance. Perhaps, in this age of microfabrication, spin systems will also be found in which restricted spin diffusion in a plane or on a line can be detected and manipulated. In the meantime, these observations on classical spin diffusion have been offered as a window onto the groundwork of a many-spin effect. If fabrication technology advances so far as to allow the deposition of spins in more than three dimensions, further investigations will be in order. For a dipole-coupled four-dimensional lattice, all bets are off.

¹²It is worth noting that there are several sources of anisotropy in the spin diffusion problem. In high applied field, the field orientation establishes one preferred direction and axis of symmetry, while the lattice has its own independent set of symmetries. The anisotropy of the diffusion constant will depend upon how these two sets of symmetries interact. Even in zero-field, diffusion may be sensitive to the geometry of the initial disturbance, though we expect this sensitivity to be mild in the limit of long wavelengths and high temperatures.

Chapter 8

Quantum Spin Dynamics and Spin Coherent States

To this point, we have concerned ourselves entirely with classical spin diffusion. Our model has been one of classical gyromagnets precessing continuously under the influence of local magnetic fields. Nevertheless, the dynamics of interacting nuclear spins are, at root, quantum mechanical. What account can be made of the quantum origins of spin diffusion? All of the theoretical models of spin diffusion against which classical simulations were tested are quantum models. None of these models, however, affords a direct microscopic comparison with the computational model presented so far. None is amenable to simulation at the level of individual spins. While the values of the spin diffusion constant derived from quantum theories and classical simulations do agree in practice, and while one can make a case for the accuracy of the classical results, it would be instructive to understand the precise nature of the quantum corrections to these results. Precisely when do quantum deviations from classical behavior become important? This Chapter lays out a theoretical framework for exploring such a question. A full quantum simulation of large numbers of dipole-coupled spins will remain for the time being out of reach; but, by re-casting the quantum problem in terms of “quasi-classical” quantum states known as spin coherent states, we may lend new credence to the classical simulations, and we may investigate in close detail the connections between quantum and classical spin diffusion.

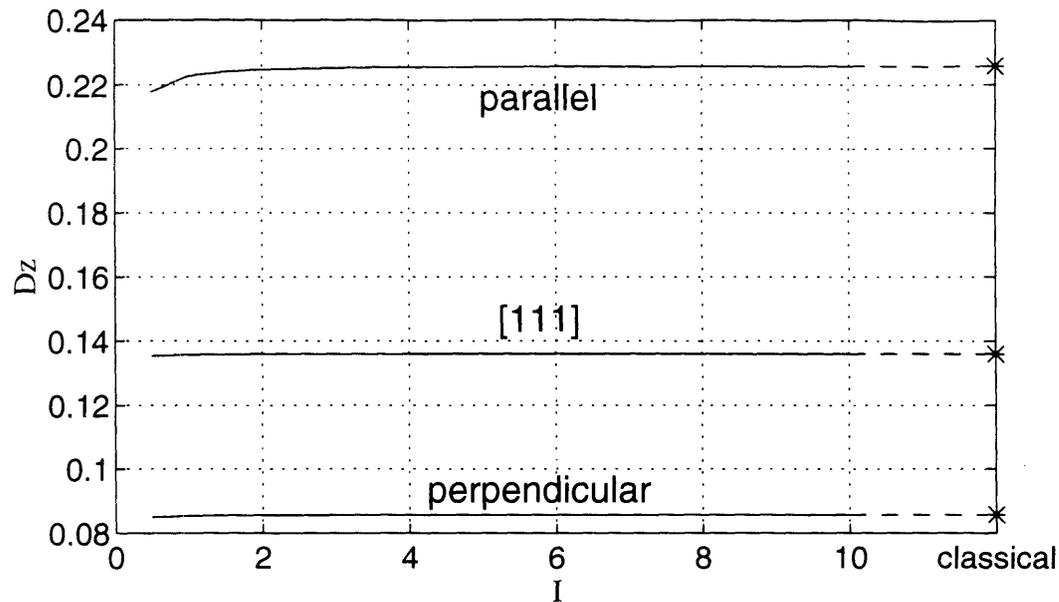


Figure 8-1: Spin diffusion constant D_z predicted by the Redfield and Yu moment theory, as a function of spin quantum number I . Curves are shown for diffusion parallel and perpendicular to a field along [100], and for diffusion along [100] with a field in the [111] direction.

8.1 Comparison of Classical and Quantum Spin Diffusion Using Moment Theory

If we wish simply to compare the classical and quantum values of the spin diffusion constant, disregarding the precise details of the dynamics, we may again make use of the moment theory of Redfield and Yu (Ref. [26] and Appendix A). In Ref. [41], Tang and Waugh argue indirectly for the validity of the classical results by comparing the exact Van Vleck moments with their classical limits. The Redfield and Yu theory provides a direct link between the spin diffusion constant and a related set of moments, evaluated in the limit of long wavelength and high temperature. In Figure 8-1, the predicted value of D_z is plotted as a function of spin quantum number I . The quantum result rapidly approaches its classical counterpart, and even for spin 1/2, the quantum correction is only on the order of 0.5% to 4% of the total value. Moment arguments predict unequivocally, if somewhat abstractly, that the classical results should be faithful guides for our more quantum mechanical expectations.

8.2 The Quantum Dynamical Problem

For a study of quantum spin dynamics, the individual dipoles $\{\mathbf{m}_j\}$ of Chapter 6 must be replaced by spin operators $\{\mathbf{I}_j\}$ with the usual commutation relations

$$[I_{j\alpha}, I_{k\beta}] = i\delta_{jk}\epsilon_{\alpha\beta\gamma}I_{j\gamma}, \quad i, j = 1, 2, \dots, N \quad \alpha, \beta, \gamma = x, y, z \quad (8.1)$$

Henceforward, we shall let $\hbar = 1$ and $\gamma = 1$, unless the context demands otherwise. The spin quantum number, which we shall take to be the same for all spins in the lattice, will be referred to simply as I . The Hamiltonian for our spin system is now

$$\mathcal{H} = -\sum_j \mathbf{I}_j \cdot \mathbf{B}_0 - \frac{1}{2} \sum_{j,k \neq j} \mathbf{I}_j \cdot \mathbf{B}_{jk} \quad (8.2)$$

and the local field $\mathbf{B}_j \equiv \mathbf{B}_0 + \sum_{k \neq j} \mathbf{B}_{jk}$ takes the form appropriate to the case in question, whether high-field ($\mathbf{B}_j = \mathbf{B}_0 + \sum_{k \neq j} b(\mathbf{r}_{jk}) \{I_{k_x} \hat{\mathbf{x}} + I_{k_y} \hat{\mathbf{y}} - 2I_{k_z} \hat{\mathbf{z}}\}$), zero-field ($\mathbf{B}_j = \sum_{k \neq j} \frac{1}{r_{jk}^3} \{3(\mathbf{I}_k \cdot \hat{\mathbf{r}}_{jk}) \hat{\mathbf{r}}_{jk} - \mathbf{I}_k\}$), or intermediate-field ($\mathbf{B}_j = \mathbf{B}_0 + \sum_{k \neq j} \frac{1}{r_{jk}^3} \{3(\mathbf{I}_k \cdot \hat{\mathbf{r}}_{jk}) \hat{\mathbf{r}}_{jk} - \mathbf{I}_k\}$). The spatially inhomogeneous initial condition of the lattice may be represented by the density operator¹

$$\rho(0) = \frac{\exp(-\sum_j \beta_j I_{j_z})}{\text{Tr} \{ \exp(-\sum_j \beta_j I_{j_z}) \}} \quad (8.3)$$

where $\beta_j \equiv \hbar\omega_0/k_B T_j$, with $\hbar\omega_0$ a characteristic energy ($\omega_0 \sim -\gamma B_0$ in high field) and T_j a local spin “temperature” at lattice site j . In other words, we choose a local Boltzmann distribution for expected magnetization at each spin site, with site-to-site variations expressed as variations in the Boltzmann exponent β_j . A sinusoidal magnetization disturbance corresponds to sinusoidal variation of β_j along an axis of choice. As the disturbance dissipates, we expect the system to approach thermal equilibrium with $T_j = T$ for all j .²

¹This choice of density function does not include any initial correlations among transverse spin components, such as would be required for an initial disturbance in interspin energy.

²Of course, the action of the dipolar hamiltonian will also produce new spin-spin correlations, but we may either trust that these are small in the case of high applied field (cf. Ref. [28]) or we

In order to characterize the transport of magnetization across the lattice, we seek average spin trajectories represented by the expectation values $\{\langle \mathbf{I}_j \rangle(t)\}$. The quantum equations of motion are

$$\frac{d\langle \mathbf{I}_j \rangle}{dt} = i\langle [\mathcal{H}, \mathbf{I}_j] \rangle = \langle \mathbf{I}_j \times \mathbf{B}_j \rangle \quad (8.4)$$

Due to the noncommutation of the operators $\{I_{j_x}, I_{j_y}, I_{j_z}\}$, $\langle \mathbf{I}_j \times \mathbf{B}_j \rangle \neq \langle \mathbf{I}_j \rangle \times \langle \mathbf{B}_j \rangle$ for a coupled spin system (since both $\mathbf{I}_j(t)$ and $\mathbf{B}_j(t)$ will in general depend in a complicated fashion upon all the components of \mathbf{I}_j). Consequently, the classical differential equations of Chapters 6 and 7 do not suffice to describe the dynamics. On the other hand, it is not feasible to diagonalize quantum Hamiltonians \mathcal{H} for the large numbers of spins needed to model diffusion effects, and an alternative approach to quantum simulations is needed. The beginnings of such an approach can be constructed, and the connection between quantum and classical dynamics can be made explicit, by use of spin coherent states.

8.3 Spin Coherent States – An Introduction

Unlike the standard basis of Zeeman states $|I, m\rangle$, spin coherent states [46][47][48][49] are an overcomplete set of states with the minimum uncertainty product allowed by the Heisenberg uncertainty relations. In this and other respects, they are analogous to the canonical coherent states for the harmonic oscillator. Radcliffe [46] was the first to identify this class of states in 1971, following the introduction of the canonical coherent states by Glauber [50][51][52] in 1963. Subsequent work has identified both of these state constructs as members of a family of “generalized coherent states” which may be built around arbitrary symmetry groups [48][49][53]. The spin coherent states (SCS) are generated by rotations of a single base state, called the “fiducial state,”

may rely on the calculations to be described below to define a diffusion constant without reference to spin temperature per se.

chosen from the normal Hilbert space. Formally,

$$|g\rangle = T^I(g)|\Psi_0\rangle \quad g \in SU(2) \quad (8.5)$$

where $T^I(g)$ is the representation of an element of the rotation group $SU(2)$. We may choose $|\Psi_0\rangle$ to be a Zeeman state $|I, m\rangle$ and let $T^I(g)$ be the familiar unitary representation parametrized by Euler angles (α, β, γ) : $T^I(g) = \exp(i\alpha I_z) \exp(i\beta I_y) \exp(i\gamma I_z)$. The action of $\exp(i\gamma I_z)$ upon $|I, m\rangle$ is merely to generate a phase $\exp(i\gamma m)$ which may be chosen equal to unity, since states in the Hilbert space are only defined up to a complex phase factor. Taking as our z axis the quantization axis of $|\Psi_0\rangle$, and recasting (β, α) as an inclination and azimuth (θ, ϕ) , we may parametrize the SCS by a unit vector $\mathbf{n} = \sin \theta \cos \phi \hat{\mathbf{x}} + \sin \theta \sin \phi \hat{\mathbf{y}} + \cos \theta \hat{\mathbf{z}}$ as follows:³

$$\begin{aligned} |\mathbf{n}\rangle &= D(\mathbf{n})|\Psi_0\rangle \\ D(\mathbf{n}) &\equiv \exp(i\phi I_z) \exp(i\theta I_y) = \exp(i\theta(-\sin \phi \hat{\mathbf{x}} + \cos \phi \hat{\mathbf{y}}) \cdot \mathbf{I}) \end{aligned} \quad (8.6)$$

In other words, the spin coherent states $\{|\mathbf{n}\rangle\}$ consist of all possible rotations of the state vector $|\Psi_0\rangle$ about any axis perpendicular to the starting quantization axis.

Such manipulations may seem somewhat arbitrary at first sight, but they are in practice quite familiar in nuclear magnetic resonance. An operation which sets in motion the paradigmatic time-domain NMR experiment – the application of a $\pi/2$ pulse to an initial state of spin polarization (say $|I = +\frac{1}{2}, m = +\frac{1}{2}\rangle$, or simply $|+\frac{1}{2}\rangle$) – in fact creates a spin coherent state:

$$\exp\left(i\frac{\pi}{2}I_y\right)|+\frac{1}{2}\rangle = 2^{-\frac{1}{2}}\left(|+\frac{1}{2}\rangle + |-\frac{1}{2}\rangle\right) \equiv |\mathbf{n} = \hat{\mathbf{x}}\rangle \quad (8.7)$$

The next phase of the basic NMR experiment – free precession in an external magnetic field – also has a simple interpretation in terms of spin coherent states. Free

³a similar definition may be arrived at using more general arguments for an arbitrary representation $T^I(g)$ (see Ref. [49], Chapter 4 for details). Alternative complex parametrizations of the SCS also exist.

precession may itself be described by a rotation operator $\exp(i\omega_0 t I_z)$ (for a constant field $\mathbf{B}_0 = B_0 \hat{\mathbf{z}}$). The group property of rotations guarantees that this rotation, when combined with the initial pulse which created the state $|\hat{\mathbf{x}}\rangle$, will yield another spin coherent state, up to a phase. For the particular case in question,

$$\begin{aligned}
\exp(i\omega_0 t I_z) |\hat{\mathbf{x}}\rangle &= \exp(i\omega_0 t I_z) \exp\left(i\frac{\pi}{2} I_y\right) \left|+\frac{1}{2}\right\rangle \\
&= \exp\left(i\frac{\pi}{2} \cos \omega_0 t I_y - \sin \omega_0 t I_x\right) \exp(i\omega_0 t I_z) \left|+\frac{1}{2}\right\rangle \\
&= \exp(+i\omega_0 t/2) \exp\left(i\frac{\pi}{2} (\cos \omega_0 t I_y - \sin \omega_0 t I_x)\right) \left|+\frac{1}{2}\right\rangle \\
&= \exp(+i\omega_0 t/2) |\mathbf{n}(t)\rangle = \cos \omega_0 t \hat{\mathbf{x}} + \sin \omega_0 t \hat{\mathbf{y}} \tag{8.8}
\end{aligned}$$

The behavior of the SCS under rotation bears a close relation to the usual description of spin operator rotations. Rotations of spin states in the Schrodinger picture take one spin coherent state into another, corresponding simply to motions in the parameter-space of the SCS. The coherent state labels $\{\mathbf{n}\}$ evolve precisely as would spin operators in the Heisenberg picture, or, for that matter, as would classical vectors subject to the equivalent rotations. All these properties are immediate consequences of the definitions (8.5) and (8.6). As it happens, these definitions lead to a surprising wealth of additional features which will form the basis of our arguments in coming Sections. Some of these useful features are summarized below:

- *Minimal uncertainty product:* If we choose as our fiducial state the Zeeman state $|I, I\rangle$, we gain a particular advantage: the Heisenberg uncertainty relation for the components of \mathbf{I} is saturated. Our state $|\Psi_0\rangle = |I, I\rangle$ satisfies

$$\Delta I_x^2 \Delta I_y^2 = \frac{1}{4} \langle I_z \rangle^2 = \frac{I^2}{4} \tag{8.9}$$

where $\Delta I_\alpha^2 \equiv \langle I_\alpha^2 \rangle - \langle I_\alpha \rangle^2$. By applying matching rotations to the spin components and to the states, $I_\alpha \rightarrow \tilde{I}_\alpha = D(\mathbf{n}) I_\alpha D^{-1}(\mathbf{n})$ and $|\Psi_0\rangle \rightarrow D(\mathbf{n}) |\Psi_0\rangle = |\mathbf{n}\rangle$, we may transform Eq. (8.9) to the form

$$\Delta \tilde{I}_x^2 \Delta \tilde{I}_y^2 = \frac{1}{4} \langle \tilde{I}_z \rangle^2 = \frac{I^2}{4} \tag{8.10}$$

The spin coherent state $|\mathbf{n}\rangle$ minimizes the uncertainty product in a transformed coordinate system with quantization axis \mathbf{n} . This again may be seen largely as a matter of definition, but here the definition has the useful implication that the SCS with this choice of fiducial state all share the optimal properties of the stretched state $|I, I\rangle$.⁴ The SCS so chosen are all spin “wavepackets” with the minimum allowable extent. We have already established, furthermore, that the SCS wavepackets undergo no spreading with rotation (i.e. one optimal wavepacket is mapped smoothly onto another). Analogously, in a harmonic oscillator potential, the wavepackets of canonical coherent states may be shown to “bounce” back and forth like classical oscillators, without any quantum mechanical mixing or spreading.⁵ It appears that, at least as regards rotational behavior, the SCS are “nearly classical.”

- *Relation to the Zeeman basis $|I, m\rangle$:* What do the set of optimal SCS look like? Since they are, after all, legitimate states in the Hilbert space, they can be decomposed in the usual Zeeman basis:

$$|\mathbf{n}\rangle = \sum_m |I, m\rangle \langle I, m|\mathbf{n}\rangle$$

$$\langle I, m|\mathbf{n}\rangle = \binom{2I}{I-m}^{\frac{1}{2}} \left(\cos\frac{\theta}{2}\right)^{I+m} \left(\sin\frac{\theta}{2}\right)^{I-m} \exp(i(I-m)\phi) \quad (8.11)$$

The square magnitude of the overlap

$$|\Psi_m(\theta, \phi)|^2 \equiv |\langle I, m|\mathbf{n}\rangle|^2 = \binom{2I}{I-m} \left(\cos^2\frac{\theta}{2}\right)^{I+m} \left(\sin^2\frac{\theta}{2}\right)^{I-m} \quad (8.12)$$

is plotted as a function of θ and ϕ for a variety of I and m values in Figure 8-2.

⁴It is perhaps worth noting that $|\Psi_0\rangle = |I, -I\rangle$ is also an optimal choice of fiducial vector in the sense defined above, and is distinguished from our choice essentially by a change of sign in various matrix elements. Radcliffe [46] uses $|I, I\rangle$ as his starting vector, whereas Perelomov [49] chooses $|I, -I\rangle$.

⁵In the limit of $I \rightarrow \infty$, the spin coherent states may in fact be shown to contract into the canonical coherent states, for in this limit both are derived from a Hilbert space characterized by equally-spaced manifolds of states (cf. Ref. [46], for example).

The function $|\Psi_m(\theta, \phi)|^2$ is a probability density function describing a distribution of vector orientations peaked at $\cos\theta = \frac{m}{I}$. One may observe that in the quantum limit ($I = \frac{1}{2}$), the distributions are broad, and a wide range of SCS vector orientations are contained in $|I, m\rangle$, whereas $|\Psi_m|^2$ collapses to well-defined cones in the classical limit of large I . It can be argued that the cones define an expected orientational distribution for pure states $|I, m\rangle$, and the collapse of the overlap probability to this base distribution implies that in the vicinity of the classical limit, the SCS are essentially angular delta functions, better known as classical vectors. Let us make this “picture” of spin wavepackets more precise using a simple geometric construction. For a pure Zeeman state $|I, m\rangle$, the inclination angle, defined as $\vartheta = \cos^{-1}\left(\frac{\langle I_z \rangle}{\langle |\mathbf{I}| \rangle}\right) = \cos^{-1}\left(\frac{m}{(I(I+1))^{1/2}}\right)$ is known with certainty. Consequently, the azimuthal phase φ is entirely indeterminate. ($\Delta I_x^2 + \Delta I_y^2 = \langle I_x^2 + I_y^2 \rangle = I(I+1) - m^2$, and the uncertainty in a measurement of the transverse component of \mathbf{I} is as large as the component itself.) Although the noncommutation of the three components of \mathbf{I} precludes true visualization in angular space (simultaneous eigenstates labelled by three c-number vector components cannot be constructed), the cone serves as a convenient mnemonic for the probable results of measurements.⁶ While the pure states are represented as cones symmetric about the z axis with various opening angles, the SCS may be seen as rotated cones of fixed opening angle. The optimal SCS set corresponds to cones of minimal width $\vartheta = \cos^{-1}\left(\frac{I}{(I(I+1))^{1/2}}\right)$, which do indeed collapse to simple vectors in the classical limit ($\vartheta \rightarrow 0$ as $I \rightarrow \infty$). The family of spin coherent states interpolates in a rather intuitive fashion between the classical and the quantum regimes.

- *Overcompleteness:* Any basis of the Hilbert space for spin I must contain only $2I + 1$ basis vectors, whereas the continuously-parametrized SCS form an infinite set. The SCS must be overcomplete. Indeed, one may compute the overlap of

⁶Actually, the range of values that can result from measurements of spin components is also limited. In the case of a spin-1/2, only four points on the cone are allowed, corresponding to measured (x, y) values of $(+1/2, +1/2)$, $(+1/2, -1/2)$, $(-1/2, +1/2)$, and $(-1/2, -1/2)$.

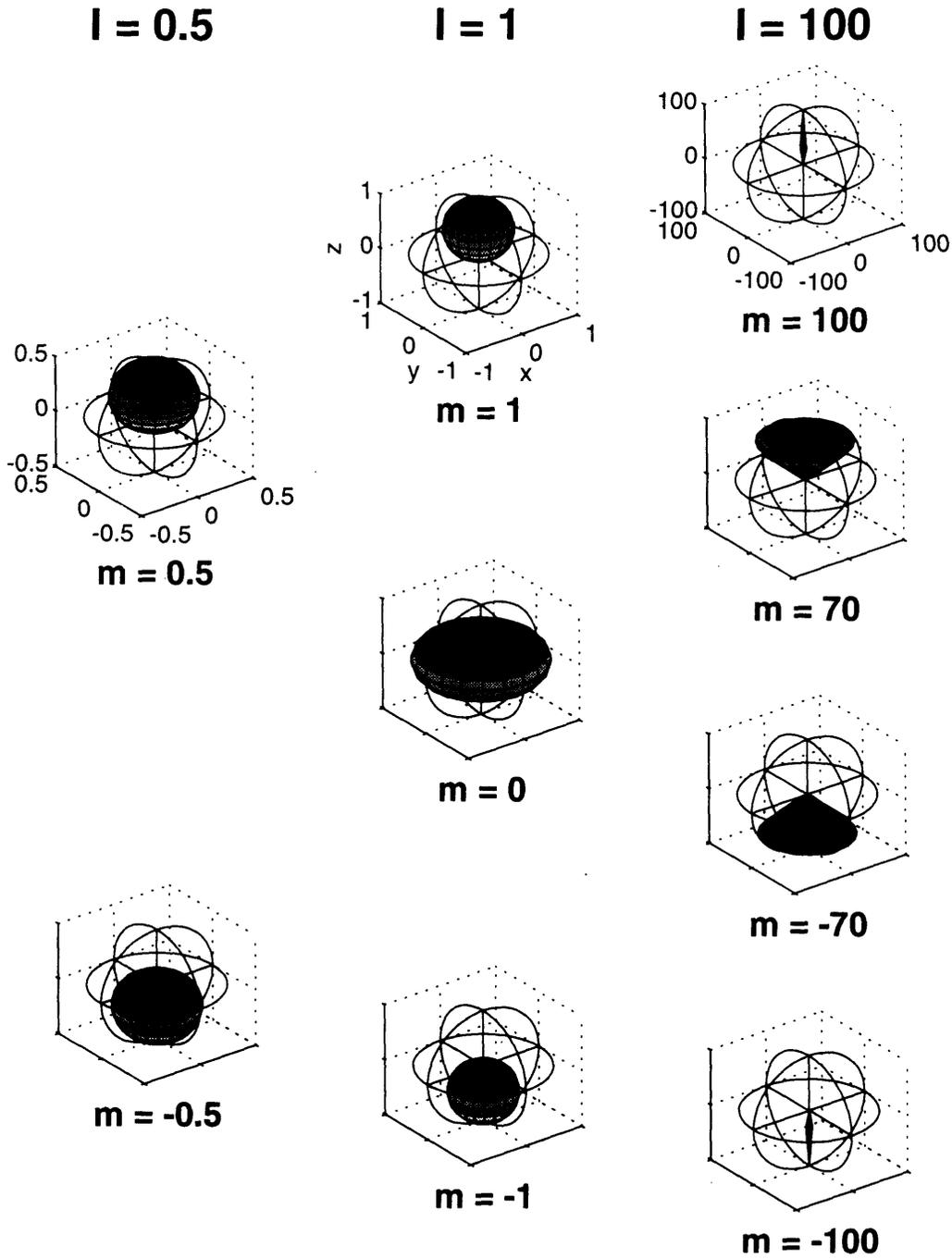


Figure 8-2: Angular plot of the overlap of spin coherent states with the usual Zeeman states. The function $|\Psi_m(\theta, \phi)|^2 \equiv |\langle I, m | \mathbf{n} \rangle|^2$ (normalized to a maximum value of I) is plotted for various I and m .

two distinct states $|\mathbf{n}\rangle$ and $|\mathbf{n}'\rangle$ as

$$\begin{aligned}\langle \mathbf{n}' | \mathbf{n} \rangle &= \left(\cos \frac{\theta}{2} \cos \frac{\theta'}{2} + \sin \frac{\theta}{2} \sin \frac{\theta'}{2} e^{i(\phi - \phi')} \right)^{2I} \\ &= \exp(iIA(\mathbf{n}', \mathbf{n}, \mathbf{n}_0)) \left(\frac{1 + \mathbf{n} \cdot \mathbf{n}'}{2} \right)^I\end{aligned}\quad (8.13)$$

$A(\mathbf{n}', \mathbf{n}, \mathbf{n}_0)$ is the area of the geodesical triangle formed by the vertices \mathbf{n}' , \mathbf{n} , and $\mathbf{n}_0 = \hat{\mathbf{z}}$. The overlap probability distribution $((1 + \mathbf{n} \cdot \mathbf{n}')/2)^{2I}$ is plotted in Figure 8-3 for $\mathbf{n}' = \hat{\mathbf{z}}$. We observe once again that the SCS collapse to vectors for large I .

- *Resolution of unity:* One prerequisite of any set of generalized coherent states is a resolution of unity akin to the completeness formula for basis vectors, $\mathbb{1} = \sum_m |m\rangle\langle m|$. For the SCS set, the relevant expression is

$$\mathbb{1} = \int d\mu |\mathbf{n}\rangle\langle \mathbf{n}| \quad d\mu = \frac{2I+1}{4\pi} \sin\theta d\theta d\phi = \frac{2I+1}{4\pi} d\Omega \quad (8.14)$$

This property allows insertion of complete sets of spin coherent states into inner products or operator expressions.

- *Diagonal representation of operators:* Any bounded operator A in the Hilbert space, not merely the identity operator, admits of a diagonal representation in terms of spin coherent states:

$$A = \int d\mu \mathcal{A}(\mathbf{n}) |\mathbf{n}\rangle\langle \mathbf{n}| \quad (8.15)$$

$\mathcal{A}(\mathbf{n})$ is a unique c-number representative of A , and is called the symbol of the operator. When A is equal to the spin operator \mathbf{I} , the diagonal symbol \mathcal{I} takes a particularly simple form:

$$\mathcal{I}(\mathbf{n}) = (I+1)\mathbf{n} \quad (8.16)$$

- *Operator matrix elements in the spin coherent state representation:* Diagonal

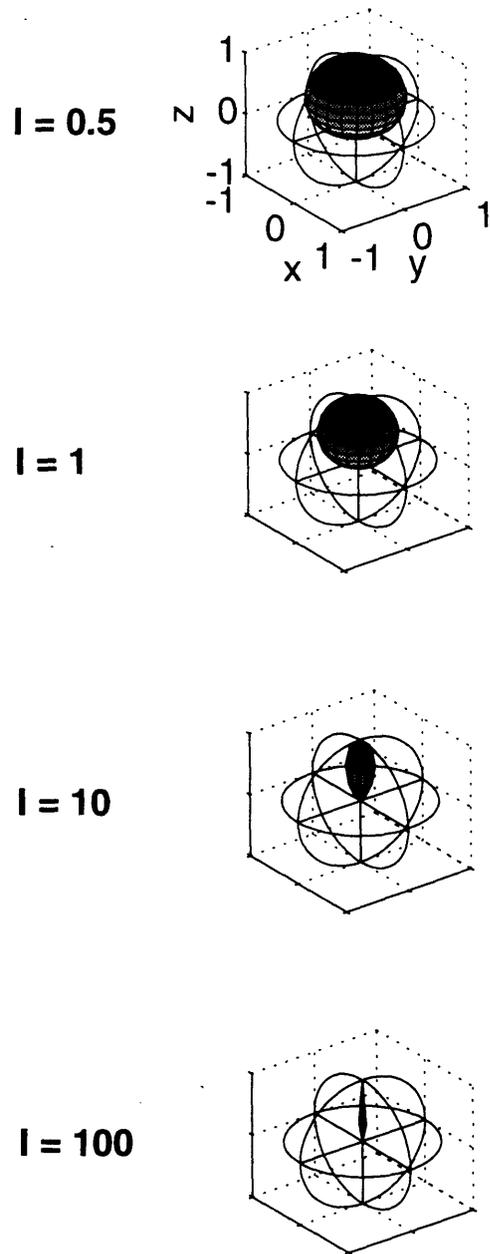


Figure 8-3: Angular plot of the mutual overlap of spin coherent states. The function $|\Psi_z(\theta, \phi)|^2 \equiv |\langle \mathbf{n} | \mathbf{n}' = \hat{\mathbf{z}} \rangle|^2$ is plotted for various values of spin quantum number I .

matrix elements of operators in the SCS representation also form unique symbols (distinct from the diagonal symbols above). Diagonal and off-diagonal elements for several spin operators of interest to us are listed below:

$$\langle \mathbf{n} | \mathbf{I} | \mathbf{n} \rangle = I \mathbf{n} \quad (8.17)$$

$$\langle \mathbf{n}' | \mathbf{I} | \mathbf{n} \rangle = \langle \mathbf{n}' | \mathbf{n} \rangle I \frac{\mathbf{n} + \mathbf{n}' + i(\mathbf{n} \times \mathbf{n}')}{1 + \mathbf{n} \cdot \mathbf{n}'} \equiv \langle \mathbf{n}' | \mathbf{n} \rangle I \boldsymbol{\eta}(\mathbf{n}, \mathbf{n}') \quad (8.18)$$

$$\langle \mathbf{n} | e^{-\beta I_z} | \mathbf{n} \rangle = \left(\cosh \frac{\beta}{2} - \sinh \frac{\beta}{2} \cos \theta \right)^{2I} \quad (8.19)$$

$$\begin{aligned} \langle \mathbf{n}' | e^{-\beta I_z} | \mathbf{n} \rangle &= e^{-\beta I} \left(\cos \frac{\theta}{2} \cos \frac{\theta'}{2} + e^{\beta} \sin \frac{\theta}{2} \sin \frac{\theta'}{2} e^{i(\phi - \phi')} \right)^{2I} \\ &= \langle \mathbf{n}' | \mathbf{n} \rangle \left(\cosh \frac{\beta}{2} - \sinh \frac{\beta}{2} \eta_z \right)^{2I} \end{aligned} \quad (8.20)$$

- *Traces:* Since we will ultimately be interested in the expectation values of spin operators, $\langle I_\alpha \rangle = Tr \{ \rho I_\alpha \}$, we will require an expression for the trace in terms of spin coherent states. The trace of an operator A may be written either as

$$Tr \{ A \} = \int d\mu \langle \mathbf{n} | A | \mathbf{n} \rangle \quad (8.21)$$

or as

$$Tr \{ A \} = \int d\mu A(\mathbf{n}) \quad (8.22)$$

The second expression may be derived from the first by insertion of the diagonal representation of A and extraction of a resolution of unity.

- *Product rule for traces:* Finally, we would like an expression for the trace of a product of operators. Using the properties listed so far, we write

$$\begin{aligned} Tr \{ AB \} &= \int d\mu \langle \mathbf{n} | AB | \mathbf{n} \rangle \\ &= \int d\mu d\mu' \langle \mathbf{n} | \mathbf{n}' \rangle A(\mathbf{n}') \langle \mathbf{n}' | B | \mathbf{n} \rangle = \int d\mu d\mu' A(\mathbf{n}') \langle \mathbf{n}' | B | \mathbf{n} \rangle \langle \mathbf{n} | \mathbf{n}' \rangle \\ &= \int d\mu' A(\mathbf{n}') \langle \mathbf{n}' | B | \mathbf{n}' \rangle \\ &= \int d\mu' B(\mathbf{n}') \langle \mathbf{n}' | A | \mathbf{n}' \rangle \end{aligned} \quad (8.23)$$

Suggestive c-number expressions may be derived for the product symbol $\mathcal{AB}(\mathbf{n})$ [54][55], but they will not be of use to us here.

In summary, a spin coherent state $|\mathbf{n}\rangle$ represents a spin wavefunction centered about the classical vector $I\mathbf{n}$. The SCS maintain their coherent character under unitary rotations, and for a suitable choice of fiducial state, $\Psi_0 = |I, \pm I\rangle$, they have a minimal uncertainty product. In these respects, they are an appealing analogue for classical spins. They may indeed be said to be the most “classical” of allowed quantum spin states – the closest one can get while still respecting quantum constraints. As such, they have been used in the theoretical physics community to explore the classical limit of quantum spin systems [56][57] and to formulate path integrals for spin problems [58][59][60][61][62].

8.4 Spin Coherent States and Geometrical Phase

One interesting consequence of the phase convention used in defining spin coherent states (Eq. (8.6)) has not been spelled out clearly in the literature. It therefore merits a brief excursion here. In Section 8.3 above, it was mentioned that any rotation operator applied to a spin coherent state produces another spin coherent state, up to a phase. It turns out that this phase is the geometrical phase first identified by Berry [63] for cyclic quantum evolutions. To see this, we note that a spin coherent state $|\mathbf{n}\rangle$ is defined with no rotation about the tilted quantization axis \mathbf{n} (the Euler angle γ is set to 0). If we are to be consistent, we must maintain this definition through all subsequent transformations in the Hilbert space. Let us associate with the coherent state $|\mathbf{n}\rangle$ an axis system characterized by the triad $\{\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, \hat{\mathbf{n}}_3\}$, with $\hat{\mathbf{n}}_3$ lying along \mathbf{n} . A general rotation $T^I(g)$ which takes \mathbf{n} to \mathbf{n}_g will map the original triad to a new one $\{\hat{\mathbf{n}}_{g1}, \hat{\mathbf{n}}_{g2}, \hat{\mathbf{n}}_{g3}\}$. Due to the curvature of the sphere, the new axis system will be skewed with respect to the triad $\{\hat{\mathbf{n}}'_{g1}, \hat{\mathbf{n}}'_{g2}, \hat{\mathbf{n}}'_{g3} = \hat{\mathbf{n}}_{g3}\}$ that would have been generated by tilting directly from the z axis, as would be necessary to create a coherent state $|\mathbf{n}_g\rangle$ with zero phase. The extra rotation (which, being a rotation about \mathbf{n}_g , amounts to a complex phase factor multiplying the state $|\mathbf{n}_g\rangle$) is given by

the holonomy transformation associated with parallel transport of our triad around the sphere from \mathbf{n} to \mathbf{n}_g . As discussed by Anandan and Stodolsky [64] for a similar construction (not involving the SCS), this transformation takes the form $\exp(i\gamma I_z)$, where $\gamma = \int_{\Sigma} R d\Sigma$, R is the Gaussian curvature on the sphere, and Σ is the part of the spherical surface bounded by the path of the quantization axis from $\hat{\mathbf{z}}$ to \mathbf{n} to \mathbf{n}_g . On the unit sphere, $R = radius^{-2} = 1$, and $\gamma = A(\mathbf{n}_g, \mathbf{n}, \mathbf{n}_0)$ is the area of the geodesical triangle formed by the vertices \mathbf{n}_g , \mathbf{n} , and $\mathbf{n}_0 = \hat{\mathbf{z}}$: a term we have encountered earlier in Eq. (8.13). When a series of rotations is strung together into a cyclic path, beginning and ending at state $|\hat{\mathbf{z}}\rangle \equiv |I, m\rangle$, the areas of adjacent triangles add together, and the net phase accumulated is m times the solid angle subtended by the path as seen from the center of the sphere.⁷ This is precisely Berry's phase.

Perelomov [49] has characterized the phase generated by rotation of a spin coherent state, but has not made the association with Berry's phase explicit. Bose and Dutta-Roy [65] calculate a geometric phase for spin coherent states, but they do so by reexpressing the SCS in terms of Heisenberg-Weyl operators for the canonical coherent states and invoking the adiabatic limit. The present exposition is intended to emphasize that the SCS have Berry's geometrical phase "built-in." The separation of dynamics from geometry is intrinsic to the definition of the SCS, and does not require the adiabatic limit. This is consistent with the more general description of geometric phase by Aharonov and Anandan [66] in terms of motion in projective Hilbert space. Experiments in nuclear magnetic resonance and other modalities have distinguished geometrical from dynamical phase accumulation in both the adiabatic and the non-adiabatic limits [67]. The spin coherent states, it seems, are a natural theoretical tool for separating the dynamical from the geometrical, the motion of the spin from the motion of the reference frame. Not only do they boast an "almost classical" nature, but they also contain all the subtleties associated with quantum states, and they bear a special relation to the geometry of quantum evolution [68][65].

⁷In the course of a circuit C , $|I, m\rangle \rightarrow \exp(i(\int_C \gamma)I_z) |I, m\rangle = \exp(im \int_C \gamma) |I, m\rangle$.

8.5 Spin Coherent States and Quasi-Classical Quantum Dynamics

We return now to the task of comparing classical and quantum spin diffusion. Let us first express the quantum equations of motion (8.4) in terms of spin coherent states. Choosing the Heisenberg picture, in which the states and the density matrix are constant while the operators evolve, we write

$$Tr \{ \rho(0) \dot{\mathbf{I}}_j(t) \} = Tr \{ \rho(0) \mathbf{I}_j(t) \times \mathbf{B}_j(t) \} \quad (8.24)$$

Since the operators representing spins at different lattice sites commute, we may decompose a state of the lattice into a direct product of states for individual spin sites. For spin coherent states in particular,

$$|\{\mathbf{n}\}\rangle \equiv \bigotimes_j |\mathbf{n}_j\rangle \quad (8.25)$$

Now we use the trace formulae of Section 8.3, with the notation $d\{\mu\} \equiv \prod_k d\mu_k$. The left-hand side of Eq. (8.24) is

$$Tr \{ \rho(0) \dot{\mathbf{I}}_j(t) \} = \int d\{\mu\} \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle \dot{\mathcal{I}}_j(\{\mathbf{n}\}, t) \quad (8.26)$$

The right-hand side may be expanded as follows:

$$\begin{aligned} Tr \{ \rho(0) \mathbf{I}_j \times \mathbf{B}_j \} &= \int d\{\mu\} \langle \{\mathbf{n}\} | \rho(0) \mathbf{I}_j(t) \times \mathbf{B}_j(t) | \{\mathbf{n}\} \rangle \\ &= \int d\{\mu\} d\{\mu'\} \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}'\} \rangle \mathcal{I}_j(\{\mathbf{n}'\}, t) \times \langle \{\mathbf{n}'\} | \mathbf{B}_j(t) | \{\mathbf{n}\} \rangle \\ &= \int d\{\mu\} d\{\mu'\} d\{\mu''\} \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}'\} | \{\mathbf{n}''\} \rangle \langle \{\mathbf{n}''\} | \{\mathbf{n}\} \rangle \\ &\quad \mathcal{I}_j(\{\mathbf{n}'\}, t) \times \mathcal{B}_j(\{\mathbf{n}''\}, t) \\ &= \int d\{\mu'\} d\{\mu''\} \langle \{\mathbf{n}''\} | \rho(0) | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}'\} | \{\mathbf{n}''\} \rangle \\ &\quad \mathcal{I}_j(\{\mathbf{n}'\}, t) \times \mathcal{B}_j(\{\mathbf{n}''\}, t) \end{aligned} \quad (8.27)$$

The final contraction in (8.27) is achieved by moving the factor $\langle \{\mathbf{n}''\} | \{\mathbf{n}\} \rangle$ to the front of the integrand and removing a resolution of unity: $\int d\{\mu\} | \{\mathbf{n}\} \rangle \langle \{\mathbf{n}\} |$. Combining left-hand and right-hand expressions, and relabelling integration indices, we arrive at the equation

$$\begin{aligned} \int d\{\mu\} \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle \dot{\mathcal{I}}_j(\{\mathbf{n}\}, t) = \\ \int d\{\mu\} d\{\mu'\} \langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}'\} | \rho(0) | \{\mathbf{n}\} \rangle \mathcal{I}_j(\{\mathbf{n}\}, t) \times \mathcal{B}_j(\{\mathbf{n}'\}, t) \end{aligned} \quad (8.28)$$

Solution of this integrodifferential expression for the spin symbols $\mathcal{I}_j(\{\mathbf{n}\}, t)$ constitutes a complete solution of the problem, since with the symbols in hand we may immediately calculate the desired expectation values,

$$\langle \mathbf{I}_j \rangle(t) = Tr \{ \rho(0) \mathbf{I}_j(t) \} = \int d\{\mu\} \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle \mathcal{I}_j(\{\mathbf{n}\}, t) \quad (8.29)$$

Eq. (8.28) as written is difficult to solve. If our initial lattice configuration is such that the density operator $\rho(0)$ admits the following approximation

$$\langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}'\} | \rho(0) | \{\mathbf{n}\} \rangle \approx \langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}'\} | \{\mathbf{n}\} \rangle \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle \quad (8.30)$$

a substantial simplification may be achieved. Inserting expression (8.30) into (8.28) above, and identifying a diagonal representation of the local field operator $\mathbf{B}_j(t)$ ($\mathbf{B}_j(t) = \int d\{\mu'\} \mathcal{B}_j(\{\mathbf{n}'\}, t) | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}'\} |$), we obtain

$$\begin{aligned} \int d\{\mu\} \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle \dot{\mathcal{I}}_j(\{\mathbf{n}\}, t) \approx \\ \int d\{\mu\} \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle \mathcal{I}_j(\{\mathbf{n}\}, t) \times \langle \{\mathbf{n}\} | \mathbf{B}_j(t) | \{\mathbf{n}\} \rangle \end{aligned} \quad (8.31)$$

The form of $\rho(0)$ is arbitrary, at least within the constraints of the approximation (8.30), so that we may equate the integrands

$$\dot{\mathcal{I}}_j(\{\mathbf{n}\}, t) \approx \mathcal{I}_j(\{\mathbf{n}\}, t) \times \langle \{\mathbf{n}\} | \mathbf{B}_j(t) | \{\mathbf{n}\} \rangle \quad (8.32)$$

At time $t = 0$, the Heisenberg operators $\mathbf{I}_j(t)$ coincide with the usual spin operators \mathbf{I}_j , and the symbols are simply $\mathcal{I}_j(\{\mathbf{n}\}, 0) = (I + 1)\mathbf{n}_j$. The diagonal matrix elements of the local field operator $\langle \{\mathbf{n}\} | \mathbf{B}_j(0) | \{\mathbf{n}\} \rangle$ reduce to the classical functions $\mathbf{B}_j(\{I\mathbf{n}\}, 0)$, with each spin component I_{k_α} in the operator expression replaced by the corresponding component of a classical vector In_{k_α} . The result is

$$\dot{\mathbf{n}}_j(0) \approx \mathbf{n}_j(0) \times \mathbf{B}_j(\{I\mathbf{n}\}, 0) \quad (8.33)$$

This equation is just the familiar classical Bloch equation evaluated at time zero. Consequently, the time-dependent functions $In_j(t)$ will evolve precisely as classical spin vectors $\mathbf{m}_j(t)$ for all time, subject to the classical equations of motion

$$\dot{\mathbf{m}}_j(t) \approx \mathbf{m}_j(t) \times \mathbf{B}_j(t) \quad (8.34)$$

With the determination of $\mathcal{I}_j(\{\mathbf{n}\}, t) = (I + 1)\mathbf{n}_j(\{\mathbf{n}\}, t)$ having proceeded along classical lines, only integration against the weight function $\langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle$ remains (cf. Eq. (8.29)). Referring to Eq. (8.3) and substituting the diagonal matrix elements of Eq. (8.19) gives

$$\begin{aligned} \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle &= \frac{\langle \{\mathbf{n}\} | \exp(-\sum_j \beta_j I_{jz}) | \{\mathbf{n}\} \rangle}{\text{Tr} \left\{ \exp(-\sum_j \beta_j I_{jz}) \right\}} \\ &= \prod_j \frac{\left(\cosh \frac{\beta_j}{2} - \sinh \frac{\beta_j}{2} \cos \theta_j \right)^{2I}}{\int d \cos \theta_j \left(\cosh \frac{\beta_j}{2} - \sinh \frac{\beta_j}{2} \cos \theta_j \right)^{2I}} \end{aligned} \quad (8.35)$$

In the high-temperature limit ($\beta_j \ll 1$), this function reduces to a Boltzmann distribution

$$\langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle \xrightarrow{\beta_j \ll 1} \prod_j \frac{\exp(-\beta_j I \cos \theta_j)}{\int d(I \cos \theta_j) \exp(-\beta_j I \cos \theta_j)} \quad (8.36)$$

where $I \cos \theta_j$ plays the role of z component of a classical magnetization vector.⁸

⁸In any case, for arbitrary temperature, the weight function of Eq. (8.35) represents a normalized

We may then view the expectation value expression as an ensemble integral over a classical lattice. Since we are ultimately interested in a bulk parameter rather than the true ensemble average of any individual spin magnetization, we may select one or several representative lattice configurations and evaluate the spin diffusion constant. The correspondence with the classical calculations of Chapter 7 is now complete.

Under what physical conditions is the approximation (8.30) valid? The expression (8.20) for the off-diagonal elements of an exponential spin operator gives

$$\langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}'\} | \rho(0) | \{\mathbf{n}\} \rangle = \langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}'\} | \{\mathbf{n}\} \rangle \frac{\prod_j \left(\cosh \frac{\beta_j}{2} - \sinh \frac{\beta_j}{2} \eta_{jz} \right)^{2I}}{\text{Tr} \left\{ \exp \left(- \sum_j \beta_j I_{jz} \right) \right\}} \quad (8.37)$$

with the complex vector $\boldsymbol{\eta}_j$ defined as in Eq. (8.18):

$$\boldsymbol{\eta}_j \equiv \frac{\mathbf{n}_j + \mathbf{n}'_j + i (\mathbf{n}_j \times \mathbf{n}'_j)}{1 + \mathbf{n}_j \cdot \mathbf{n}'_j} \quad (8.38)$$

For large spin quantum number I , the prefactor $|\langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle|^2 = \prod_j \left((1 + \mathbf{n}_j \cdot \mathbf{n}'_j) / 2 \right)^{2I}$ vanishes for all \mathbf{n}'_j which are not very near \mathbf{n}_j , whilst the resolution of unity (8.14) guarantees a unit integral. Thus, the prefactor approaches a product of delta functions $\prod_j \delta(\mathbf{n}'_j - \mathbf{n}_j)$, and η_{jz} may be replaced by $\cos \theta_j$, justifying the classical approximation. These arguments show how the disappearance of complex interference terms accompanies the classical limit. The expectation value expressions for quantum spin operators in the SCS representation contract smoothly to classical values as the spin quantum number is increased.

For small I , closer to the “quantum limit” of $I = 1/2$ realized in a physical systems like CaF_2 , there is considerable latitude in the values of $\{\mathbf{n}'\}$, and the complex terms in η_{jz} may not be ignored. Nevertheless, under certain conditions, cancellations allow us to recover the classical results. At the high temperatures which are standard in many NMR experiments ($T \sim 300^\circ$, $\beta \sim 10^{-5}$ for fields of several Tesla), the exponential density operator may be expanded to first order in β , yielding the following expression

probability distribution, and the integral (8.29) may be evaluated by Monte-Carlo techniques.

for the left-hand side of (8.30):

$$\begin{aligned}
\langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle &\xrightarrow{\beta_j \ll 1} \langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}\} | \frac{\mathbb{1} - \sum_j \beta_j I_{jz}}{\text{Tr} \left\{ \exp \left(- \sum_j \beta_j I_{jz} \right) \right\}} | \{\mathbf{n}\} \rangle \\
&= \langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}'\} | \{\mathbf{n}\} \rangle \frac{1 - \sum_j \beta_j \eta_{jz}}{\text{Tr} \left\{ \exp \left(- \sum_j \beta_j I_{jz} \right) \right\}}
\end{aligned} \tag{8.39}$$

If the spatial variation in β_j is slow enough that many spins lie together in regions of nearly uniform spin temperature, then the imaginary terms in η_j for different spins in each subregion tend to interfere with one another⁹, and the real terms tend to accumulate to an average value somewhat less than $\sum_j \cos \theta_j$. The result is that the sum is “buffered” against variations in $\{\mathbf{n}'\}$, at least among the most probable configurations, and the predominant $\{\mathbf{n}'\}$ dependence lies in the prefactor $|\langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle|^2$. At the expense of a partial reduction in the effective magnitude of β_j for each subregion of the lattice, we may replace (8.39) with

$$\begin{aligned}
\langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle &\approx \langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}'\} | \{\mathbf{n}\} \rangle \frac{1 - \sum_j \bar{\beta}_j \cos \theta_j}{\text{Tr} \left\{ \exp \left(- \sum_j \beta_j I_{jz} \right) \right\}} \\
&\approx \langle \{\mathbf{n}\} | \{\mathbf{n}'\} \rangle \langle \{\mathbf{n}'\} | \{\mathbf{n}\} \rangle \langle \{\mathbf{n}\} | \rho(0) | \{\mathbf{n}\} \rangle
\end{aligned} \tag{8.40}$$

and (8.30) is satisfied. In the long-wavelength, high-temperature limit, we expect the quantum results to be reasonably well approximated by classical calculations, even for spins as low as $I = 1/2$.

A more complete assessment of the size and character of quantum corrections to the classical dynamical model may be possible using the SCS formalism. For example, one might imagine expanding expressions like Eq. (8.28) in powers of β . The resulting integrals over $\{\mathbf{n}'\}$ for corrections of first and higher orders suggest a model of interacting “virtual lattices,” in which spins respond not only to other spins in their home lattice $\{\mathbf{n}\}$ but also to spins in a suitably chosen set of accessory

⁹ $Im(\eta_{jz}) = (\mathbf{n}_j \times \mathbf{n}'_j)_z / (1 + \mathbf{n}_j \cdot \mathbf{n}'_j)$ is symmetric about 0, and sums over highly weighted or “representative” choices of $\{\mathbf{n}'\}$ vanish in the limit of large particle number.

lattices $\{\mathbf{n}'\}$.¹⁰ Takano [60][61] has designed a strategy for path integration using the SCS, and this strategy might well be modified to form a dynamical path integral (though for reasons of convergence this strategy is likely to be successful only for small numbers of interacting spins). The work of Takahashi and Shibata [55][54] (who derive differential operator equations for spin distribution functions using the product rules for diagonal symbols mentioned briefly in Section 8.3) might also be extended to large systems of dipole-coupled spins, though substantial approximations might be required to render the mathematics tractable.

Complete SCS-based simulations of quantum spin dynamics are possible for Hamiltonians consisting of uncoupled spin rotations, including arbitrary time-dependent rotations. Such simulations, however, provide no information not already available from calculations of spin operator rotations in the Heisenberg picture. Under the influence of spin-spin couplings, the SCS wavefunctions are subject to spreading, which severely hampers the calculations and necessitates the unpleasant integrodifferential equation (8.28). Nevertheless, we have demonstrated that under physical conditions of interest to us, the SCS expectation value expressions may be manipulated to justify classical simulations of spin diffusion, and to offer insight into the connection between classical and quantum spin dynamics.

¹⁰Such a construction would be reminiscent of the Trotter-Suzuki style of path integration for spin systems, which maps the quantum problem to an equivalent classical problem of higher dimensionality.

Epilogue

As-tu tué le Jaseroque?

Viens à mon coeur, fils rayonnais!

O jour frabbejeais! Calleau, callai!

Il cortule, dans sa joie.

— L. Carroll, francisé

Thus concludes our several chapters' quest, or, it might be said, inquest. By methods involving the pencil, the computer, and the spectrometer, we have aimed at the interrogation of nuclear spins coupled by the dipole interaction.

We have seen that, in the few-spin limit, the dipole-dipole coupling drives coherent magnetization exchange which can be fashioned into a convenient tool for the measurement of molecular structure. Broadband dipolar recoupling experiments have successfully identified internuclear distances and coupling patterns in single amino acids and other test molecules, and they are currently being applied to the study of more complex biomolecules.

In the opposite many-particle limit, the same dipolar interactions are responsible for the spatial transport of spin magnetization and energy. Classical simulations of spin diffusion offer insight into the microscopic underpinnings of the spin diffusion process, highlighting in particular the role of conservation principles in diffusive spin dynamics, and the interplay of local and global effects in spin systems of various dimensionalities. Furthermore, while the dynamics underlying the essentially classical picture of a precessing fictitious spin in Part I are quantum mechanical in their details, spin diffusion affords an interesting case study for the comparison of classical and

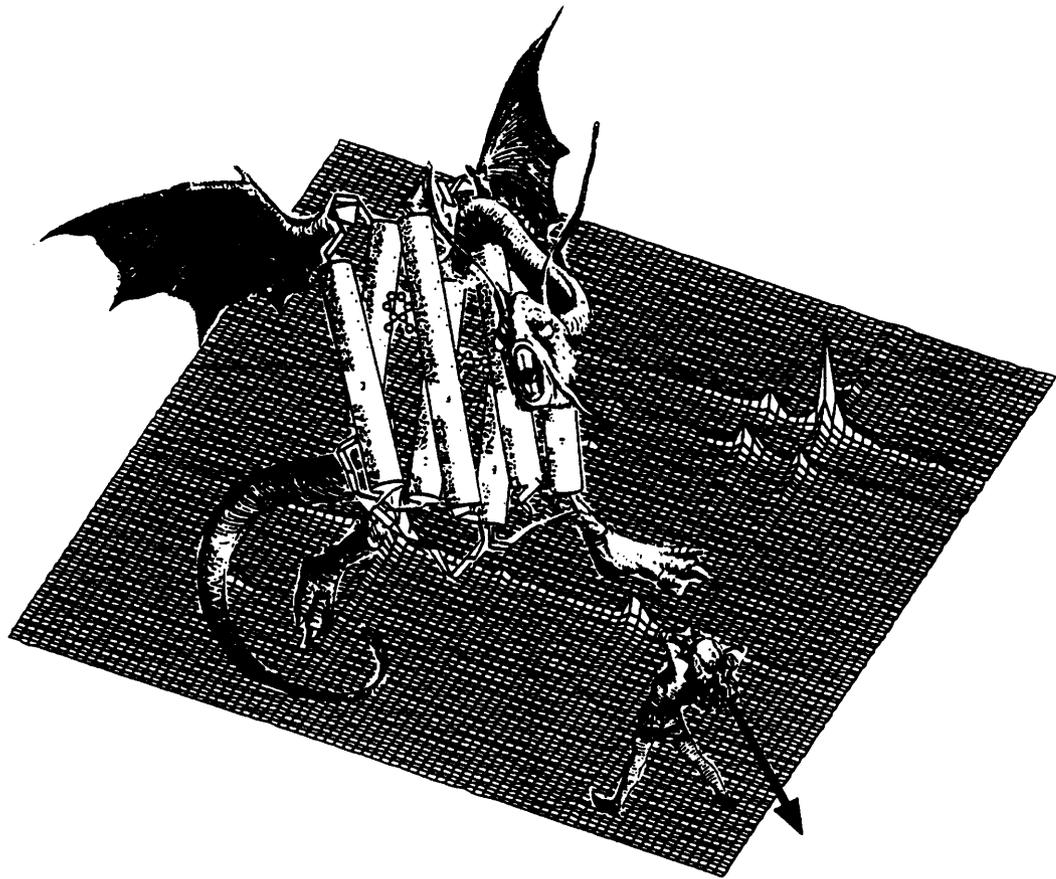
quantum dynamics in extended spin systems. Spin coherent states offer at least the beginnings of a mathematical technology for performing this comparison. One who wishes to extend the comparison further, and to embark upon a program of frank quantum-mechanical simulation, is perhaps best advised to follow the example of the hero of Lewis Carroll's *Jabberwocky*:

He took his vorpal sword in hand:

Long time the manxome foe he sought –

So rested he by the Tumtum tree,

And stood awhile in thought.



Appendix A

Evaluation of Moments in the Theory of Redfield and Yu

By characterizing the linear response of spin magnetization to a small perturbing magnetic field, and assuming that this response obeys a diffusion equation, Redfield and Yu in Ref. [26] derive the approximate expression given earlier for D_Z :

$$D_Z = \left(\frac{M_2}{k^2}\right) \left(\frac{\pi M_2}{2M_4}\right)^{\frac{1}{2}} \quad (\text{A.1})$$

In the limit of long wavelength (cf. [25]), M_2 and M_2/M_4 may be written as

$$M_2 = \frac{\frac{1}{2}k^2 \sum_j x_{ij}^2 \text{Tr} \{[\mathcal{H}, I_{i_z}][\mathcal{H}, I_{j_z}]\}}{\hbar^2 \text{Tr} \{I_{i_z}^2\}} \quad (\text{A.2})$$

$$\frac{M_2}{M_4} = \frac{-\hbar^2 \sum_j x_{ij}^2 \text{Tr} \{[\mathcal{H}, I_{i_z}][\mathcal{H}, I_{j_z}]\}}{2 \sum_j x_{ij}^2 \text{Tr} \{[\mathcal{H}, [\mathcal{H}, I_{i_z}]] [\mathcal{H}, [\mathcal{H}, I_{j_z}]]\}} \quad (\text{A.3})$$

Here, $x_{ij}^2 = (x_i - x_j)^2$ are the squared separations of spins i and j in the direction of the initial wavevector, and \mathbf{I}_i and \mathbf{I}_j are quantum spin operators. Inserting the high-field truncated form of the dipole Hamiltonian \mathcal{H} and expanding the commutators, Redfield and Yu arrive at the following expressions for traces:

$$\frac{\text{Tr} \{[\mathcal{H}, I_{i_z}][\mathcal{H}, I_{j_z}]\}}{\text{Tr} \{I_{i_z}^2\}} = \frac{2}{3} I(I+1) A_{ij}^2 \quad (\text{A.4})$$

$$\begin{aligned}
\frac{\text{Tr} \{ [\mathcal{H}, [\mathcal{H}, I_{i_z}]] [\mathcal{H}, [\mathcal{H}, I_{j_z}]] \}}{\text{Tr} \{ I_{i_z}^2 \}} &= \\
\frac{2}{9} I^2 (I+1)^2 \sum_k &\left\{ \begin{aligned} &3A_{ik}^2 A_{jk}^2 - A_{ij}^2 [4A_{ik}^2 + 4A_{jk}^2 + (B_{ik} - B_{jk})^2] \\ &+ 2A_{ik} A_{jk} A_{ij} (B_{ik} + B_{jk} - 2B_{ij}) \end{aligned} \right\} \\
-\frac{1}{15} I(I+1) &\left\{ 10A_{ij}^4 + (2I-1)(2I+3)A_{ij}^2 (B_{ij}^2 + 4A_{ij}^2) \right\}
\end{aligned} \tag{A.5}$$

where $\hbar^{-1}A_{ij} = (3 \cos^2 \theta_{ij} - 1)/2r_{ij}^3$ and $B_{ij} = -2A_{ij}$ for the dipolar interaction. The sums are evaluated for $i \neq j \neq k$. An adjustment of notation from Redfield and Yu's symbol A_{ij} to the symbol $b_{ij} = \hbar^{-1}A_{ij}$ of Chapters 6 and 7, and some further rearrangement, yields

$$\begin{aligned}
\frac{\text{Tr} \{ [\mathcal{H}, I_{i_z}] [\mathcal{H}, I_{j_z}] \}}{\text{Tr} \{ I_{i_z}^2 \}} &= \frac{2}{3} \hbar^2 I(I+1) b_{ij}^2 \tag{A.6} \\
\frac{\text{Tr} \{ [\mathcal{H}, [\mathcal{H}, I_{i_z}]] [\mathcal{H}, [\mathcal{H}, I_{j_z}]] \}}{\text{Tr} \{ I_{i_z}^2 \}} &= \\
\hbar^4 I^2 (I+1)^2 &\left[\begin{aligned} &\frac{2}{9} \sum_k \left\{ \begin{aligned} &3b_{ik}^2 b_{jk}^2 - 8b_{ij}^2 (b_{ik} - b_{jk})^2 \\ &- 4b_{ij} b_{ik} b_{jk} (b_{ik} + b_{jk}) \end{aligned} \right\} \\ &- b_{ij}^4 \left\{ \frac{32}{15} - \frac{14}{15I(I+1)} \right\} \end{aligned} \right]
\end{aligned} \tag{A.7}$$

Defining

$$A_{ij} \equiv \frac{1}{3} b_{ij}^2 \tag{A.8}$$

$$B_{ijk} \equiv \frac{2}{9} \left\{ 3b_{ik}^2 b_{jk}^2 - 8b_{ij}^2 (b_{ik} - b_{jk})^2 - 4b_{ij} b_{ik} b_{jk} (b_{ik} + b_{jk}) \right\} \tag{A.9}$$

$$C_{ij} \equiv -b_{ij}^4 \left\{ \frac{32}{15} - \frac{14}{15I(I+1)} \right\} \tag{A.10}$$

and combining equations (A.1) through (A.10), we can express D_Z directly in terms of lattice sums:

$$D_Z = [I(I+1)]^{\frac{1}{2}} \left(\frac{-\frac{\pi}{2} (\sum_j' x_{ij}^2 \mathcal{A}_{ij})^3}{\sum_j \sum_k x_{ij}^2 \mathcal{B}_{ijk} + \sum_j x_{ij}^2 \mathcal{C}_{ij}} \right)^{\frac{1}{2}} \quad ijk \neq \quad (\text{A.11})$$

For the evaluation of spin quantum number dependence in Section 8.1, this expression will suffice. Apart from the trivial factor of $[I(I+1)]^{\frac{1}{2}}$, all of the dependence of D_Z upon I enters in the second term of \mathcal{C}_{ij} . It is this dependence which is plotted in Figure 8-1, with the initial factor representing the magnitude of spin angular momentum scaled to unity in accordance with the reduced unit system used throughout this work (cf. Section 6.4).

The study of concentration dependence undertaken in Section 7.2 requires several additional steps. First, we must evaluate the classical limit of (A.11). To do this, we return to the trace expressions (A.6) - (A.7) and let $I \rightarrow \infty$ and $\hbar \rightarrow 0$ such that $\hbar^2 I(I+1) = 1$ in reduced units.¹ The effect of this operation is to remove the prefactor $[I(I+1)]^{\frac{1}{2}}$ and to truncate \mathcal{C}_{ij} to $-\frac{32}{15}b_{ij}^4$. Moving to a lattice with concentration c and specifying that the sums are to be taken only over occupied sites, we arrive at Equations (7.9) and (7.10), reproduced here:

$$D_Z = \left(\frac{-\frac{\pi}{2} (\sum_j' x_{ij}^2 \mathcal{A}_{ij})^3}{\sum_j \sum_k' x_{ij}^2 \mathcal{B}_{ijk} + \sum_j x_{ij}^2 \mathcal{C}_{ij}} \right)^{\frac{1}{2}} \quad ijk \neq \quad (\text{A.12})$$

$$\begin{aligned} \mathcal{A}_{ij} &= \frac{1}{3} \left(\frac{3 \cos^2 \theta_{ij} - 1}{2} \right)^2 = \frac{1}{3} b(\mathbf{r}_{ij})^2 \equiv \frac{1}{3} b_{ij}^2 \\ \mathcal{B}_{ijk} &= \frac{2}{9} \left\{ 8b_{ij}^2 (b_{ik} - b_{jk})^2 + 4b_{ij} b_{ik} b_{jk} (b_{ik} + b_{jk}) - 3b_{ik}^2 b_{jk}^2 \right\} \\ \mathcal{C}_{ij} &= -\frac{32}{15} b_{ij}^4 \end{aligned} \quad (\text{A.13})$$

¹For unit γ , this corresponds to setting the magnetic moment of a single spin to 1, by our reduced-unit prescription.

Arguing, as we did in the text of Section 7.2, that the primed sums may be replaced by appropriately scaled sums over the full lattice (cf. Ref. [44]),

$$\begin{aligned}\sum'_j &\rightarrow c \sum_j \\ \sum'_j \sum'_k &\rightarrow c^2 \sum_j \sum_k\end{aligned}\tag{A.14}$$

we write

$$\begin{aligned}D_Z &\rightarrow \left(\frac{\frac{\pi}{2} (c \sum_j x_{ij}^2 \mathcal{A}_{ij})^3}{c^2 \sum_j \sum_k x_{ij}^2 \mathcal{B}_{ijk} + c \sum_j x_{ij}^2 \mathcal{C}_{ij}} \right)^{\frac{1}{2}} && ijk \neq \\ &\equiv \frac{\chi_1 c}{(1 + \chi_2 c)^{\frac{1}{2}}}\end{aligned}\tag{A.15}$$

defining

$$\begin{aligned}\chi_1 &= \left(\frac{\frac{\pi}{2} (\sum_j x_{ij}^2 \mathcal{A}_{ij})^3}{-\sum_j x_{ij}^2 \mathcal{C}_{ij}} \right)^{\frac{1}{2}} \\ \chi_2 &= \frac{\sum_j \sum_k x_{ij}^2 \mathcal{B}_{ijk}}{\sum_j x_{ij}^2 \mathcal{C}_{ij}} && ijk \neq\end{aligned}\tag{A.16}$$

It is convenient to replace the restricted sums ($ijk \neq$) in χ_1 and χ_2 with unrestricted sums, so that convolutions may be used in their evaluation. Since $x_{ii} = b_{ii} = 0$, χ_1 is unaffected by this change, but extra terms do appear in χ_2 which must later be subtracted. The final results are

$$\begin{aligned}\chi_1 &= \frac{1}{24} \left(\frac{5\pi (\sum_j x_{ij}^2 b_{ij})^3}{\sum_j x_{ij}^2 b_{ij}^4} \right)^{\frac{1}{2}} \\ \chi_2 &= \frac{\frac{5}{48} \sum_j \sum_k x_{ij}^2 [8b_{ij}^2 (b_{ik}^2 - b_{jk}^2) + 4b_{ij} b_{ik} b_{jk} (b_{ik} + b_{jk}) - 3b_{ik}^2 b_{jk}^2]}{\sum_j x_{ij}^2 b_{ij}^4} - \frac{5}{3}\end{aligned}\tag{A.17}$$

As in the case of local field calculations, three-dimensional convolutions facilitate

the numerical determination of χ_2 . The following convolution relations are of use in evaluating terms involving less than three coupling arrays b :

$$\begin{aligned}
\sum_j \sum_k x_{ij}^2 b_{ij}^2 b_{ik}^2 &= \sum_j \sum_k x_{ji}^2 b_{ji}^2 b_{ik}^2 = \sum_j \sum_k (x^2 b^2)_{ji} b_{ik}^2 \\
&= \sum_j \left((x^2 b^2) \otimes b^2 \right)_{ji} = \sum_j \left((x^2 b^2) \otimes b^2 \right)_{ij} \\
\sum_j \sum_k x_{ij}^2 b_{ij}^2 b_{jk}^2 &= \sum_j \left((x^2 b^2) \otimes b^2 \right)_{ij} \\
\sum_j \sum_k x_{ij}^2 b_{ik}^2 b_{jk}^2 &= \sum_j \sum_k (x_{ik} + x_{kj})^2 b_{ik}^2 b_{jk}^2 \\
&= \sum_j \sum_k x_{ik}^2 b_{ik}^2 b_{jk}^2 + 2x_{ik} b_{ik}^2 x_{kj} b_{kj}^2 + b_{ik}^2 x_{kj}^2 b_{kj}^2 \\
&= \sum_k \left((x^2 b^2) \otimes b^2 \right)_{ik} + 2 \left((x b^2) \otimes (x b^2) \right)_{ik} + \left(b^2 \otimes (x^2 b^2) \right)_{ik}
\end{aligned} \tag{A.18}$$

The remaining terms in χ_2 are not amenable to simple convolutions, but numerical evaluation and physical intuition suggest that their contribution to the total value is small. The procedure which computes lattice coordinates and coupling matrices for the classical simulations of spin diffusion (cf. Appendix B) may be used to generate arrays x_{ij} and b_{ij} .

Appendix B

Matlab Code

The code to follow is designed for Matlab Version 4.0. With some slight modifications (involving primarily the treatment of global variables and three-dimensional plotting), it should also be backwards-compatible with Matlab 3.5. Though this appendix, in its bulk, may not at first appear to merit the epithet “parsimonious” applied in Chapter 6 to the Matlab programming environment, in fact quite a few simulation options and high-level plotting operations are included which would require more space still in conventional programming languages. The reduction in speed which generally results from using an interpreted language like Matlab (as opposed to a compiled language like Fortran) is in part allayed by Matlab’s practice of dynamically “compiling” functions at their first use. The advanced debugging and graphics capabilities of the Matlab environment provide substantial additional programming advantages.

B.1 Broadband Dipolar Recoupling

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %                                                                                               %
3 %               CC23PI.M                                                                                               %
4 %                                                                                               %
5 % Simulates fictitious 2-3 magnetization trajectory                                       %
6 % of a dipole-coupled homonuclear spin pair under MAS                                       %
7 % and a train of delta-function pi pulses.                                               %
8 %                                                                                               %
9 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

10 disp('Program cc23pi.m')
11 disp('')
12 disp(['If you have not entered input parameters ' ...
13       'and pulse sequence already, '])
14 disp(' type "<inputfilename>" and/or "<sequencefilename>" ')
15 disp(' followed by "return" (spelled out). ')

16 keyboard

17 %-----

18 t0=clock;

19 %initialize variables: call variable initialization m file
20 %               for cc2 simulation family
21 cc2_ini

22 %manipulate pi pulse sequence file parameters
23 nstep_pc=round(t_cycle_duration*nstep_r);
24 if npulse==0 | npcycle==0,
25     npulse_time=0;
26     n_first_cycle=0;
27 else
28     npulse_time=max([round(pulse_param(:,1))*nstep_r) ; ...
29                     ones(1,npulse)]);
30     n_first_cycle=max([round(t_first_cycle*nstep_r) ; 1]);
31 end
32 pulse_phase=pulse_param(:,2)*drfac;

33 nsigrot=nstep_r;
34 if rem(npulse,2)~=0,
```

```

35     nsigrot=nsigrot*2;
36     %         disp(['number of pulse cycles adjusted to ' ...
37     %         num2str(nsigrot)])
38 end

39 global nsigrot npulse npcycle nstep_pc npulse_time ...
40     n_first_cycle pulse_phase;

41 %initialize signal matrix
42 sig=zeros(nsig,3);

43 %-----

44 %begin calculations
45 disp('Don't rush me, I'm thinking...')

46 %initialize time and signal vectors
47 sig=zeros(nsig,3);
48 tindex_r=[2:nstep_r];
49 trot=tr*[1:nstep_r]/nstep_r;
50 nstep_s=max([round(dw*nstep_r/tr) 1]);
51 [npower,dummy]=idecompose(nsig*nstep_s,nstep_r);
52 global tindex_r trot nstep_s npower;

53 disp(['Averaging over ' num2str(ncr) ' powder angles.'])
54 disp(['     Alpha ranges from ' num2str(acr_min) ' to ' ...
55     num2str(acr_max)])
56 disp(['     Beta ranges from ' num2str(bcr_min) ' to ' ...
57     num2str(bcr_max)])
58 disp(['     Gamma ranges from ' num2str(gcr_min) ' to ' ...
59     num2str(gcr_max)])
60 [wi_l_f,ws_l_f,wis_l_f]=pwdr_comps(abg_min,abg_max,brl);
61 hold off; clg;

62 for icr=1:ncr;
63     if resflag,
64         sigpt=ccrsevolve2(wi_l_f(icr,:),ws_l_f(icr,:),...
65         wis_l_f(icr,:),wr);
66     else
67         sigpt=ccpievolve(wi_l_f(icr,:),ws_l_f(icr,:),...
68         wis_l_f(icr,:),wr);
69     end
70     sig=sig+real(sigpt);
71     if rem(icr,max([round(ncr/100) 1]))==0,
72         disp([' current angle index (fraction):      ' ...

```

```

73         num2str(icr) ' (' num2str(icr/ncr) ')']])
74     plot((0:nsig-1)*dw*1000,sig(:,3)/icr)
75     drawnow
76     ylabel('Avg. diff. z magnetization')
77     xlabel('Mix time (ms)')
78     end
79 end

80 %normalize signal
81 sig=sig/ncr;

82 disp('')
83 telapsed=etime(clock,t0);
84 disp(['Elapsed time ' num2str(telapsed) 'seconds'])
85 disp('')

86 %-----

87 %plot data
88 hold off;
89 plot((0:nsig-1)*dw*1000,sig(:,3))
90 ylabel('Avg. diff. z magnetization')
91 xlabel('Mix time (ms)')
92 disp('Strike any key to continue...')
93 pause

94 pl3d(sig(:,1),sig(:,2),sig(:,3))
95 title('Full 3D fictitious 2-3 magnetization trajectory')
96 grid

97 %-----

98 %option to save data to disk
99 disp('If you wish to save simulation data to a file, ')
100 disp('type ''save <filename> sig''.')
101 disp('')
102 disp('Thank you for flying the friendly skies...')

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %                                                                 %
3 %                   CC2_INI.M                                     %
4 %                                                                 %
5 % variable initialization and manipulation file %
6 % for cc2 simulation family                                     %
7 %                                                                 %

```

```

8  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9  %initialize variables
10 twopi=2.0*pi;
11 sqrtfac=sqrt(1.0/6.0);
12 drfac=pi/180;

13 %manipulate input file parameters
14 w_iso=[vi_iso vs_iso]*twopi;
15 w_aniso=[vi_aniso vs_aniso]*twopi;
16 abg=[a_pi_c b_pi_c g_pi_c
17      a_ps_c b_ps_c g_ps_c
18      a_pis_c b_pis_c g_pis_c]*drfac;
19 bis=vbis*twopi;
20 jis=vjis*twopi;
21 if vr~=0
22 tr=1.0/abs(vr);
23 end
24 wr=vr*twopi;
25 brl=dbrl*drfac;
26 abg_min=[acr_min bcr_min gcr_min]*drfac;
27 abg_max=[acr_max bcr_max gcr_max]*drfac;
28 if exist('fwhm')==0,
29 fwhm=0;
30 end
31 if exist('t2zq')==1,
32 if t2zq==0,
33 r2zq=0;
34 else
35 r2zq=1/t2zq;
36 end
37 else
38 r2zq=0;
39 end

40 d2_pi_c=wigner2(abg(1,:));
41 wip2=-eta_i*sqrtfac*w_aniso(1);
42 wi_p=[wip2 0 w_aniso(1) 0 wip2];
43 wi_c=wi_p*d2_pi_c;

44 d2_ps_c=wigner2(abg(2,:));
45 wsp2=-eta_s*sqrtfac*w_aniso(2);
46 ws_p=[wsp2 0 w_aniso(2) 0 wsp2];
47 ws_c=ws_p*d2_ps_c;

```

```

48 d2_pis_c=wigner2(abg(3,:));
49 wis_p=[0 0 -bis 0 0];
50 wis_c=wis_p*d2_pis_c;

51 global w_iso wi_c ws_c wis_c;
52 global tr nstep_r ncr nsig dw;
53 global bis jis fwhm t2zq r2zq;

1 function [wi_l_f,ws_l_f,wis_l_f]=pwdr_comps(...
2     abg_min,abg_max,brl,seedset)

3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 %
5 %                               PWDR_COMPS.M
6 % transforms to rotor frame and forms fourier components
7 % of I and S spin CSA and dipole coupling tensors
8 % input: angmin,angmax: 3-element vectors of transformation
9 %                               Euler angles (in radians), giving lower
10 %                              and upper bounds of powder average
11 %      brl: rotor inclination angle (in radians)
12 %      seedset: a flag indicating whether to set random
13 %              number generator seed to zero at beginning of
14 %              calculations.
15 %              If seedset=0, seed is not set, otherwise it is.
16 %              If no seedset argument is entered, the seed is
17 %              set by default.
18 % output: wi_l_f,ws_l_f,wis_l_f: ncr x 3 matrices of lab-frame
19 %                               I, S, and IS fourier components
20 %
21 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

22 global w_iso wi_c ws_c wis_c;
23 global tr nstep_r ncr nsig dw;
24 global bis jis fwhm t2zq r2zq;

25 %generate crystallite matrices
26 if nargin<4, seedset=1; end;
27 if seedset,
28     rand('seed',0);
29 end
30 delacr=abg_max(1)-abg_min(1);
31 cosbcrmin=cos(abg_min(2));
32 delcosbcr=cos(abg_max(2))-cosbcrmin;
33 delgcr=abg_max(3)-abg_min(3);
34 acr=abg_min(1)*ones(ncr,1) + delacr*rand(ncr,1);

```

```

35 cosbcr=cosbcrmin*ones(ncr,1) + delcosbcr*rand(ncr,1);
36 bcr=acos(cosbcr); %N.B. acos will return values from -pi to pi
37 gcr=abg_min(3)*ones(ncr,1) + delgcr*rand(ncr,1);

38 %transform to rotor frame
39 d2_c_r=wig2arr(acr,bcr,gcr);
40 r1=d2_c_r(:,1:5).'; r2=d2_c_r(:,6:10).'; r3=d2_c_r(:,11:15).';
41 r4=d2_c_r(:,16:20).'; r5=d2_c_r(:,21:25).';
42 wi_r=[ (wi_c*r1).' (wi_c*r2).' (wi_c*r3).' (wi_c*r4).' ...
43         (wi_c*r5).' ];
44 ws_r=[ (ws_c*r1).' (ws_c*r2).' (ws_c*r3).' (ws_c*r4).' ...
45         (ws_c*r5).' ];
46 wis_r=[ (wis_c*r1).' (wis_c*r2).' (wis_c*r3).' (wis_c*r4).' ...
47         (wis_c*r5).' ];

48 %form fourier components
49 rd2_r_l=redwig2(brl);
50 wi_l_f(:,5:-1:1)=[wi_r(:,1)*rd2_r_l(1,3) ...
51                  wi_r(:,2)*rd2_r_l(2,3) ...
52                  wi_r(:,3)*rd2_r_l(3,3) ...
53                  wi_r(:,4)*rd2_r_l(4,3) ...
54                  wi_r(:,5)*rd2_r_l(5,3)];
55 wi_l_f(:,3)=wi_l_f(:,3)+w_iso(1)*ones(ncr,1);
56 ws_l_f(:,5:-1:1)=[ws_r(:,1)*rd2_r_l(1,3) ...
57                  ws_r(:,2)*rd2_r_l(2,3) ...
58                  ws_r(:,3)*rd2_r_l(3,3) ...
59                  ws_r(:,4)*rd2_r_l(4,3) ...
60                  ws_r(:,5)*rd2_r_l(5,3)];
61 ws_l_f(:,3)=ws_l_f(:,3)+w_iso(2)*ones(ncr,1);
62 wis_l_f(:,5:-1:1)=[wis_r(:,1)*rd2_r_l(1,3) ...
63                   wis_r(:,2)*rd2_r_l(2,3) ...
64                   wis_r(:,3)*rd2_r_l(3,3) ...
65                   wis_r(:,4)*rd2_r_l(4,3) ...
66                   wis_r(:,5)*rd2_r_l(5,3)];

1 function sig=ccpievolve(wi,ws,wis,wr)

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %
4 %           CCPIEVOLVE.M
5 %
6 % calculates fictitious 2-3 magnetization trajectory for
7 % cc23pi.m for one set of powder angles
8 % input: wi: a 5-element row vector of I-spin fourier comp.'s
9 %         ws: a 5-element row vector of S-spin fourier comp.'s

```

```

10 %      wis: a 5-element row vector of fourier comp.'s for      %
11 %          I-S coupling                                       %
12 %      wr: the rotor angular frequency                         %
13 %          (all in radians/sec)                               %
14 % output: an nsig x 3 matrix of x,y,z components             %
15 %                                                             %
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

17 global w_iso wi_c ws_c wis_c;
18 global tr nstep_r ncr nsig dw;
19 global bis jis fwhm t2zq r2zq;
20 global nsigrot npulse npcycle nstep_pc npulse_time ...
21     n_first_cycle pulse_phase;
22 global tindex_r trot nstep_s npower;

23 %initialize variables
24 sig=zeros(nsig,3);
25 wbis=-wis; wbis(3)=wbis(3)+jis;
26 wdel=wi-ws;
27 if nstep_s>2,
28     l=eye(3); lr=1; lsigpower=1; lsigmod=1;
29     larr=ones(nstep_r,1)*[1 0 0 0 1 0 0 0 1];
30     lrpower=ones(npower,1)*[1 0 0 0 1 0 0 0 1];
31 else
32     dlarr=ones(nstep_r,1)*[1 0 0 0 1 0 0 0 1];
33 end

34 %rotor period evolution loop
35 %t0=clock;
36 for istep_r=tindex_r;
37     wd=four_to_fun(wdel,wr,trot(istep_r));
38     wb=four_to_fun(wbis,wr,trot(istep_r));

39     a=[-r2zq  -wd      0.0
40         wd      -r2zq  -wb
41         0.0    wb      0.0 ];

42     dl=expm(a*tr/nstep_r);      %include error handling?
43                                 %expm is fastest of built-in
44                                 %matlab matrix exponential
45                                 %routines
46     if nstep_s<=2,
47         dlarr(istep_r,:)=dl(:).';
48     else
49         l(:)=larr(istep_r-1,:);

```

```

50     l=d1*l;
51     larr(istep_r,:)=l(:).';
52     end
53 end
54 %telapsed=etime(clock,t0);
55 %disp(['rotor loop time ' num2str(telapsed) 'seconds'])

56 if nstep_s>2,
57     %calculate necessary powers rotor-period propagator
58     for ipower=2:(npower+1);
59         lr=l*lr;
60         lrpower(ipower,:)=lr(:).';
61     end
62 end

63 %signal loop
64 %t0=clock;
65 icycle=1; ipulse=1;
66 if nstep_s<=2,
67     lsig=eye(3); dlsig=lsig;
68     for isig=1:nsig;
69         sig(isig,:)=lsig(:,3).';
70         for istep=0:(nstep_s-1);
71             istep_s=(isig-1)*nstep_s+istep;
72             [ncycle_rs,istep_rs]=idecompose(istep_s,nstep_r);
73             dlsig(:)=dlarr(istep_rs+1,:);
74             lsig=dlsig*lsig;
75             ntimetest=n_first_cycle+ ...
76                 npulse_time(ipulse)+ ...
77                 (icycle-1)*nstep_pc;
78             if max([istep_s;1])==ntimetest,
79                 lsig=[1 0 0; 0 -1 0; 0 0 -1]*lsig;
80                 if ipulse==npulse,
81                     ipulse=0;
82                     icycle=min([icycle+1;npcycle]);
83                 end
84                 ipulse=max([min([ipulse+1;npulse]) ; 1]);
85             end
86         end
87     end
88 else
89     lprev=eye(3);
90     for isig=1:nsig;
91         istep_s=(isig-1)*nstep_s;
92         [ncycle_rs,istep_rs]=idecompose(istep_s,nstep_r);

```

```

93     lsigpower(:)=lrpower(ncycle_rs+1,:);
94     lsigmod(:)=larr(istep_rs+1,:);
95     lsig=lsigmod*lsigpower*lprev;
96     sig(isig,:)=lsig(:,3).';
97     for istep=0:(nstep_s-1);
98         istep_s=(isig-1)*nstep_s+istep;
99         [ncycle_rs,istep_rs]=idecompose(istep_s,nstep_r);
100        ntimetest=n_first_cycle+ ...
101            npulse_time(ipulse)+ ...
102            (icycle-1)*nstep_pc;
103        if max([istep_s;1])==ntimetest,
104            lsig=[1 0 0; 0 -1 0; 0 0 -1]*lsig;
105            lsigpower(:)=lrpower(ncycle_rs+1,:);
106            lsigmod(:)=larr(istep_rs+1,:);
107            lprev=inv(lsigmod*lsigpower)*lsig;
108            if ipulse==npulse,
109                ipulse=0;
110                icycle=min([icycle+1;npcycle]);
111            end
112            ipulse=max([min([ipulse+1;npulse]) ; 1]);
113        end
114    end
115 end
116 end
117 %telapsed=etime(clock,t0);
118 %disp(['signal loop time ' num2str(telapsed) 'seconds'])

```

```

1 function sig=ccrsevolve2(wi,ws,wis,wr)

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %
4 %             CCRSEVOLVE2.M
5 %
6 % calculates fictitious 2-3 magnetization trajectory for
7 % cc23pi.m (resflag=1) for one set of powder angles
8 %
9 % input: wi: a 5-element row vector of I-spin fourier comp.'s
10 %         ws: a 5-element row vector of S-spin fourier comp.'s
11 %         wis: a 5-element row vector of fourier comp.'s for
12 %             I-S coupling
13 %         wr: the rotor angular frequency
14 %             (all in radians/sec)
15 % output: an nsig x 3 matrix of x,y,z components
16 %
17 % calculates nearest-resonant modulated dipole field component%

```

```

18 % ignoring effect of pulses on modulating CSA difference      %
19 % (i.e. calculates wbres in CSA-difference interaction frame  %
20 % but not in pi pulse toggling frame)                        %
21 %                                                            %
22 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

23 global w_iso wi_c ws_c wis_c;
24 global tr nstep_r ncr nsig dw;
25 global bis jis fwhm t2zq r2zq;
26 global nsigrot npulse npcycle nstep_pc npulse_time ...
27     n_first_cycle pulse_phase;
28 global tindex_r trot nstep_s npower;

29 %initialize variables
30 sig=zeros(nsig,3);
31 wbis=-wis; wbis(3)=wbis(3)+jis;
32 wdel=wi-ws;
33 wdeliso=wdel(3);
34 wdel(3)=0;
35 wi(3)=0;
36 ws(3)=0;

37 %calculate nearest-resonant dipole component
38 for istep_r=tindex_r,
39     phidel=rotphase(wdel,wr,0,trot(istep_r));
40     wb=four_to_fun(wbis,wr,trot(istep_r));
41     wbmod(istep_r)=wb*exp(i*phidel);
42 end
43 polar(angle(wbmod),abs(wbmod))
44 pause(1)
45 wbmodspec=fftshift(fft(wbmod))/nstep_r;
46 sbnum=(1:nstep_r)-floor(nstep_r/2+1);
47 bar(sbnum,abs(wbmodspec))
48 title('Absolute-Value CSA-Modulated Dipole Spectrum')
49 xlabel('sideband number      *=nearest resonant component')
50 [dummy,ires]=min(abs(wdeliso-sbnum*wr));
51 wbres=wbmodspec(ires);
52 sbres=ires-floor(nstep_r/2+1);
53 hold on, plot(sbres,abs(wbres),'*'), hold off
54 disp(['Sideband number ' num2str(sbres) ' is nearest resonant'])
55 disp(['Absolute magnitude is ' num2str(abs(wbres)) 'rad/sec'])
56 pause(1)
57 wd=wdeliso-sbres*wr;

58 aplus=[-r2zq      -wd      0.0

```

```

59         wd         -r2zq         -abs(wbres)
60         0.0         abs(wbres)  0.0 ];
61 [vaplus,daplus]=eig(aplus);

62 aminus=[-r2zq     wd         0.0
63         -wd     -r2zq         -abs(wbres)
64         0.0     abs(wbres)  0.0 ];
65 [vaminus,daminus]=eig(aminus);

66 %signal loop
67 %t0=clock;
68 icycle=1; ipulse=1;
69 sig(1,:)= [0 0 1];
70 siglast=sig(1,:).';
71 tlast=0;
72 for isig=2:nsig;
73     istep_s=(isig-1)*nstep_s;
74     tsig=istep_s*tr/nstep_r;
75     if rem(ipulse-1+npulse*(icycle-1),2)==0,
76         siglast=(vaplus*diag(exp(diag(daplus)*...
77             (tsig-tlast)))/vaplus)*siglast;
78     else
79         siglast=(vaminus*diag(exp(diag(daminus)*...
80             (tsig-tlast)))/vaminus)*siglast;
81     end
82     sig(isig,:)=siglast.';
83     tlast=tsig;
84     for istep=0:(nstep_s-1);
85         istep_s=(isig-1)*nstep_s+istep;
86         tsig=istep_s*tr/nstep_r;
87         [ncycle_rs,istep_rs]=idecompose(istep_s,nstep_r);
88         ntimetest=n_first_cycle+ ...
89             npulse_time(ipulse)+ ...
90             (icycle-1)*nstep_pc;
91         if max([istep_s;1])==ntimetest,
92             %disp('pulse now...')
93             if rem(ipulse-1+npulse*(icycle-1),2)==0,
94                 siglast=(vaplus*diag(exp(diag(daplus)*...
95                     (tsig-tlast)))/vaplus)*siglast;
96             else
97                 siglast=(vaminus*diag(exp(diag(daminus)*...
98                     (tsig-tlast)))/vaminus)*siglast;
99             end
100            tlast=tsig;
101            if ipulse==npulse,

```

```

102         ipulse=0;
103         icycle=min([icycle+1;npcycle]);
104     end
105     ipulse=max([min([ipulse+1;npulse]) ; 1]);
106 end
107 end
108 end
109 %telapsed=etime(clock,t0);
110 %disp(['signal loop time ' num2str(telapsed) 'seconds'])

```

```

1 function [ncycle,istep]=idecompose(i,n)

```

```

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %
4 % IDECOMPOSE.M
5 %
6 % decomposes integers for cycle calculations.%
7 %
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

9 ncycle=fix(i/n);
10 istep=rem(i,n);

```

```

1 function pl3d(vx,vy,vz,linetype,gridflag,fac,mulvec)

```

```

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %
4 % PL3D.M
5 % plots 3D vector trajectories on a sphere
6 %
7 % update of plot3d2.m for Matlab4.0, using the new
8 % 3D plotting functionality.
9 % DKS 4/27/93
10 %
11 % input arguments:
12 % vx,vy,vz: column vectors of x,y,z coordinates.
13 % linetype: linetype string. Default is '-y'.
14 % gridflag: 0 --> plot only cartesian axes and
15 % great circles (default).
16 % 1 --> plot spherical grid.
17 % fac: a 2-element vector fac=[sc em] containing
18 % scaling factors for v
19 % sc: a constant scale factor for each trajectory.
20 % Default is sc=0. If sc=0 or the argument
21 % is omitted, the axes and great circles are

```

```

22 %           scaled to the maximum excursion of the           %
23 %           trajectory (after exponential multiplication). %
24 %           Otherwise, the axes and great circles are       %
25 %           scaled to the maximum of sc times the           %
26 %           trajectory (after exponential multiplication). %
27 %           If sc<0, the axes and great circles are scaled %
28 %           to the absolute magnitude of sc.                 %
29 %           em: an exponential multiplication factor for v.   %
30 %           em is the exponent of e by which v will be      %
31 %           attenuated at its last point. Default is em=0.  %
32 %           mulvec: an npt-element vector to be multiplied  %
33 %           pointwise by vx,vy,vz.                            %
34 %                                                            %
35 % The axes and viewpoint of the plot may be controlled      %
36 % with the Matlab4.0 commands AXIS and VIEW.                %
37 %                                                            %
38 % Features to appear later:                                  %
39 %     1) Option for different foreground & background        %
40 %     linetypes                                              %
41 %     2) Ability to plot multiple trajectories on the       %
42 %     same axes (currently, the best way to do this is      %
43 %     with the 'hold' command and repeat calls to pl3d) %
44 %                                                            %
45 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
46 drfac=pi/180;

47 %fill out argument list
48 npt=length(vx);
49 if nargin < 4,
50     linetype='-y';
51     gridflag=0;
52     fac=[0 0];
53     mulvec=ones(npt,1);
54 elseif nargin < 5,
55     gridflag=0;
56     fac=[0 0];
57     mulvec=ones(npt,1);
58 elseif nargin < 6,
59     fac=[0 0];
60     mulvec=ones(npt,1);
61 elseif nargin < 7,
62     mulvec=ones(npt,1);
63 end

```

```

64 %set scaling
65 sc=fac(1); em=fac(2);
66 if sc<=0,
67     mulfac=mulvec.*exp(-em*(0:(npt-1))'/(npt-1));
68 else
69     mulfac=sc*mulvec.*exp(-em*(0:(npt-1))'/(npt-1));
70 end
71 vxmul=vx.*mulfac;
72 vymul=vy.*mulfac;
73 vzmul=vz.*mulfac;
74 if sc<0,
75     maxsc=abs(sc);
76 else
77     maxsc=max(sqrt(vxmul.^2+vymul.^2+vzmul.^2));
78 end

79 %plot great circles
80 ang=2*pi*(0:180)'/180;
81 %xy
82 xc=maxsc*sin(ang); yc=maxsc*cos(ang); zc=maxsc*zeros(181,1);
83 plot3(xc,yc,zc,'w')
84 axis([-maxsc maxsc -maxsc maxsc -maxsc maxsc])
85 axis('image')
86 hold on
87 %xz
88 plot3(xc,zc,yc,'w')
89 %yz
90 plot3(zc,xc,yc,'w')
91 xlabel('x')
92 ylabel('y')
93 zlabel('z')

94 %plot spherical grid
95 if gridflag,
96     nphi=12;
97     for iphi=1:nphi,
98         phi=2*pi*(iphi-1)/nphi;
99         plot3(xc*cos(phi),xc*sin(phi),yc,':')
100     end
101     ntheta=6;
102     for itheta=1:ntheta,
103         theta=pi*(itheta-1)/ntheta;
104         plot3(yc*sin(theta),xc*sin(theta),...
105             maxsc*cos(theta)*ones(181,1),':')
106     end

```

```

107 end

108 %plot axes
109 plot3(maxsc*[-1 -1/180 1/180 1]',zeros(4,1),zeros(4,1),'w')
110 plot3(zeros(4,1), maxsc*[-1 -1/180 1/180 1]', zeros(4,1),'w')
111 plot3(zeros(4,1),zeros(4,1),maxsc*[-1 -1/180 1/180 1]', 'w')

112 %plot vector data
113 plot3(vxmul,vymul,vzmul,linetype)

114 hold off

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %
3  %          ZNAC_CC23.M
4  %
5  % Sample input file for cc23pi.m.
6  % (parameters for 13C2-ZnAc)
7  %
8  % All dimensioned parameters have units of
9  % Hz, sec, and degrees.
10 %
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

12 resflag=1;      % 1 --> perform calculations in the
13                 % rotating frame of the nearest-resonant
14                 % dipole component and the toggling frame
15                 % of a train of delta-function pi pulses
16                 % 0 --> perform full calculation

17 % I spin parameters
18 vi_iso=13085;   % isotropic chemical shift
19 vi_aniso=-6460; % chemical shift anisotropy
20 eta_i=0.34;    % CS asymmetry parameter
21 a_pi_c=0; b_pi_c=90; g_pi_c=0; % alpha, beta, and gamma
22                 % Euler angles for
23                 % principal-axis to
24                 % crystal-axis
25                 % transformation

26 % S spin paramters
27 vs_iso=0;      % isotropic chemical shift
28 vs_aniso=-1870; % chemical shift anisotropy
29 eta_s=0.17;   % CS asymmetry parameter
30 a_ps_c=0; b_ps_c=0; g_ps_c=0; % alpha, beta, gamma

```

```

31 % I-S coupling parameters
32 vbis=2200; % dipole coupling strength
33 a_pis_c=0; b_pis_c=0; g_pis_c=0;% alpha, beta, gamma
34 vjis=50; % J-coupling strength

35 % MAS and powder average parameters
36 vr=5000; % MAS rotor speed
37 dbrl=atan(sqrt(2.0))*180/pi; % rotor inclination angle
38 acr_min=0; acr_max=360; % min and max alpha for
39 % crystal- to rotor-frame
40 % transformation
41 bcr_min=0; bcr_max=180; % min and max beta
42 gcr_min=0; gcr_max=360; % min and max gamma
43 ncr=1; % number of crystallites
44 % in the powder average

45 % data acquisition parameters and calculation step size
46 nsig=100; % number of data points
47 nstep_r=100; % number of internal steps
48 % per rotor period
49 dw=1/10/vr; % time increment
50 % (should be a rational
51 % fraction of rotor
52 % period for
53 % proper synchronization)

54 % relaxation parameters
55 t2zq=0.006; % zero-quantum T2

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % %
3 % ONESEQ.M %
4 % %
5 % sample pulse sequence parameter file %
6 % (one pi pulse per rotor period) %
7 % %
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

9 npulse=1; %number of pi pulses in cycle
10 npcycle=100; %number of cycles
11 t_first_cycle=0; %time(rotor periods) to start 1st cycle
12 t_cycle_duration=1; %duration(rotor periods) of cycle
13 pulse_param=[0.5 0.0]; %matrix of pulse parameters
14 % time(rotor periods),phase(degrees)

```

B.2 Classical Spin Diffusion

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %                                                                                                                                              %
3  %                CMLATTICE.M                                                                                                                                              %
4  %                                                                                                                                              %
5  % Simulates the dynamics of a classical coupled spin lattice.                                                                                                                                              %
6  %                                                                                                                                              %
7  % DKS 4/94                                                                                                                                              %
8  %                                                                                                                                              %
9  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

10 disp('Program cmlattice.m')
11 disp('')

12 clear global

13 %input parameters
14 cmlinput;

15 %-----

16 t0=clock;

17 %-----

18 %begin classical calculations
19 disp('Calculating...')

20 %initialize dimension matrices and binomial coefficient matrix
21 dIk=[1 2 3; lsz msz nsz];
22 d3to4=[1 2 3 4; lsz msz nsz 1];
23 d4to3=[4 1 2 3; 1 lsz msz nsz];
24 btree=binomtree(nderivmax);

25 %multiply dw and divide w_iso and -wbis by an appropriate factor
26 %to avoid over/underflow.
27 wi=w_iso/scalefac;
28 wb=-wbis/scalefac;
29 dt=dw*scalefac;

30 %save global variables for use by other functions
31 global btree wi wb dt dIk d3to4 d4to3;
```

```

32 if ~jflag,
33     %form coupling arrays
34     if tflag,
35         [Cmat,Chat]=cmlChat;
36     else
37         [Cm11,Cm12,Cm13,Cm22,Cm23,Cm33,...
38             Ch11,Ch12,Ch13,Ch22,Ch23,Ch33]=cmlCh_full;
39     end
40 end

41 %generate initial lattice configuration
42 disp('generating initial lattice configuration...')
43 [Ixsig,Iysig,Izsig]=cmlinit;

44 %choose vacancy sites
45 cind=(rand(1,lsz*msz*nsz)<=cset);
46 cact=nnz(cind)/(lsz*msz*nsz);
47 disp(['target concentration = ' num2str(cset) ...
48     ', actual concentration = ' num2str(cact)])
49 Ixsig=Ixsig.*cind;
50 Iysig=Iysig.*cind;
51 Izsig=Izsig.*cind;

52 %calculate dynamical evolution
53 disp('calculating dynamical evolution...')
54 cmlevolve(Ixsig,Iysig,Izsig,cind);

55 %-----

56 disp('')
57 telapsed=etime(clock,t0);
58 disp(['Elapsed time ' num2str(telapsed) 'seconds'])

59 %-----

60 %process and plot data
61 procflag=1;
62 cmlproc

63 disp('')
64 disp('Thank you for flying the friendly skies...')

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 %
                                     CMLINPUT.M
                                     %

```

```

4 % %
5 % input parameters for cmlattice.m %
6 % %
7 % DKS 4/94 %
8 % %
9 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

10 %spin system parameters

11 %lattice size
12 lsz=16; msz=16; nsz=16; % x,y,z dimension

13 %lattice shape
14 lunit=1; munit=1; nunit=1; % x,y,z unit cell length
15 alpha=90; beta=90; gamma=90; % unit cell angles

16 %interaction region size
17 isz=lsz; jsz=msz; ksz=nsz; % x,y,z size of interaction region
18 protectaxes=[]; % for simultaneous multiple runs
19 rfact=eps; % tolerance in spin position

20 %lattice orientation (with respect to B0)
21 %qtheta=90; % [100] orientation option
22 %qphi=0;
23 %ostring=' [100] ';

24 %qtheta=90; % [010] orientation option
25 %qphi=90;
26 %ostring=' [010] ';

27 %qtheta=0; % [001] orientation option
28 %qphi=0;
29 %ostring=' [001] ';

30 qtheta=54.7; % [111] orientation option
31 qphi=45;
32 ostring=' [111] ';

33 %qtheta=45; % [011] orientation option
34 %qphi=0;
35 %ostring=' [011] ';

36 %concentration
37 cset=1; % concentration set point

```

```

38 %interaction strengths
39 twopi=2*pi;
40 vbis=1/twopi;           % dipole coupling strength
41 v_iso=0;               % isotropic chemical shift
42 %v_iso=zeros(1,lsz*msz*nsz); % can be set to an array:
43                       %   different shifts for each spin
44 wbis=twopi*vbis;       % conversions to rad/sec
45 w_iso=twopi*v_iso;     %   "   "   "

46 %-----

47 %observation/calculation parameters

48 %measurement parameters
49 dw=0.2;                % time increment
50 nsig=100;              % number of time steps

51 %flags
52 pflag=1; dflag=0;      % plotting and debugging options
53 aterm=1; bterm=1;     % switches for
54                       %   longitudinal (aterm) and
55                       %   transverse (bterm) parts of
56                       %   the dipolar Hamiltonian
57 corrflag=0;           % switch for Eint fixing using
58                       %   artificial correlations
59 setflag=0;            % switch for pre-set c0
60 jflag=0;              % Heisenberg exchange coupling
61 tflag=1;              % dipole-dipole coupling:
62                       %   1 --> high-field truncated
63                       %   0 --> full hamiltonian
64 diffflag=0;          % type of diffusion to calculate
65                       %   1 --> interspin energy
66                       %   2 --> magnetization
67 convflag=1; swapflag=0; % convolution/FFT parameters

68 %iteration numbers
69 nderivmax=10;         % max. # of deriv.'s in Taylor series
70 nderivmin=4;         % min. " " " " " "
71 nnorm=10;            % normalize magnetizations every
72                       %   nnorm timesteps

73 %errors and tolerances
74 err0=1e-3; terr=1e100; % error parameters
75 scalefac=2^5;        % scale factor to avoid
76                       %   over/underflow

```

```

77 %energies and effective temperatures
78 enint=1e-3;           % target interspin energy
79 c0=1;                % correlation parameter
80 kdir=1;              % direction of disturbance wavevector
81 lambda=8;            % wavelength of disturbance
82 lambdax=lambda;     % extra wavelength for Eint fixing
83 Mamp=0.5;           % amplitude of disturbance

84 %-----

85 %global variable declarations
86 % (several statements to stay under max line length)
87 global lsz msz nsz lunit munit nunit ...
88     alpha beta gamma isz jsz ksz ...
89     rfact qtheta qphi ostring ...
90     cset ...
91     twopi vbis wbis v_iso w_iso ...
92     dw nsig;
93 global pflag dflag aterm bterm corrflag setflag ...
94     jflag tflag diffflag convflag swapflag;
95 global nderivmax nderivmin nnorm ...
96     err0 terr scalefac ...
97     enint c0 ...
98     lambda lambdax kdir Mamp;
99 global protectaxes;

100 %-----

101 %quantities to calculate

102 cmoplist=[
103     ['Ix' blanks(63-2)  ]
104     ['Iy' blanks(63-2)  ]
105     ['Iz' blanks(63-2)  ]
106     ['Ixdot' blanks(63-5)]
107     ['Iydot' blanks(63-5)]
108     ['Izdot' blanks(63-5)]
109     ['Bx' blanks(63-2)  ]
110     ['By' blanks(63-2)  ]
111     ['Bz' blanks(63-2)  ]
112     ['Bxdot' blanks(63-5)]
113     ['Bydot' blanks(63-5)]
114     ['Bzdot' blanks(63-5)]
115     ['(Ix.*Bx+Iy.*By+Iz.*Bz)/2' blanks(63-24)]

```

```

116      ['(Ixdot.*Bx+Iydot.*By+Izdot.*Bz+Ix.*Bxdot+'...
117          'Iy.*Bydot+Iz.*Bzdot)/2']
118      ['sqrt(Ix.^2+Iy.^2+Iz.^2)' blanks(63-23)]
119      ];

120  testlist=[
121      ['Ix(:,1)' blanks(123-7)]
122      ['Iy(:,1)' blanks(123-7)]
123      ['Iz(:,1)' blanks(123-7)]
124      ['Ixdot(:,1)' blanks(123-10)]
125      ['Iydot(:,1)' blanks(123-10)]
126      ['Izdot(:,1)' blanks(123-10)]
127      ['Bx(:,1)' blanks(123-7)]
128      ['By(:,1)' blanks(123-7)]
129      ['Bz(:,1)' blanks(123-7)]
130      ['Bxdot(:,1)' blanks(123-10)]
131      ['Bydot(:,1)' blanks(123-10)]
132      ['Bzdot(:,1)' blanks(123-10)]
133      ['(Ix(:,1)*Bx(:,1)+Iy(:,1)*By(:,1)+'...
134          'Iz(:,1)*Bz(:,1))/2' blanks(123-51)]
135      ['(Ixdot(:,1).*Bx(:,1)+Iydot(:,1).*By(:,1)+'...
136          'Izdot(:,1).*Bz(:,1)+Ix(:,1).*Bxdot(:,1)+'...
137          'Iy(:,1).*Bydot(:,1)+Iz(:,1).*Bzdot(:,1))/2']
138      ['sqrt(Ix(:,1)^2+Iy(:,1)^2+Iz(:,1)^2)' blanks(123-35)]
139      ];

140  namelist=[
141      'Ix      '
142      'Iy      '
143      'Iz      '
144      'Ixdot   '
145      'Iydot   '
146      'Izdot   '
147      'Bx      '
148      'By      '
149      'Bz      '
150      'Bxdot   '
151      'Bydot   '
152      'Bzdot   '
153      'eint    '
154      'eintdot'
155      'Imag    '];

156  opnum=length(namelist(:,1));

```

```

157 global opnum cmoplist testlist namelist;
158 for iop=1:opnum,
159     if isempty(protectaxes),
160         eval(['global ' deblank(namelist(iop,:)) 'sum;']);
161     else
162         eval(['global ' deblank(namelist(iop,:)) 'sums;']);
163     end
164     eval(['global ' deblank(namelist(iop,:)) 'mean;']);
165     eval(['global ' deblank(namelist(iop,:)) 'test;']);
166 end

```

```

1 function B=binomtree(nmax)

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %                                                                 %
4 %             BINOMTREE.M                                         %
5 % calculates the tree of binomial coefficients                    %
6 % (Pascal's triangle) up to order nmax                          %
7 %                                                                 %
8 % input:  nmax = maximum order of the binomial expansion        %
9 %          to be calculated                                       %
10 % output: B = (nmax+1) by (nmax+1) matrix of binomial          %
11 %          coefficients, with order increasing top to bottom    %
12 %          and term number increasing from left to right       %
13 %                                                                 %
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

15 B = zeros(nmax+1);
16 B(1,1) = 1;
17 B(2,:) = [1 1 zeros(1,nmax-1)];

18 for n = 2:nmax,
19     B(n+1,:) = B(n,:) + [0 B(n,1:nmax)];
20 end

```

```

1 function [Cmat,Chat]=cmlChat

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %                                                                 %
4 %             CMLCHAT.M                                           %
5 % Construct high field truncated dipole coupling matrix and its %
6 % 3D FFT.                                                         %
7 %                                                                 %
8 % Global variables used:                                         %
9 %     lsz,msz,nsz: x,y,z dimensions of spin lattice             %

```

```

10 %      lunit,munit,nunit: unit cell x,y,z lengths      %
11 %      alpha,beta,gamma: unit cell angles              %
12 %      isz,jsz,ksz: x,y,z dimensions of interaction region %
13 %      qtheta,qphi: lattice orientation angles          %
14 %      (w.r.t applied B field)                        %
15 %                                                      %
16 % Output: Cmat: lsz x msz x nsz array of dipole couplings %
17 %      (A_FILES format) centered at (1,1,1)           %
18 %      Chat: 3D FFT of Cmat                            %
19 %                                                      %
20 % DKS 10/93                                           %
21 % DKS 10/30/93: modify indices to allow filling of even %
22 %      dimensional matrices                            %
23 % DKS 11/2/93: fix indices, save global coupling matrices %
24 % DKS 1/12/94: add option for non-cubic geometry      %
25 %                                                      %
26 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

27 %global variable declarations
28 % (several statements to stay under max line length)
29 global lsz msz nsz lunit munit nunit ...
30     alpha beta gamma isz jsz ksz ...
31     rfact qtheta qphi ostring ...
32     cset ...
33     twopi vbis wbis v_iso w_iso ...
34     dw nsig;
35 global pflag dflag aterm bterm corrflag setflag ...
36     jflag tflag diffflag convflag swapflag;
37 global nderivmax nderivmin nnorm ...
38     err0 terr scalefac ...
39     enint c0 ...
40     lambda lambdax kdir Mamp;
41 global protectaxes;

42 global Cmat Chat;

43 %initialize variables
44 [Cmat,dCmat]=a3_zeros([1 2 3; lsz msz nsz]);
45 %make accurate magic angle
46 if qtheta < 54.8 & qtheta > 54.6,
47     qtheta=atan(sqrt(2))*180/acos(-1);
48 end
49 %initialize constants
50 drfac=pi/180;
51 sinalpha=sin(alpha*drfac);

```

```

52 sinbeta=sin(beta*drfac);
53 singamma=sin(gamma*drfac);
54 if qtheta ~= 0,
55     qctheta=cos(qtheta*drfac);
56     qstheta=sin(qtheta*drfac);
57     qcphi=cos(qphi*drfac);
58     qsphi=sin(qphi*drfac);
59 end

60 %choice of center location in lattice
61 lcell=floor(lsz/2)+1;
62 mcell=floor(msz/2)+1;
63 ncell=floor(nsz/2)+1;
64 %dimensions of interaction shell
65 icell=floor((isz-1)/2);
66 jcell=floor((jsz-1)/2);
67 kcell=floor((ksz-1)/2);
68 iend=1-rem(isz,2);
69 jend=1-rem(jsz,2);
70 kend=1-rem(ksz,2);

71 %calculate Cmat
72 disp('making coupling matrix...')
73 nc=[-(kcell+kend); 1 ; kcell];
74 mc=[-(jcell+jend); 1 ; jcell];
75 lc=[-(icell+iend); 1 ; icell];
76 [C,dC]=a3_coord([lc mc nc]);
77 Xc=lunit*sinalpha*ano_arr(C,dC,1);
78 Yc=munit*sinbeta*singamma*ano_arr(C,dC,2);
79 Zc=nunit*ano_arr(C,dC,3);
80 R2c=Xc.*Xc+Yc.*Yc+Zc.*Zc;
81 Rc=sqrt(R2c);
82 [iind,jind]=find(abs(Rc)<rfact);
83 %^when rfact is not very small, only some neighbors considered.
84 %In most cases rfact=1e-32 and abs(Cmat)>rfact.
85 Rc(iind,jind)=Rc(iind,jind)+Inf;
86 Costhc=Zc./Rc;
87 if abs(qtheta)>eps,
88     Costhc=((Xc*qcphi+Yc*qsphi)*qstheta+Zc*qctheta)./Rc;
89     %^coordinate transform for lattices other than [001]
90 end

91 %form a coupling matrix "centered" at (lcell,mcell,ncell)
92 lind=lcell+(-(icell+iend):icell);
93 mnind=kron(ones(1,2*kcell+1+kend),mcell+(-(jcell+jend):jcell));

```

```

94  nmind=kron(ncell+(-(kcell+kend):kcell),ones(1,2*jcell+1+jend))-1;

95  Cmat(lind,nmind*msz+mnind)=(1-3*Costhc.*Costhc)./(Rc.^3)/2;

96  %rearrange indices so that coupling matrix is "centered"
97  %at (1,1,1) for correct convolution with spin vector array
98  lind=[lcell:lsz 1:(lcell-1)];
99  mnind=kron(ones(1,nsz),[mcell:msz 1:(mcell-1)]);
100 nmind=kron([ncell:nsz 1:(ncell-1)],ones(1,msz))-1;

101  Cmat=Cmat(lind,nmind*msz+mnind);

102  %make Chat: 3d fft of coupling matrix for calculating local field
103  disp('3D FFT of coupling matrix...')
104  if swapflag,
105      %octant-swapped 3D FFT (zeroth lag at center of spectrum)
106      [Chat,dChat]=a3_fftsxyz(Cmat,dCmat);
107  else
108      %unswapped 3D FFT (only to be multiplied and inverted)
109      [Chat,dChat]=a3_fftxyz(Cmat,dCmat);
110  end

1  function [Cm11,Cm12,Cm13,Cm22,Cm23,Cm33,...
2          Ch11,Ch12,Ch13,Ch22,Ch23,Ch33]=cmlCh_full

3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  %          cmlCh_full.m          %
5  %          %          %          %
6  % Construct full (untruncated) dipole coupling tensor elements %
7  % and their 3D FFT's          %
8  %          %          %          %
9  % Global variables used:      %
10 %      lsz,msz,nsz: x,y,z dimensions of spin lattice          %
11 %      lunit,munit,nunit: unit cell x,y,z lengths            %
12 %      alpha,beta,gamma: unit cell angles                    %
13 %      isz,jsz,ksz: x,y,z dimensions of interaction region   %
14 %      qtheta,qphi: lattice orientation angles                %
15 %          (w.r.t defined z axis)                             %
16 %          %          %          %
17 % Output: Ch##: lsz x msz x nsz arrays of dipole coupling   %
18 %          tensor elements centered at (1,1,1)                %
19 %      Ch11: FFT(3*Rx*Rx-1)                                    %
20 %      Ch12: FFT(3*Rx*Ry)                                    %
21 %      Ch13: FFT(3*Rx*Rz)                                    %
22 %      Ch22: FFT(3*Ry*Ry-1)                                    %

```

```

23 %          Ch23: FFT(3*Ry*Rz)                                %
24 %          Ch33: FFT(3*Rz*Rz-1)                              %
25 %                                                         %
26 % DKS 10/93                                                %
27 % DKS 10/30/93: modify indices to allow filling of         %
28 %                even dimensional matrices                 %
29 % DKS 11/2/93: fix indices, save global coupling matrices %
30 % DKS 1/12/94: add option for non-cubic geometry          %
31 %                                                         %
32 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

33 %global variable declarations
34 % (several statements to stay under max line length)
35 global lsz msz nsz lunit munit nunit ...
36     alpha beta gamma isz jsz ksz ...
37     rfact qtheta qphi ostring ...
38     cset ...
39     twopi vbis wbis v_iso w_iso ...
40     dw nsig;
41 global pflag dflag aterm bterm corrflag setflag ...
42     jflag tflag diffflag convflag swapflag;
43 global nderivmax nderivmin nnorm ...
44     err0 terr scalefac ...
45     enint c0 ...
46     lambda lambdax kdir Mamp;
47 global protectaxes;

48 global Cm11 Cm12 Cm13 Cm22 Cm23 Cm33 ...
49     Ch11 Ch12 Ch13 Ch22 Ch23 Ch33;

50 %initialize variables
51 [Cm11,dCm11]=a3_zeros([1 2 3; lsz msz nsz]);
52 Cm12=Cm11; dCm12=dCm11;
53 Cm13=Cm11; dCm13=dCm11;
54 Cm22=Cm11; dCm22=dCm11;
55 Cm23=Cm11; dCm23=dCm11;
56 Cm33=Cm11; dCm33=dCm11;

57 [Ch11,dCh11]=a3_zeros([1 2 3; lsz msz nsz]);
58 Ch12=Ch11; dCh12=dCh11;
59 Ch13=Ch11; dCh13=dCh11;
60 Ch22=Ch11; dCh22=dCh11;
61 Ch23=Ch11; dCh23=dCh11;
62 Ch33=Ch11; dCh33=dCh11;

```

```

63 %make accurate magic angle
64 if qtheta < 54.8 & qtheta > 54.6,
65     qtheta=atan(sqrt(2))*180/acos(-1);
66 end
67 %initialize constants
68 drfac=pi/180;
69 sinalpha=sin(alpha*drfac);
70 sinbeta=sin(beta*drfac);
71 singamma=sin(gamma*drfac);
72 if qtheta ~= 0,
73     qctheta=cos(qtheta*drfac);
74     qstheta=sin(qtheta*drfac);
75     qcphi=cos(qphi*drfac);
76     qsphi=sin(qphi*drfac);
77 end

78 %choice of center location in lattice
79 lcell=floor(lsz/2)+1;
80 mcell=floor(msz/2)+1;
81 ncell=floor(nsz/2)+1;
82 %dimensions of interaction shell
83 icell=floor((isz-1)/2);
84 jcell=floor((jsz-1)/2);
85 kcell=floor((ksz-1)/2);
86 iend=1-rem(isz,2);
87 jend=1-rem(jsz,2);
88 kend=1-rem(ksz,2);

89 %calculate coupling matrices
90 disp('making coupling matrix...')
91 nc=[-(kcell+kend); 1 ; kcell];
92 mc=[-(jcell+jend); 1 ; jcell];
93 lc=[-(icell+iend); 1 ; icell];
94 [C,dC]=a3_coord([lc mc nc]);
95 Xc=lunit*sinalpha*ano_arr(C,dC,1);
96 Yc=munit*sinbeta*singamma*ano_arr(C,dC,2);
97 Zc=nunit*ano_arr(C,dC,3);
98 R2c=Xc.*Xc+Yc.*Yc+Zc.*Zc;
99 Rc=sqrt(R2c);
100 [iind,jind]=find(abs(Rc)<rfact);
101 %^when rfact is not very small, only some neighbors considered.
102 %In most cases rfact=1e-16 and abs(Cmat)>rfact.
103 Rc(iind,jind)=Rc(iind,jind)+Inf;
104 R3c=Rc.^3;
105 Xu=Xc./Rc;

```

```

106 Yu=Yc./Rc;
107 Zu=Zc./Rc;
108 if abs(qtheta)>eps,
109     %coordinate transform for lattices other than [001]
110     Xu=((Xc*qcphi+Yc*qsphi)*qctheta-Zc*qstheta)./Rc;
111     Yu=(-Xc*qsphi+Yc*qcphi)./Rc;
112     Zu=((Xc*qcphi+Yc*qsphi)*qstheta+Zc*qctheta)./Rc;
113 end

114 %form coupling matrices "centered" at (lcell,mcell,ncell)
115 lind=lcell+(-(icell+iend):icell);
116 mnind=kron(ones(1,2*kcell+1+kend),mcell+(-(jcell+jend):jcell));
117 nmind=kron(ncell+(-(kcell+kend):kcell),ones(1,2*jcell+1+jend))-1;

118 Cm11(lind,nmind*msz+mnind)=(3*Xu.*Xu-1)./R3c;
119 Cm12(lind,nmind*msz+mnind)=(3*Xu.*Yu )./R3c;
120 Cm13(lind,nmind*msz+mnind)=(3*Xu.*Zu )./R3c;
121 Cm22(lind,nmind*msz+mnind)=(3*Yu.*Yu-1)./R3c;
122 Cm23(lind,nmind*msz+mnind)=(3*Yu.*Zu )./R3c;
123 Cm33(lind,nmind*msz+mnind)=(3*Zu.*Zu-1)./R3c;

124 %rearrange indices so that coupling matrix is "centered"
125 %at (1,1,1) for correct convolution with spin vector array
126 lind=[lcell:lsz 1:(lcell-1)];
127 mnind=kron(ones(1,nsz),[mcell:msz 1:(mcell-1)]);
128 nmind=kron([ncell:nsz 1:(ncell-1)],ones(1,msz))-1;

129 Cm11=Cm11(lind,nmind*msz+mnind);
130 Cm12=Cm12(lind,nmind*msz+mnind);
131 Cm13=Cm13(lind,nmind*msz+mnind);
132 Cm22=Cm22(lind,nmind*msz+mnind);
133 Cm23=Cm23(lind,nmind*msz+mnind);
134 Cm33=Cm33(lind,nmind*msz+mnind);

135 %make 3D FFT's for calculating local field
136 disp('3D FFT of coupling matrices...')
137 if swapflag,
138     %octant-swapped 3D FFT (zeroth lag at center of spectrum)
139     [Ch11,dCh11]=a3_fftsxyz(Cm11,dCm11);
140     [Ch12,dCh12]=a3_fftsxyz(Cm12,dCm12);
141     [Ch13,dCh13]=a3_fftsxyz(Cm13,dCm13);
142     [Ch22,dCh22]=a3_fftsxyz(Cm22,dCm22);
143     [Ch23,dCh23]=a3_fftsxyz(Cm23,dCm23);
144     [Ch33,dCh33]=a3_fftsxyz(Cm33,dCm33);
145 else

```

```

146 %unswapped 3D FFT (only to be multiplied and inverted)
147 [Ch11,dCh11]=a3_fftxyz(Cm11,dCm11);
148 [Ch12,dCh12]=a3_fftxyz(Cm12,dCm12);
149 [Ch13,dCh13]=a3_fftxyz(Cm13,dCm13);
150 [Ch22,dCh22]=a3_fftxyz(Cm22,dCm22);
151 [Ch23,dCh23]=a3_fftxyz(Cm23,dCm23);
152 [Ch33,dCh33]=a3_fftxyz(Cm33,dCm33);
153 end

```

```

1 function [nx,ny,nz]=cmlinit

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %
4 %                               CMLINIT.M
5 %
6 % Sets up initial lattice configuration for cmlattice.m.
7 %
8 % Global variables used:
9 %     corrflag: 1 --> introduce interspin correlations
10 %              to simulate high temp.
11 %              0 --> use uncorrelated initial configuration
12 %     setflag:  1 --> use pre-set value of cfac (cfac=c0)
13 %              0 --> calculate cfac as needed in secant rule
14 %     c0: pre-set value of cfac
15 %     lsz,msz,nsz: integer x,y,z lattice sizes
16 %     kdir: direction of k vector for initial disturbance
17 %     Mamp: amplitude of disturbance
18 %     enint: target interspin energy
19 %     tflag:  1 --> use high field truncated hamiltonian
20 %              0 --> use zero field (untruncated) hamiltonian
21 %     Cmat,Chat: coupling matrix and its 3D fft for tflag=1
22 %     Cm11,Cm12,Cm13,Cm22,Cm23,Cm33: coupling tensor elements
23 %     Ch11,Ch12,Ch13,Ch22,Ch23,Ch33: 3D FFT's
24 %     d4to3: dimension matrix for array resizing
25 %     scalefac: scale factor for B field calculations
26 %
27 % Output:
28 %     nx,ny,nz: 1 x lsz x msz x nsz arrays of spin components
29 %
30 % DKS 10/93-11/93
31 %
32 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

33 %global variable declarations
34 % (several statements to stay under max line length)

```

```

35 global lsz msz nsz lunit munit nunit ...
36     alpha beta gamma isz jsz ksz ...
37     rfact qtheta qphi ostring ...
38     cset ...
39     twopi vbis wbis v_iso w_iso ...
40     dw nsig;
41 global pflag dflag aterm bterm corrflag setflag ...
42     jflag tflag diffflag convflag swapflag;
43 global nderivmax nderivmin nnorm ...
44     err0 terr scalefac ...
45     enint c0 ...
46     lambda lambdax kdir Mamp;
47 global protectaxes;

48 global dIk d3to4 d4to3 btree wi wb dt;

49 %access coupling arrays
50 if ~jflag,
51     if tflag,
52         global Cmat Chat;
53     else
54         global Cm11 Cm12 Cm13 Cm22 Cm23 Cm33 ...
55             Ch11 Ch12 Ch13 Ch22 Ch23 Ch33;
56     end
57 end

58 disp('cmlinit: setting up initial lattice configuration...')

59 nxyz=lsz*msz*nsz;

60 if diffflag, %measure diffusion of interspin energy
61     lamz=2;
62     lamx=2*lambdax;
63     if kdir > 3,
64         if kdir==111,
65             lamx=lamx*sqrt(3);
66         else
67             lamx=lamx*sqrt(2);
68         end
69     end
70 else
71     lamz=lambda;
72     if kdir > 3,
73         if kdir==111,
74             lamz=lamz*sqrt(3);

```

```

75     else
76         lamz=lamz*sqrt(2);
77     end
78 end
79 lamx=lamz;
80 end
81 kmag=twopi/lamz;
82 kmagx=twopi/lamx;

83 if kdir==1,
84     Mpol=Mamp*cos(kmag*(0:(lsz-1)))');
85     symfun=cos(kmagx*(0:(lsz-1)))');
86     M0= Mpol*ones(1,msz*nsz);
87     M0=an_ztox(M0,[1 2 3 4; lsz msz nsz 1]);
88 elseif kdir==2,
89     Mpol=Mamp*cos(kmag*(0:(msz-1)))');
90     symfun=cos(kmagx*(0:(msz-1)))');
91     M0= Mpol*ones(1,lsz*nsz);
92     M0=a3_ztox(M0,[2 3 1; msz nsz lsz]);
93     M0=an_ztox(M0,[1 2 3 4; lsz msz nsz 1]);
94 elseif kdir==3
95     Mpol=Mamp*cos(kmag*(0:(nsz-1)))');
96     symfun=cos(kmagx*(0:(nsz-1)))');
97     M0= Mpol*ones(1,lsz*msz);
98     M0=a3_ytox(M0,[3 1 2; nsz lsz msz]);
99     M0=an_ztox(M0,[1 2 3 4; lsz msz nsz 1]);
100 elseif kdir==110
101     Mpol=Mamp*cos(kmag*(0:(2*lsz-1)))');
102     symfun=cos(kmagx*(0:(2*lsz-1)))');
103     indperm = (0:lsz-1)'*ones(1,msz)+ones(lsz,1)*(0:msz-1)+1;
104     repind = (1:msz)'*ones(1,nsz);
105     M0=zeros(lsz,msz);
106     M0(:)=Mpol(indperm);
107     M0=M0(:,repind(:));
108     M0=an_ztox(M0,[1 2 3 4; lsz msz nsz 1]);
109 elseif kdir==101
110     Mpol=Mamp*cos(kmag*(0:(2*nsz-1)))');
111     symfun=cos(kmagx*(0:(2*nsz-1)))');
112     indperm = (0:nsz-1)'*ones(1,lsz)+ones(nsz,1)*(0:lsz-1)+1;
113     repind = (1:lsz)'*ones(1,msz);
114     M0=zeros(nsz,lsz);
115     M0(:)=Mpol(indperm);
116     M0=M0(:,repind(:));
117     M0=a3_ytox(M0,[3 1 2; nsz lsz msz]);
118     M0=an_ztox(M0,[1 2 3 4; lsz msz nsz 1]);

```

```

119 elseif kdir==011
120     Mpol=Mamp*cos(kmag*(0:(2*msz-1))');
121     symfun=cos(kmagx*(0:(2*msz-1))');
122     indperm = (0:msz-1)'*ones(1,nsz)+ones(msz,1)*(0:nsz-1)+1;
123     repind = (1:nsz)'*ones(1,lsz);
124     M0=zeros(msz,nsz);
125     M0(:)=Mpol(indperm);
126     M0=M0(:,repind(:));
127     M0=a3_ztox(M0,[2 3 1; msz nsz lsz]);
128     M0=an_ztox(M0,[1 2 3 4; lsz msz nsz 1]);
129 elseif kdir==111
130     lmnsz=max(max(lsz,msz),nsz);
131     Mpol=Mamp*cos(kmag*(0:(3*lmnsz-1))');
132     symfun=cos(kmagx*(0:(3*lmnsz-1))');
133     indpermxy = (0:lsz-1)'*ones(1,msz)+ones(lsz,1)*(0:msz-1);
134     indperm=kron(ones(1,nsz),indpermxy)+...
135         kron(0:nsz-1,ones(lsz,msz))+1;
136     M0=zeros(lsz,msz*nsz);
137     M0(:)=Mpol(indperm);
138     M0=reshape(M0,lsz,msz*nsz);
139     M0=an_ztox(M0,[1 2 3 4; lsz msz nsz 1]);
140 end

141 if corrflag, %use correlated nx,ny

142     if setflag, %use pre-set value of cfac (cfac=c0)

143         %z components
144         nz=sign(M0).*(2*rand(1,nxyz).^((1-abs(M0))./(1+abs(M0))))-1);
145         sintheta=sqrt(1-nz.*nz);

146         %random numbers
147         ranset=rand(1,nxyz);
148         signs=sign(2*rand(1,nxyz)-1);

149         %x,y components
150         [nx,ny]=cmlinitxy(c0,sintheta,ranset,signs,symfun);

151         disp(['c0 = ' num2str(c0)])

152     else %force interspin energy to enint (ignoring c0)

153         nattempt=1;
154         nattemptmax=10;
155         goflag=1;

```

```

156 while (nattempt<=nattemptmax) & goflag, %multiple attempt loop
157     goflag=0;

158     %z components
159     nz=sign(M0).*(2*rand(1,nxyz).^((1-abs(M0))./(1+abs(M0)))-1);
160     sintheta=sqrt(1-nz.*nz);

161     %random numbers
162     ranset=rand(1,nxyz);
163     signs=sign(2*rand(1,nxyz)-1);

164     %calculate initial average interspin energy
165     %(not crucial: may simply set eint0=0)
166     Mcorr=xcorr(Mpol,'biased');
167     if tflag,
168         Csum=swapud(xyzsums(an_ztox(Cmat,d3to4),d4to3,kdir));
169     else
170         if kdir==1,
171             Csum=swapud(xyzsums(an_ztox...
172                 (Cm11+Cm12+Cm13,d3to4),d4to3,kdir));
173         elseif kdir==2,
174             Csum=swapud(xyzsums(an_ztox...
175                 (Cm12+Cm22+Cm23,d3to4),d4to3,kdir));
176         else
177             Csum=swapud(xyzsums(an_ztox...
178                 (Cm13+Cm23+Cm33,d3to4),d4to3,kdir));
179         end
180     end
181     lrange=floor(lsz/2);
182     mrange=floor(msz/2);
183     nrange=floor(nsz/2);
184     iend=1-rem(lsz,2);
185     jend=1-rem(msz,2);
186     kend=1-rem(nsz,2);
187     if kdir==1,
188         eint0=Mcorr(lsz+(-lrange:(lrange-iend)))'*Csum;
189     elseif kdir==2,
190         eint0=Mcorr(msz+(-mrange:(mrange-jend)))'*Csum;
191     else
192         eint0=Mcorr(nsz+(-nrangle:(nrangle-kend)))'*Csum;
193     end

194     %force initial average interspin energy to value eint
195     %using a secant algorithm
196     %set initial x1,y1,cfac

```

```

197     x1=0;
198     y1=eint0-enint;
199     cfac=2*abs(y1);
200     disp(['enint=' num2str(enint) ' eint0=' num2str(eint0)])
201     loopcount=1;
202     maxloop=20;
203     while (abs(y1)>(abs(enint/100)+eps))&(loopcount<=maxloop),
204         %^secant loop
205         if abs(cfac)>1
206             cfac=sign(cfac);
207         end

208         [nx,ny]=cmlinityx(cfac,sintheta,ranset,signs,symfun);

209         [Bx,By,Bz]=Bc(nx,ny,nz);
210         eint=scalefac*mean(nx.*Bx+ny.*By+nz.*Bz)/2;

211         disp([' cfac=' num2str(cfac) ' eint=' num2str(eint)])

212         if cfac==x1,
213             disp(['**Cannot achieve desired interspin energy' ...
214                 ' with this polarization'])
215             goflag=1;
216             nattempt=nattempt+1;
217             y1=0;
218         else
219             x2=x1;
220             y2=y1;
221             x1=cfac;
222             y1=eint-enint;
223             cfac=x1-y1*(x1-x2)/(y1-y2);
224         end

225         loopcount=loopcount+1;

226     end %end secant-rule loop

227     if loopcount>maxloop,
228         disp(['**Maximum loop count reached.'])
229         disp(['**Cannot achieve desired interspin energy.'])
230         goflag=1;
231         nattempt=nattempt+1;
232     end

233 end %end attempt loop

```

```

234     if nattempt>nattemptmax,
235         disp(' ')
236         disp(['**WARNING: Desired interspin energy cannot be ' ...
237             'achieved with current parameters.'])
238         disp(['**      Proceeding with the nearest ' ...
239             'accessible energy.'])
240         disp(' ')
241     end

242 end %end setflag condition

243 else %choose uniformly random nx,ny

244     nz=sign(M0).*(2*rand(1,nxyz).^((1-abs(M0))./(1+abs(M0)))-1);
245     sintheta=sqrt(1-nz.*nz);

246     phi=twopi*rand(1,nxyz);
247     nx=sintheta.*cos(phi);
248     ny=sintheta.*sin(phi);

249 end %corrflag condition

250 [Bx,By,Bz]=Bc(nx,ny,nz);
251 eint=scalefac*mean(nx.*Bx+ny.*By+nz.*Bz)/2;
252 disp(['interspin energy = ' num2str(eint)])
253 disp(['mean x-component = ' num2str(mean(nx))])
254 disp(['mean y-component = ' num2str(mean(ny))])
255 disp(['mean z-component = ' num2str(mean(nz))])

1 function [nx,ny]=cmlinitxy(cfac,sintheta,ranset,signs,symfun)

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %
4 %                               CMLINITXY.M
5 %
6 % Calculates correlated transverse magnetization components
7 % for cmlinit.m.
8 %
9 % Input:
10 %     cfac: coefficient from secant-rule search
11 %     sintheta: magnitude of transverse magnetization
12 %     ranset: random #'s for magnetization component generation%
13 %     signs: random signs " " " "
14 %     symfun: symmetry function for x magnetization %

```

```

15 % %
16 % Global variables used: %
17 %     lsz,msz,nsz: integer x,y,z lattice sizes %
18 %     kdir: direction of k vector for initial disturbance %
19 %     d3to4: dimension matrix for array reshaping %
20 % %
21 % Output: %
22 %     nx,ny: 1 x lsz x msz x nsz arrays of spin components %
23 % %
24 % DKS 11/5/93 %
25 % %
26 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

27 %global variable declarations
28 % (several statements to stay under max line length)
29 global lsz msz nsz lunit munit nunit ...
30     alpha beta gamma isz jsz ksz ...
31     rfact qtheta qphi ostring ...
32     cset ...
33     twopi vbis wbis v_iso w_iso ...
34     dw nsig;
35 global pflag dflag aterm bterm corrflag setflag ...
36     jflag tflag diffflag convflag swapflag;
37 global nderivmax nderivmin nnorm ...
38     err0 terr scalefac ...
39     enint c0 ...
40     lambda lambdax kdir Mamp;
41 global protectaxes;

42 global dlk d3to4 d4to3 btree wi wb dt;

43 if cfac>=0,
44     altfun=symfun;
45     if kdir==1,
46         altfun(rem(1:lsz,2))=-altfun(rem(1:lsz,2));
47         pmat=cfac*an_ztox(altfun*ones(1,msz*nsz),d3to4);
48     elseif kdir==2,
49         altfun(rem(1:msz,2))=-altfun(rem(1:msz,2));
50         pmat=a3_ztox(altfun*ones(1,nsz*lsz),...
51             [2 3 1; msz nsz lsz]);
52         pmat=cfac*an_ztox(pmat,d3to4);
53     elseif kdir==3,
54         altfun(rem(1:nsz,2))=-altfun(rem(1:nsz,2));
55         pmat=a3_ytox(altfun*ones(1,lsz*msz),...
56             [3 1 2; nsz lsz msz]);

```

```

57     pmat=cfac*an_ztox(pmat,d3to4);
58 elseif kdir==110
59     altfun(rem(1:2*lsz,2))=-altfun(rem(1:2*lsz,2));
60     indperm = (0:lsz-1)'*ones(1,msz)+ones(lsz,1)*(0:msz-1)+1;
61     repind = (1:msz)'*ones(1,nsz);
62     a0=zeros(lsz,msz);
63     a0(:)=altfun(indperm);
64     a0=a0(:,repind(:));
65     pmat=cfac*an_ztox(a0,d3to4);
66 elseif kdir==101
67     altfun(rem(1:2*nsz,2))=-altfun(rem(1:2*nsz,2));
68     indperm = (0:nsz-1)'*ones(1,lsz)+ones(nsz,1)*(0:lsz-1)+1;
69     repind = (1:lsz)'*ones(1,msz);
70     a0=zeros(nsz,lsz);
71     a0(:)=altfun(indperm);
72     a0=a0(:,repind(:));
73     a0=a3_ytox(a0,[3 1 2; nsz lsz msz]);
74     pmat=cfac*an_ztox(a0,d3to4);
75 elseif kdir==011
76     altfun(rem(1:2*msz,2))=-altfun(rem(1:2*msz,2));
77     indperm = (0:msz-1)'*ones(1,nsz)+ones(msz,1)*(0:nsz-1)+1;
78     repind = (1:nsz)'*ones(1,lsz);
79     a0=zeros(msz,nsz);
80     a0(:)=altfun(indperm);
81     a0=a0(:,repind(:));
82     a0=a3_ztox(a0,[2 3 1; msz nsz lsz]);
83     pmat=cfac*an_ztox(a0,d3to4);
84 elseif kdir==111
85     lmnsz=max(max(lsz,msz),nsz);
86     altfun(rem(1:2*lmnsz,2))=-altfun(rem(1:2*lmnsz,2));
87     indpermxy = (0:lsz-1)'*ones(1,msz)+ones(lsz,1)*(0:msz-1);
88     indperm=kron(ones(1,nsz),indpermxy)+...
89         kron(0:nsz-1,ones(lsz,msz))+1;
90     a0=zeros(lsz,msz*nsz);
91     a0(:)=altfun(indperm);
92     a0=reshape(a0,lsz,msz*nsz);
93     pmat=cfac*an_ztox(a0,d3to4);
94 end
95 else
96     if kdir==1,
97         [pmat,dpmat]=a3_ytox(abs(symfun)*ones(1,msz*nsz),...
98             [1 2 3; lsz msz nsz]);
99         pmat(rem(1:msz,2),:)=~pmat(rem(1:msz,2),:);
100        [pmat,dpmat]=a3_ytox(pmat,dpmat);
101        pmat(rem(1:nsz,2),:)=~pmat(rem(1:nsz,2),:);

```

```

102     pmat=a3_ytox(pmat,dpmat);
103     pmat=abs(cfac)*an_ztox(pmat,d3to4);
104     elseif kdir==2,
105         [pmat,dpmat]=a3_ytox(abs(symfun)*ones(1,nsz*lsz),...
106             [2 3 1; msz nsz lsz]);
107         pmat(rem(1:nsz,2),:)=~pmat(rem(1:nsz,2),:);
108         [pmat,dpmat]=a3_ytox(pmat,dpmat);
109         pmat(rem(1:lsz,2),:)=~pmat(rem(1:lsz,2),:);
110         pmat=abs(cfac)*an_ztox(pmat,d3to4);
111     elseif kdir==3,
112         [pmat,dpmat]=a3_ytox(abs(symfun)*ones(1,lsz*msz),...
113             [3 1 2; nsz lsz msz]);
114         pmat(rem(1:lsz,2),:)=~pmat(rem(1:lsz,2),:);
115         [pmat,dpmat]=a3_ytox(pmat,dpmat);
116         pmat(rem(1:msz,2),:)=~pmat(rem(1:msz,2),:);
117         pmat=a3_ztox(pmat,dpmat);
118         pmat=abs(cfac)*an_ztox(pmat,d3to4);
119     elseif kdir==110
120         indperm = (0:lsz-1)'*ones(1,msz)+ones(lsz,1)*(0:msz-1)+1;
121         repind = (1:msz)'*ones(1,nsz);
122         a0=zeros(lsz,msz);
123         a0(:)=symfun(indperm);
124         a0=a0(:,repind(:));
125         [pmat,dpmat]=a3_ytox(a0,[1 2 3; lsz msz nsz]);
126         pmat(rem(1:msz,2),:)=~pmat(rem(1:msz,2),:);
127         [pmat,dpmat]=a3_ytox(pmat,dpmat);
128         pmat(rem(1:nsz,2),:)=~pmat(rem(1:nsz,2),:);
129         pmat=a3_ytox(pmat,dpmat);
130         pmat=abs(cfac)*an_ztox(pmat,d3to4);
131     elseif kdir==101
132         indperm = (0:nsz-1)'*ones(1,lsz)+ones(nsz,1)*(0:lsz-1)+1;
133         repind = (1:lsz)'*ones(1,msz);
134         a0=zeros(nsz,lsz);
135         a0(:)=symfun(indperm);
136         a0=a0(:,repind(:));
137         [pmat,dpmat]=a3_ytox(a0,[3 1 2; nsz lsz msz]);
138         pmat(rem(1:lsz,2),:)=~pmat(rem(1:lsz,2),:);
139         [pmat,dpmat]=a3_ytox(pmat,dpmat);
140         pmat(rem(1:msz,2),:)=~pmat(rem(1:msz,2),:);
141         pmat=a3_ztox(pmat,dpmat);
142         pmat=abs(cfac)*an_ztox(pmat,d3to4);
143     elseif kdir==011
144         indperm = (0:msz-1)'*ones(1,nsz)+ones(msz,1)*(0:nsz-1)+1;
145         repind = (1:nsz)'*ones(1,lsz);
146         a0=zeros(msz,nsz);

```

```

147     a0(:)=symfun(indperm);
148     a0=a0(:,repind(:));
149     [pmat,dpmat]=a3_ytox(a0,[2 3 1; msz nsz lsz]);
150     pmat(rem(1:nsz,2),:)=pmat(rem(1:nsz,2),:);
151     [pmat,dpmat]=a3_ytox(pmat,dpmat);
152     pmat(rem(1:lsz,2),:)=pmat(rem(1:lsz,2),:);
153     pmat=abs(cfac)*an_ztox(pmat,d3to4);
154 elseif kdir==111
155     indpermxy = (0:lsz-1)'*ones(1,msz)+ones(lsz,1)*(0:msz-1);
156     indperm=kron(ones(1,nsz),indpermxy)+...
157         kron(0:nsz-1,ones(lsz,msz))+1;
158     a0=zeros(lsz,msz*nsz);
159     a0(:)=symfun(indperm);
160     a0=reshape(a0,lsz,msz*nsz);
161     [pmat,dpmat]=a3_ytox(a0,[1 2 3; lsz msz nsz]);
162     pmat(rem(1:msz,2),:)=pmat(rem(1:msz,2),:);
163     [pmat,dpmat]=a3_ytox(pmat,dpmat);
164     pmat(rem(1:nsz,2),:)=pmat(rem(1:nsz,2),:);
165     pmat=a3_ytox(pmat,dpmat);
166     pmat=abs(cfac)*an_ztox(pmat,d3to4);
167     end
168 end

169 phi=zeros(1,lsz*msz*nsz);
170 lset=(pmat<=0);
171 gset=(pmat>0);
172 phi(lset)=pi*signs(lset).*ranset(lset).^(1+pmat(lset));
173 phi(gset)=pi*signs(gset).*(1-ranset(gset).^(1-pmat(gset)));

174 nx=sintheta.*cos(phi);
175 ny=sintheta.*sin(phi);

1 function          cmlevolve(Ixsig,Iysig,Izsig,cind)

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %
4 %                      CMLEVOLVE.M
5 %
6 % Calculates classical spin trajectories for a lattice using
7 % Taylor series algorithm. Includes test for Taylor series
8 % convergence and adaptive derivative number.
9 % Includes treatment of random lattice-site vacancies.
10 %
11 % input:
12 %          Ixsig,Iysig,Izsig: 1 x lsz x msz x nsz arrays

```

```

13 %           of initial spin components %
14 %   cind: indices of occupied lattice sites %
15 % %
16 % global variables used: %
17 %   wi: 1 x lsz x msz x nsz array %
18 %           of scaled chemical shifts %
19 %   wb: the scaled dipole coupling strength %
20 %   nsig: number of time points %
21 %   dw: time increment (sec) %
22 %   dt: scaled time increment %
23 %   nderivmax: max. number of derivatives to use %
24 %           in Taylor series (nderiv=2 --> use previous %
25 %           signal and first derivative) %
26 %   nderivmin: min. number of derivatives to use in %
27 %           Taylor series %
28 %   err0: initial target error %
29 %   terr: time constant of error growth %
30 %   aterm,bterm: flags to include or omit A(longitudinal) %
31 %           and B(transverse) dipole field components %
32 %   nnorm: number of time steps between normalizations %
33 %           of trajectories. %
34 %   scalefac: time-coupling scaling factor %
35 %   kdir: direction of k vector of initial disturbance %
36 %           (1 --> x , 2 --> y , 3 --> z) %
37 %   jflag: 1 --> use I*S nearest-neighbor coupling %
38 %           0 --> use dipole coupling %
39 %   tflag: 1 --> use high field truncated dipole coupling %
40 %           0 --> use full (untruncated) dipole coupling %
41 %   protectaxes: vector of axes to protect from projection %
42 %           (1=x,2=y,3=z, any order in the vector) %
43 %           (for storage of multiple sublattice results %
44 %           from a single run) %
45 % %
46 % output: global variables from lists in cmlinput.m %
47 %   <opname>sum: z sums of operators %
48 %   <opname>mean: lattice averages of operators %
49 %   <opname>test: values for one test lattice site %
50 %   e.g. Ixsum,Iysum,Izsum: nsz x nsig vectors of %
51 %           time-evolved spin components %
52 %           Ixdotsum,Iydotsum,Izdotsum: vectors of 1st deriv.'s %
53 %           Bxsum,Bysum,Bzsum: vectors of local fields %
54 % %
55 % DKS 10/8/93 %
56 % DKS 1/10/94 Added protectaxes functionality %
57 % %

```

```

58 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
59 %global variable declarations
60 % (several statements to stay under max line length)
61 global lsz msz nsz lunit munit nunit ...
62     alpha beta gamma isz jsz ksz ...
63     rfact qtheta qphi ostring ...
64     cset ...
65     twopi vbis wbis v_iso w_iso ...
66     dw nsig;
67 global pflag dflag aterm bterm corrflag setflag ...
68     jflag tflag diffflag convflag swapflag;
69 global nderivmax nderivmin nnorm ...
70     err0 terr scalefac ...
71     enint c0 ...
72     lambda lambdax kdir Mamp;
73 global protectaxes;

74 global btree wi wb dt dIk d3to4 d4to3;

75 %global output variables
76 global opnum cmplist testlist namelist;
77 for iop=1:opnum,
78     if isempty(protectaxes),
79         eval(['global ' deblank(namelist(iop,:)) 'sum;']);
80     else
81         eval(['global ' deblank(namelist(iop,:)) 'sums;']);
82     end
83     eval(['global ' deblank(namelist(iop,:)) 'test;']);
84 end

85 if (kdir>3) & ~isempty(protectaxes),
86     disp(['**Protected axes are not compatible with ' ...
87         'off-axis initial magnetization profiles.'])
88     disp(['**Ignoring protectaxes.'])
89 end

90 %initialize variables
91 disp('cmlevolve: initializing variables...')
92 %initialize output variables
93 for iop=1:opnum,
94     eval([deblank(namelist(iop,:)) 'test=zeros(nsig,1);']);
95 end
96 %initialize derivative arrays
97 [Ixder,dIxder]=an_zeros([1 2 3 4; nderivmax lsz msz nsz]);

```

```

98 Iyder=Ixder; dIyder=dIxder;
99 Izder=Ixder; dIzder=dIxder;
100 [Bxder,dBxder]=an_zeros([1 2 3 4; nderivmax lsz msz nsz]);
101 Byder=Bxder; dByder=dBxder;
102 Bzder=Bxder; dBzder=dBxder;

103 %signal loop
104 disp('cmlevolve: signal loop...')
105 %t0=clock;
106 for isig=1:nsig-1
107     %if rem(isig,max([round(nsig/10) 1]))==0,
108     %disp('  ')
109     disp(['time step #' num2str(isig)])
110     %end

111     if rem(isig,nnorm)==0,
112         disp('renormalizing trajectories...')
113         Inorm=sqrt(Ixsig.^2+Iysig.^2+Izsig.^2);
114         Ixsig(cind)=Ixsig(cind)./Inorm(cind);
115         Iysig(cind)=Iysig(cind)./Inorm(cind);
116         Izsig(cind)=Izsig(cind)./Inorm(cind);
117     end

118     Ixder(1,cind)= Ixsig(cind);
119     Iyder(1,cind)= Iysig(cind);
120     Izder(1,cind)= Izsig(cind);

121     [Bxder(1,:),Byder(1,:),Bzder(1,.)]=Bc(Ixsig,Iysig,Izsig);
122     Bzder(1,.)= wi+Bzder(1,.);

123     Ixder(2,cind)= Iysig(cind).*Bzder(1,cind) - ...
124         Izsig(cind).*Byder(1,cind);
125     Iyder(2,cind)= Izsig(cind).*Bxder(1,cind) - ...
126         Ixsig(cind).*Bzder(1,cind);
127     Izder(2,cind)= Ixsig(cind).*Byder(1,cind) - ...
128         Iysig(cind).*Bxder(1,cind);

129     [Bxder(2,:),Byder(2,:),Bzder(2,.)]=Bc(...
130         Ixder(2,:),Iyder(2,:),Izder(2,));

131     Ix=Ixsig;
132     Iy=Iysig;
133     Iz=Izsig;
134     Ixdot=scalefac*Ixder(2,.);
135     Iydot=scalefac*Iyder(2,.);

```

```

136  Izdot=scalefac*Izder(2,:);
137  Bx=scalefac*Bxder(1,:);
138  By=scalefac*Byder(1,:);
139  Bz=scalefac*Bzder(1,:);
140  Bxdot=scalefac*scalefac*Bxder(2,:);
141  Bydot=scalefac*scalefac*Byder(2,:);
142  Bzdot=scalefac*scalefac*Bzder(2,:);

143  for iop=1:opnum,
144      if kdir<=3,
145          if isempty(protectaxes),
146              eval([deblank(namelist(iop,:)) 'sum(:,isig)=xyzsums(' ...
147                  cmoplist(iop,:) ',d4to3,kdir);']);
148          else
149              eval([deblank(namelist(iop,:)) ...
150                  'sums(isig,:)=xyzprotsums(' ...
151                  cmoplist(iop,:) ',d4to3,kdir,' ...
152                  'protectaxes);']);
153          end
154      else
155          eval([deblank(namelist(iop,:)) 'sum(:,isig)=projsums(' ...
156                  cmoplist(iop,:) ',d4to3,kdir);']);
157      end
158      eval([deblank(namelist(iop,:)) 'test(isig)=( ' ...
159          testlist(iop,:) ');']);
160  end

161  err=err0*exp(dw*(isig-1)/terr);

162  if dflag,
163      disp(' ')
164      disp([' isig=' num2str(isig)])
165      Inorm=sqrt(Ixsig.^2+Iysig.^2+Izsig.^2);
166      disp([' Inorm from ' num2str(min(Inorm)) ' to ' ...
167          num2str(max(Inorm))])
168      disp(' ')
169      if pflag,
170          clf
171          plot(real(Inorm))
172          pause(.1)
173      end
174      disp(['      time=' num2str(dw*(isig-1)) ...
175          ' ider=1: fac=1 tfac=1 tfac/fac=1'])
176      meanterm=sqrt(mean(abs(Ixder(1,:)).^2+...
177          abs(Iyder(1,:)).^2+abs(Izder(1,:)).^2)/3);

```

```

178     disp(['          RMS truncation error = ' ...
179           num2str(meanterm)])
180 end

181 fac=1;
182 tfac=dt;
183 ider=2;

184 meanterm=sqrt(mean(abs(Ixder(ider,cind)).^2+...
185                   abs(Iyder(ider,cind)).^2+...
186                   abs(Izder(ider,cind)).^2)/3)*tfac/fac;

187 while ider<=nderivmin | (ider<=nderivmax & meanterm>err),

188     if dflag,
189         disp(['          time=' num2str(dw*(isig-1)) ...
190               ' ider=' num2str(ider) ': tfac/fac=' ...
191               num2str(tfac/fac)])
192         disp(['          RMS truncation error = ' ...
193               num2str(meanterm)])
194         if pflag,
195             subplot(111); plot(meanterm);
196             title('RMS truncation error')
197             pause(.1)
198         end
199     end

200     Ixsig(cind) = Ixsig(cind) + Ixder(ider,cind)*tfac/fac;
201     Iysig(cind) = Iysig(cind) + Iyder(ider,cind)*tfac/fac;
202     Izsig(cind) = Izsig(cind) + Izder(ider,cind)*tfac/fac;

203     Ixder(ider+1,cind) = btree(ider,1:ider) * ...
204         (Iyder(ider:-1:1,cind).*Bzder(1:ider,cind) - ...
205         Izder(ider:-1:1,cind).*Byder(1:ider,cind));
206     Iyder(ider+1,cind) = btree(ider,1:ider) * ...
207         (Izder(ider:-1:1,cind).*Bxder(1:ider,cind) - ...
208         Ixder(ider:-1:1,cind).*Bzder(1:ider,cind));
209     Izder(ider+1,cind) = btree(ider,1:ider) * ...
210         (Ixder(ider:-1:1,cind).*Byder(1:ider,cind) - ...
211         Iyder(ider:-1:1,cind).*Bxder(1:ider,cind));

212     [Bxder(ider+1,:),Byder(ider+1,:),Bzder(ider+1,:)] = Bc(...
213         Ixder(ider+1,:),Iyder(ider+1,:),Izder(ider+1,:));

```

```

214     fac=fac*ider;
215     tfac=tfac*dt;
216     ider=ider+1;

217     meanterm=sqrt(mean(abs(Ixder(ider,cind)).^2+...
218         abs(Iyder(ider,cind)).^2+...
219         abs(Izder(ider,cind)).^2)/3)*tfac/fac;

220     if (ider==(nderivmax+1)) & (meanterm>err),
221         disp('**maximum number of derivatives exceeded**')
222     end

223 end
224 %^end derivative loop

225 disp(['    number of derivatives=' num2str(ider-1)])

226 end
227 %^end signal loop

228 disp(['time step #' num2str(nsig)])

229 if rem(nsig,nnorm)==0,
230     disp('renormalizing trajectories...')
231     Inorm=sqrt(Ixsig.^2+Iysig.^2+Izsig.^2);
232     Ixsig(cind)=Ixsig(cind)./Inorm(cind);
233     Iysig(cind)=Iysig(cind)./Inorm(cind);
234     Izsig(cind)=Izsig(cind)./Inorm(cind);
235 end

236 [Bxder(1,:),Byder(1,:),Bzder(1,.)]=Bc(Ixsig,Iysig,Izsig);
237 Bzder(1,:)= wi+Bzder(1,:);

238 Ixder(2,cind)= Iysig(cind).*Bzder(1,cind) - ...
239     Izsig(cind).*Byder(1,cind);
240 Iyder(2,cind)= Izsig(cind).*Bxder(1,cind) - ...
241     Ixsig(cind).*Bzder(1,cind);
242 Izder(2,cind)= Ixsig(cind).*Byder(1,cind) - ...
243     Iysig(cind).*Bxder(1,cind);

244 Ix=Ixsig;
245 Iy=Iysig;
246 Iz=Izsig;
247 Ixdot=scalefac*Ixder(2,:);
248 Iydot=scalefac*Iyder(2,:);

```

```

249 Izdot=scalefac*Izder(2,:);
250 Bx=scalefac*Bxder(1,:);
251 By=scalefac*Byder(1,:);
252 Bz=scalefac*Bzder(1,:);
253 for iop=1:opnum,
254     if kdir<=3,
255         if isempty(protectaxes),
256             eval([deblank(namelist(iop,:)) 'sum(:,nsig)=xyzsums(' ...
257                 cmoplist(iop,:) ',d4to3,kdir);']);
258         else
259             eval([deblank(namelist(iop,:)) ...
260                 'sums(nsig,:)=xyzprotsums(' ...
261                 cmoplist(iop,:) ',d4to3,kdir,' ...
262                 'protectaxes);']);
263         end
264     else
265         eval([deblank(namelist(iop,:)) 'sum(:,nsig)=projsums(' ...
266             cmoplist(iop,:) ',d4to3,kdir);']);
267     end
268     eval([deblank(namelist(iop,:)) 'test(nsig)=( ' ...
269         testlist(iop,:) ');']);
270 end

```

```

271 %telapsed=etime(clock,t0);
272 %disp(['signal loop time ' num2str(telapsed) 'seconds'])

```

```

1 function [Bx,By,Bz]=Bc(Ix,Iy,Iz)

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %                                     Bc.m                                     %
4 %                                     %                                     %
5 % Calculates mean dipole field at each lattice site given                 %
6 % initial spin orientations for cmlevolve.m.                               %
7 %                                     %                                     %
8 % Input: Ix,Iy,Iz: 1 x lsz x msz x nsz arrays of                         %
9 %       initial spin orientations                                         %
10 %                                     %                                     %
11 % Global variables from cmlinput.m:                                       %
12 %       convflag: 1 --> calculate local fields by convolution              %
13 %                0 --> calculate local fields by summation               %
14 %       swapflag: 1 --> use properly octant-swapped 3D FFT's             %
15 %                for convolution                                          %
16 %                0 --> use unswapped 3D FFT's for convolution            %
17 %       aterm,bterm: flags to include or omit A(longitudinal)           %
18 %                and B(transverse) dipole field components               %

```

```

19 %      jflag: 1 --> use I*S nearest-neighbor coupling      %
20 %          0 --> use dipole coupling                        %
21 %      tflag: 1 --> use high field truncated dipole coupling %
22 %          0 --> use full (untruncated) dipole coupling    %
23 %      isz,jsz,ksz: interaction region size                 %
24 %                                                           %
25 % Global variables from cmlevolve.m                         %
26 %      wb: scaled, signed dipole coupling strength (rad/sec) %
27 %      dIk,d3to4,d4to3: dimension matrices for dim-shifting %
28 %      if tflag,                                           %
29 %          Cmat: dipole coupling matrix                    %
30 %          Chat: FFT of coupling matrix                    %
31 %      else                                               %
32 %          Cm11,Cm12,Cm13,Cm22,Cm23,Cm33: dipole coupling %
33 %                                     tensor elements      %
34 %          Ch11,Ch12,Ch13,Ch22,Ch23,Ch33: FFT's          %
35 %      end                                               %
36 %                                                           %
37 % Output: Bx,By,Bz: 1 x lsz x msz x nsz arrays of local fields %
38 %                                                           %
39 % DKS 9/93-10/93                                         %
40 % DKS 1/10/94 jflag option implemented                    %
41 %                                                           %
42 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

43 %global variable declarations
44 % (several statements to stay under max line length)
45 global lsz msz nsz lunit munit nunit ...
46     alpha beta gamma isz jsz ksz ...
47     rfact qtheta qphi ostring ...
48     cset ...
49     twopi vbis wbis v_iso w_iso ...
50     dw nsig;
51 global pflag dflag aterm bterm corrflag setflag ...
52     jflag tflag diffflag convflag swapflag;
53 global nderivmax nderivmin nnorm ...
54     err0 terr scalefac ...
55     enint c0 ...
56     lambda lambdax kdir Mamp;
57 global protectaxes;

58 global btree wi wb dIk d3to4 d4to3;

59 if jflag, % nearest-neighbor exchange coupling

```

```

60  Ix=an_ytox(Ix,d4to3);
61  Iy=an_ytox(Iy,d4to3);
62  Iz=an_ytox(Iz,d4to3);

63  Bx=zeros(size(Ix)); By=Bx; Bz=Bx;

64  if (isz>1),
65      lind=cshift(1:lsz,-1);
66      Bx=Bx+bterm*wb*Ix(lind,:);
67      By=By+bterm*wb*Iy(lind,:);
68      Bz=Bz+aterm*wb*Iz(lind,:);
69      if isz>2,
70          lind=cshift(1:lsz,1);
71          Bx=Bx+bterm*wb*Ix(lind,:);
72          By=By+bterm*wb*Iy(lind,:);
73          Bz=Bz+aterm*wb*Iz(lind,:);
74      end
75  end

76  if (jsz>1),
77      mind=cshift(1:msz,-1);
78      nind=1:nsz;
79      mnind=kron(ones(size(nind)),mind);
80      nmind=kron(nind,ones(size(mind)))-1;
81      Bx=Bx+bterm*wb*Ix(:,nmind*msz+mnind);
82      By=By+bterm*wb*Iy(:,nmind*msz+mnind);
83      Bz=Bz+aterm*wb*Iz(:,nmind*msz+mnind);
84      if jsz>2,
85          mind=cshift(1:msz,1);
86          mnind=kron(ones(size(nind)),mind);
87          Bx=Bx+bterm*wb*Ix(:,nmind*msz+mnind);
88          By=By+bterm*wb*Iy(:,nmind*msz+mnind);
89          Bz=Bz+aterm*wb*Iz(:,nmind*msz+mnind);
90      end
91  end

92  if (ksz>1),
93      mind=1:msz;
94      nind=cshift(1:nsz,-1);
95      mnind=kron(ones(size(nind)),mind);
96      nmind=kron(nind,ones(size(mind)))-1;
97      Bx=Bx+bterm*wb*Ix(:,nmind*msz+mnind);
98      By=By+bterm*wb*Iy(:,nmind*msz+mnind);
99      Bz=Bz+aterm*wb*Iz(:,nmind*msz+mnind);
100  if ksz>2,

```

```

101     nind=cshift(1:nsz,1);
102     nmind=kron(nind,ones(size(mind)))-1;
103     Bx=Bx+bterm*wb*Ix(:,nmind*msz+mnind);
104     By=By+bterm*wb*Iy(:,nmind*msz+mnind);
105     Bz=Bz+aterm*wb*Iz(:,nmind*msz+mnind);
106     end
107 end

108 Bx=an_ztox(Bx,d3to4);
109 By=an_ztox(By,d3to4);
110 Bz=an_ztox(Bz,d3to4);

111 else % dipole-dipole coupling

112     if tflag,
113         global Cmat Chat;
114     else
115         global Cm11 Cm12 Cm13 Cm22 Cm23 Cm33 ...
116             Ch11 Ch12 Ch13 Ch22 Ch23 Ch33;
117     end

118     %local field calculation
119     if convflag,
120         %calculate local fields by convolution
121         if swapflag,
122             % properly octant-swapped 3D FFT's
123             % disp('  FFT of trajectories')
124             Iqx=a3_fftsxyz(an_ytox(Ix,d4to3),dIk);
125             Iqy=a3_fftsxyz(an_ytox(Iy,d4to3),dIk);
126             Iqz=a3_fftsxyz(an_ytox(Iz,d4to3),dIk);

127             %disp('  calculating local fields by convolution')
128             if tflag,
129                 Bx =real( bterm*( ...
130                     wb*an_ztox(a3_ifftsxyz(Chat.*Iqx,dIk),d3to4)));
131                 By =real( bterm*( ...
132                     wb*an_ztox(a3_ifftsxyz(Chat.*Iqy,dIk),d3to4)));
133                 Bz =real(-2*aterm*( ...
134                     wb*an_ztox(a3_ifftsxyz(Chat.*Iqz,dIk),d3to4)));
135             else
136                 Bx =real( bterm*( wb*an_ztox(a3_ifftsxyz(...
137                     Ch11.*Iqx+Ch12.*Iqy+Ch13.*Iqz,dIk),d3to4)));
138                 By =real( bterm*( wb*an_ztox(a3_ifftsxyz(...
139                     Ch12.*Iqx+Ch22.*Iqy+Ch23.*Iqz,dIk),d3to4)));
140                 Bz =real( aterm*( wb*an_ztox(a3_ifftsxyz(...

```

```

141         Ch13.*Ikx+Ch23.*Iky+Ch33.*Ikz,dIk),d3to4)));
142     end
143 else
144     % unswapped 3D FFT's (only to be multiplied and inverted)
145     % disp('  FFT of trajectories')
146     Ix=a3_fftxyz(an_ytox(Ix,d4to3),dIk);
147     Iky=a3_fftxyz(an_ytox(Iy,d4to3),dIk);
148     Ikz=a3_fftxyz(an_ytox(Iz,d4to3),dIk);

149     if tflag,
150         Bx =real(  bterm*( ...
151             wb*an_ztox(a3_ifftxyz(Chat.*Ikx,dIk),d3to4)));
152         By =real(  bterm*( ...
153             wb*an_ztox(a3_ifftxyz(Chat.*Iky,dIk),d3to4)));
154         Bz =real(-2*aterm*( ...
155             wb*an_ztox(a3_ifftxyz(Chat.*Ikz,dIk),d3to4)));
156     else
157         Bx =real(  bterm*(  wb*an_ztox(a3_ifftxyz(...
158             Ch11.*Ikx+Ch12.*Iky+Ch13.*Ikz,dIk),d3to4)));
159         By =real(  bterm*(  wb*an_ztox(a3_ifftxyz(...
160             Ch12.*Ikx+Ch22.*Iky+Ch23.*Ikz,dIk),d3to4)));
161         Bz =real(  aterm*(  wb*an_ztox(a3_ifftxyz(...
162             Ch13.*Ikx+Ch23.*Iky+Ch33.*Ikz,dIk),d3to4)));
163     end
164 end

165 %disp('Local x fields via convolution:')
166 %an_ytox(Bx,d4to3)
167 %surf(real(an_ytox(Bx,d4to3)));
168 %pause(1)

169 else
170     %calculate local fields by direct summation
171     Ix3d=an_ytox(Ix,d4to3);
172     Iy3d=an_ytox(Iy,d4to3);
173     Iz3d=an_ytox(Iz,d4to3);

174     if tflag,
175         Bx =  bterm*( ...
176             wb*an_ztox(blsum(Cmat,Ix3d,isz,jsz,ksz,1,1,1),d3to4));
177         By =  bterm*( ...
178             wb*an_ztox(blsum(Cmat,Iy3d,isz,jsz,ksz,1,1,1),d3to4));
179         Bz =-2*aterm*( ...
180             wb*an_ztox(blsum(Cmat,Iz3d,isz,jsz,ksz,1,1,1),d3to4));
181     else

```

```

182     Bx = bterm*( ...
183         wb*an_ztox(blsum(Cm11,Ix3d,isz,jsz,ksz,1,1,1)+...
184         blsum(Cm12,Iy3d,isz,jsz,ksz,1,1,1)+...
185         blsum(Cm13,Iz3d,isz,jsz,ksz,1,1,1),d3to4));
186     By = bterm*( ...
187         wb*an_ztox(blsum(Cm12,Ix3d,isz,jsz,ksz,1,1,1)+...
188         blsum(Cm22,Iy3d,isz,jsz,ksz,1,1,1)+...
189         blsum(Cm23,Iz3d,isz,jsz,ksz,1,1,1),d3to4));
190     Bz = aterm*( ...
191         wb*an_ztox(blsum(Cm13,Ix3d,isz,jsz,ksz,1,1,1)+...
192         blsum(Cm23,Iy3d,isz,jsz,ksz,1,1,1)+...
193         blsum(Cm33,Iz3d,isz,jsz,ksz,1,1,1),d3to4));
194 end

195     %disp('Local x fields via direct summation:')
196     %an_ytox(Bx,d4to3)
197     %surf(real(an_ytox(Bx,d4to3)));
198     %pause(1)

```

```

199 end

```

```

200 end

```

```

1 function      Bj=blsum(Cmat,Ij,lsz,msz,nsz,lcent,mcent,ncent)

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %
4 %              BLSUM.M
5 %
6 % Calculate array of local fields for cmlattice.m (Bc.m)
7 % by direct summation.
8 %
9 % Input:      Cmat: lsz x msz x nsz dipole-coupling array
10 %            (A_FILES format)
11 %            Ij: lsz x msz x nsz array of spin vector comp.'s%
12 %            lsz,msz,nsz: dimensions of lattice
13 %            lcent,mcent,ncent: indices of lattice center
14 %
15 % Output:    Bj: lsz x msz x nsz array of local field comp.'s%
16 %
17 % DKS 5/26/93
18 %
19 % NEXT: REMOVE ZERO MULTIPLICATIONS FOR ISZ,JSZ,KSZ<LSZ,MSZ,NSZ
20 %
21 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

22  [Bj,dBj]=a3_zeros([1 2 3;lsz,msz,nsz]);

23  for il=1:lsz,
24      ldiff=lcent-il;
25      lind=[(lsz+ldiff+1-(ldiff>=0)*lsz):lsz ...
26              1:(lsz+ldiff-(ldiff>=0)*lsz)];
27      for im=1:msz,
28          mdiff=mcent-im;
29          mind=[(msz+mdiff+1-(mdiff>=0)*msz):msz ...
30                  1:(msz+mdiff-(mdiff>=0)*msz)];
31          mnind=kron(ones(1,nsz),mind);
32          for in=1:nsz,
33              ndiff=ncent-in;
34              nind=[(nsz+ndiff+1-(ndiff>=0)*nsz):nsz ...
35                      1:(nsz+ndiff-(ndiff>=0)*nsz)];
36              nmind=kron(nind,ones(1,msz))-1;
37              Cmsht=Cmat(lind,nmind*msz+mnind);
38              Bj=a3i_el(Bj,dBj,sum(sum(Cmsht.*Ij)),[il im in]);
39              %N.B. Bz must be multiplied by -2 and all components
40              %must be multiplied by the coupling constant to
41              %yield the true local fields. Do this in the
42              %calling routine.
43          end
44      end
45  end

```

```

1  function Isum=xyzsums(Imat,dImat,kdir)

```

```

2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  %
4  %                               XYZSUMS.M
5  %
6  % Converts a 4D A_FILES array of Imat on a 3D spatial lattice
7  % vs. time to an array of plane Imat sums along a chosen
8  % direction vs. time (Imat sums in columns, time in rows)
9  %
10 % Input:
11 %     Imat: nsig x lsz x msz x nsz array of z magnetization
12 %     dImat: dimension matrix for Imat
13 %           (dImat=[4 1 2 3 ; nsig lsz msz nsz])
14 %     kdir: direction of k vector for
15 %           1 --> x sums: sum y-z plane for Isum(x,t)
16 %           2 --> y sums: sum x-z plane for Isum(y,t)
17 %           3 --> z sums: sum x-y plane for Isum(z,t)

```

```

18 % %
19 % Output: %
20 % Isum: (lsz,msz,or nsz) x nsig array of plane Imat sums %
21 % %
22 % DKS 10/9/93 %
23 % DKS 10/29/93 safety conditions for when lsz, msz, or nsz = 1 %
24 % DKS 12/1/93 add normalization of sums %
25 % %
26 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

27 if kdir==1,
28     [Imatx,dImatx]=an_ytox(Imat,dImat);
29     [Imaty,dImaty]=an_ytox(Imatx,dImatx);
30     if dImaty(2,1)==1,
31         Imatysum=Imaty;
32     else
33         Imatysum=mean(Imaty);
34     end
35     dImatysum=dImaty;
36     dImatysum(2,1)=1;
37     [Imatz,dImatz]=an_ytox(Imatysum,dImatysum);
38     if dImatz(2,1)==1,
39         Imatzsum=Imatz;
40     else
41         Imatzsum=mean(Imatz);
42     end
43     dImatzsum=dImatz;
44     dImatzsum(2,1)=1;
45     [Imatyzsum,dImatyzsum]=an_ztox(Imatzsum,dImatzsum);
46     Isum=an_ztox(Imatyzsum,dImatyzsum);
47 elseif kdir==2,
48     [Imatx,dImatx]=an_ytox(Imat,dImat);
49     if dImatx(2,1)==1,
50         Imatxsum=Imatx;
51     else
52         Imatxsum=mean(Imatx);
53     end
54     dImatxsum=dImatx;
55     dImatxsum(2,1)=1;
56     [Imaty,dImaty]=an_ytox(Imatxsum,dImatxsum);
57     [Imatz,dImatz]=an_ytox(Imaty,dImaty);
58     if dImatz(2,1)==1,
59         Imatzsum=Imatz;
60     else
61         Imatzsum=mean(Imatz);

```

```

62     end
63     dImatzsum=dImatz;
64     dImatzsum(2,1)=1;
65     Isum=an_ztox(Imatzsum,dImatzsum);
66 else
67     [Imatx,dImatx]=an_ytox(Imat,dImat);
68     if dImatx(2,1)==1,
69         Imatxsum=Imatx;
70     else
71         Imatxsum=mean(Imatx);
72     end
73     dImatxsum=dImatx;
74     dImatxsum(2,1)=1;
75     [Imaty,dImaty]=an_ytox(Imatxsum,dImatxsum);
76     if dImaty(2,1)==1,
77         Imatysum=Imaty;
78     else
79         Imatysum=mean(Imaty);
80     end
81     dImatysum=dImaty;
82     dImatysum(2,1)=1;
83     Isum=an_ytox(Imatysum,dImatysum);
84 end

1 function Isum=xyzprotsums(Imat,dImat,kdir,protectaxes)

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %
4 %                               XYZPROTSUMS.M
5 %
6 % Converts a 4D A_FILES array of Imat on a 3D spatial lattice
7 % vs. time to an array of plane Imat sums along a chosen
8 % direction vs. time (Imat sums in columns, time in rows)
9 %
10 % Input:
11 %     Imat: nsig x lsz x msz x nsz array of z magnetization
12 %     dImat: dimension matrix for Imat
13 %           (dImat=[4 1 2 3 ; nsig lsz msz nsz])
14 %     kdir: direction of k vector for
15 %           1 --> x sums: sum y-z plane for Isum(x,t)
16 %           2 --> y sums: sum x-z plane for Isum(y,t)
17 %           3 --> z sums: sum x-y plane for Isum(z,t)
18 %     protectaxes: vector of axes to protect from projection
19 %                 (1=x,2=y,3=z, any order in the vector)
20 %                 (for storage of multiple sublattice

```

```

21 %                      results from a single run)          %
22 %                                                              %
23 % Output:                                                    %
24 %       Isum: nsig x (lsz x msz x nsz/protected axis dimensions)%
25 %           array of Imat projections                        %
26 %                                                              %
27 % DKS 10/9/93                                                %
28 % DKS 10/29/93 safety conditions for when lsz, msz, or nsz = 1 %
29 % DKS 12/1/93 add normalization of sums                    %
30 % DKS 1/10/94 add protectaxes functionality                %
31 %                                                              %
32 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

33 if nargin<4,
34     protectaxes=[];
35 end

36 if kdir==1,
37     [Imatx,dImatx]=an_ytox(Imat,dImat);
38     [Imaty,dImaty]=an_ytox(Imatx,dImatx);
39     if (dImaty(2,1)==1) | any(protectaxes==2),
40         Imatysum=Imaty;
41         dImatysum=dImaty;
42     else
43         Imatysum=mean(Imaty);
44         dImatysum=dImaty;
45         dImatysum(2,1)=1;
46     end
47     [Imatz,dImatz]=an_ytox(Imatysum,dImatysum);
48     if (dImatz(2,1)==1) | any(protectaxes==3),
49         Imatzsum=Imatz;
50         dImatzsum=dImatz;
51     else
52         Imatzsum=mean(Imatz);
53         dImatzsum=dImatz;
54         dImatzsum(2,1)=1;
55     end
56     Isum=an_ytox(Imatzsum,dImatzsum);
57 elseif kdir==2,
58     [Imatx,dImatx]=an_ytox(Imat,dImat);
59     if (dImatx(2,1)==1) | any(protectaxes==1),
60         Imatxsum=Imatx;
61         dImatxsum=dImatx;
62     else
63         Imatxsum=mean(Imatx);

```

```

64     dImatxsum=dImatx;
65     dImatxsum(2,1)=1;
66     end
67     [Imaty,dImaty]=an_ytox(Imatxsum,dImatxsum);
68     [Imatz,dImatz]=an_ytox(Imaty,dImaty);
69     if (dImatz(2,1)==1) | any(protectaxes==3),
70         Imatzsum=Imatz;
71         dImatzsum=dImatz;
72     else
73         Imatzsum=mean(Imatz);
74         dImatzsum=dImatz;
75         dImatzsum(2,1)=1;
76     end
77     Isum=an_ytox(Imatzsum,dImatzsum);
78 else
79     [Imatx,dImatx]=an_ytox(Imat,dImat);
80     if (dImatx(2,1)==1) | any(protectaxes==1),
81         Imatxsum=Imatx;
82         dImatxsum=dImatx;
83     else
84         Imatxsum=mean(Imatx);
85         dImatxsum=dImatx;
86         dImatxsum(2,1)=1;
87     end
88     [Imaty,dImaty]=an_ytox(Imatxsum,dImatxsum);
89     if (dImaty(2,1)==1) | any(protectaxes==2),
90         Imatysum=Imaty;
91         dImatysum=dImaty;
92     else
93         Imatysum=mean(Imaty);
94         dImatysum=dImaty;
95         dImatysum(2,1)=1;
96     end
97     [Imatxysum,dImatxysum]=an_ytox(Imatysum,dImatysum);
98     Isum=an_ytox(Imatxysum,dImatxysum);
99 end

```

```

1 function Iproj=projsums(Imat,dImat,kdir)

```

```

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %                                                                 %
4 %                               PROJSUMS.M                        %
5 %                                                                 %
6 % Projects a 3D spatial lattice onto a chosen axis.            %
7 %                                                                 %

```

```

8 % Input: %
9 %      Imat: 1 x lsz x msz x nsz array of z magnetization %
10 %      dImat: dimension matrix for Imat %
11 %      (dImat=[4 1 2 3 ; 1 lsz msz nsz]) %
12 %      kdir: integer describing projection axis %
13 % %
14 % Output: %
15 %      Isum: (?) x nsig array of plane Imat sums %
16 % %
17 % DKS 11/30/93 %
18 % %
19 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

20 lsz = dImat(2,2);
21 msz = dImat(2,3);
22 nsz = dImat(2,4);

23 if kdir == 1,
24     theta = 90;
25     phi = 0;
26 elseif kdir == 2,
27     theta = 90;
28     phi = 90;
29 elseif kdir == 3,
30     theta = 0;
31     phi = 0;
32 elseif kdir == 110,
33     theta = 90;
34     phi = 45;
35 elseif kdir == 101,
36     theta = 45;
37     phi = 0;
38 elseif kdir == 011,
39     theta = 45;
40     phi = 90;
41 elseif kdir == 111,
42     theta = 54.7;
43     phi = 45;
44 end

45 %make accurate magic angle
46 if theta < 54.8 & theta > 54.6,
47     theta=atan(sqrt(2))*180/acos(-1);
48 end
49 %initialize constants

```

```

50 drfac=pi/180;
51 if theta ~= 0,
52   ctheta=cos(theta*drfac);
53   stheta=sin(theta*drfac);
54   cphi=cos(phi*drfac);
55   sphi=sin(phi*drfac);
56 end

57 nc=[0 ; 1 ; nsz-1];
58 mc=[0 ; 1 ; msz-1];
59 lc=[0 ; 1 ; lsz-1];
60 [C,dC]=a3_coord([lc mc nc]);
61 Xc=ano_arr(C,dC,1); Yc=ano_arr(C,dC,2); Zc=ano_arr(C,dC,3);
62 if abs(theta)>eps,
63   %coordinate transform for lattices other than [100]
64   Zc=(Xc*cphi+Yc*sphi)*stheta+Zc*ctheta;
65 end

66 Imat = an_ytox(Imat,dImat);

67 inc=0.5;
68 dinc=1;
69 Itest=((Zc<inc) & (Zc>=(inc-dinc)));
70 while any(any(Itest))
71   Iproj(inc)=sum(sum(Imat(Itest)))/nnz(Itest);
72   inc=inc+dinc;
73   Itest=((Zc<inc) & (Zc>=(inc-dinc)));
74 end

75 Iproj = Iproj(:);

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %                                                                 %
3  %                               CMLPROC.M                          %
4  %                                                                 %
5  % Process data from cmlattice.m to generate and plot             %
6  % magnetization or energy profiles and spin diffusion constants. %
7  %                                                                 %
8  % DKS 10/93                                                       %
9  %                                                                 %
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

11 if procflag,

12   %make necessary constants.

```

```

13  xax=(0:nsig-1)*dw;
14  dirstring=['x','y','z'];

15  if kdir == 1,
16      nax=lsz;
17      ampnorm=2/lsz;
18  elseif kdir == 2,
19      nax=msz;
20      ampnorm=2/msz;
21  elseif kdir == 3,
22      nax=nsz;
23      ampnorm=2/nsz;
24  elseif kdir == 110,
25      nax=min(lsz,msz);
26      ampnorm=2/nsz;
27  elseif kdir == 101,
28      nax=min(lsz,nsz);
29      ampnorm=2/msz;
30  elseif kdir == 011,
31      nax=min(msz,nsz);
32      ampnorm=2/lsz;
33  elseif kdir == 111,
34      nax=min(min(lsz,msz),nsz);
35      ampnorm=2/nax;
36  end

37  if isempty(protectaxes) | (kdir>3),

38      %find means
39      for iop=1:opnum,
40          eval([deblank(namelist(iop,:)) 'mean=mean(' ...
41              deblank(namelist(iop,:)) 'sum);'])
42      end

43      %construct amplitude and diffusion constant curves
44      fullax=1:length(Izsum(:,1));
45      xyzax=1:nax;
46      kind=1:ceil((nax+1)/2);
47      kax=(kind-1)*twopi/nax;
48      ki=rem(ceil(nax/lambda),max(kind))+1;
49      kix=rem(ceil(nax/lambda),max(kind))+1;

50      Izfft=real(fft(Izsum(xyzax,:)));
51      Izdotfft=real(fft(Izdotsum(xyzax,:)));
52      Eintfft=real(fft(eintsum(xyzax,:)));

```

```

53     Eintdotfft=real(fft(eintdotsum(xyzax,:)));
54     Dmat=-Izdotfft(kind,:)./Izfft(kind,:)/...
55         ((kax(:).^2)*ones(1,nsig));
56     DEmat=-Eintdotfft(kind,:)./Eintfft(kind,:)/...
57         ((kax(:).^2)*ones(1,nsig));
58     Dk=Dmat(ki,:);
59     DEk=DEmat(kix,:);
60     Izamp=Izfft(kind,:)*ampnorm;
61     Eintamp=Eintfft(kind,:)*ampnorm;
62     Izder=Izdotfft(kind,:)*ampnorm;
63     Eintder=Eintdotfft(kind,:)*ampnorm;
64     Ak=Izamp(ki,:);
65     AEk=-Eintamp(kix,:);

66     else

67         %find proper dimension matrix for data arrays
68         dopsums=[4 1 2 3; nsig lsz msz nsz];
69         nreps=1;
70         for idir=1:3,
71             if (idir~=kdir) & ~any(protectaxes==idir),
72                 dopsums(2,idir+1)=1;
73             end
74             if any(protectaxes==idir),
75                 nreps=nreps*dopsums(2,idir+1);
76             end
77         end

78         %average over repetitions
79         for iop=1:opnum,
80             eval([' deblank(namelist(iop,:)) 'sum=' ...
81                 deblank(namelist(iop,:)) 'sums;'])
82             dopproc=dopsums;
83             for idir=1:3,
84                 eval([' deblank(namelist(iop,:)) ...
85                     'sum,dopproc]=an_ytox(' ...
86                     deblank(namelist(iop,:)) 'sum,dopproc);'])
87                 if any(protectaxes==idir) & (dopproc(2,1)>1),
88                     eval([' deblank(namelist(iop,:)) 'sum=mean(' ...
89                         deblank(namelist(iop,:)) 'sum);'])
90                     dopproc(2,1)=1;
91                 end
92             end
93             eval([' deblank(namelist(iop,:)) ...
94                 'sum,dopproc]=an_ytox(' ...

```

```

95         deblank(namelist(iop,:)) 'sum,dopproc;']
96     end

97     %permute data arrays to put kdir axis along x
98     for iop=1:opnum,
99         if kdir==1,
100             eval([deblank(namelist(iop,:)) 'sums=an_ytox(' ...
101                 deblank(namelist(iop,:)) 'sums,dopsums);'])
102             eval([deblank(namelist(iop,:)) 'sum=an_ytox(' ...
103                 deblank(namelist(iop,:)) 'sum,dopproc;'])
104         elseif kdir==2,
105             eval([deblank(namelist(iop,:)) 'sums=an_ytox(' ...
106                 deblank(namelist(iop,:)) 'sums,dopsums);'])
107             eval([deblank(namelist(iop,:)) 'sums=an_ytox(' ...
108                 deblank(namelist(iop,:)) ...
109                 'sums,dopsums(:, [2 3 4 1]);'])
110             eval([deblank(namelist(iop,:)) 'sum=an_ytox(' ...
111                 deblank(namelist(iop,:)) 'sum,dopproc;'])
112             eval([deblank(namelist(iop,:)) 'sum=an_ytox(' ...
113                 deblank(namelist(iop,:)) ...
114                 'sum,dopproc(:, [2 3 4 1]);'])
115         elseif kdir==3,
116             eval([deblank(namelist(iop,:)) 'sums=an_ztox(' ...
117                 deblank(namelist(iop,:)) 'sums,dopsums);'])
118             eval([deblank(namelist(iop,:)) 'sum=an_ztox(' ...
119                 deblank(namelist(iop,:)) 'sum,dopproc;'])
120         end
121         eval([deblank(namelist(iop,:)) 'means=mean(' ...
122             deblank(namelist(iop,:)) 'sums);'])
123         eval([deblank(namelist(iop,:)) 'mean=mean(' ...
124             deblank(namelist(iop,:)) 'sum);'])
125     end

126     %construct amplitude and diffusion constant curves
127     %multiple repetitions
128     fullax=1:length(Izsums(:,1));
129     xyzax=1:nax;
130     kind=1:ceil((nax+1)/2);
131     kax=(kind-1)*twopi/nax;
132     ki=rem(ceil(nax/lambda),max(kind))+1;
133     kix=rem(ceil(nax/lambdax),max(kind))+1;

134     Izffts=real(fft(Izsums(xyzax,:)));
135     Izdotffts=real(fft(Izdotsums(xyzax,:)));
136     Eintffts=real(fft(eintsums(xyzax,:)));

```

```

137     Eintdotffts=real(fft(eintdotsums(xyzax,:)));
138     Dmats=-Izdotffts(kind,)./Izffts(kind,)./ ...
139         ((kax(:).^2)*ones(size(Izsums(1,:))));
140     DEmats=-Eintdotffts(kind,)./Eintffts(kind,)./ ...
141         ((kax(:).^2)*ones(size(Izsums(1,:))));
142     Dks=Dmats(ki,:);
143     DEks=DEmats(kix,:);
144     Izamps=Izffts(kind,)*ampnorm;
145     Eintamps=Eintffts(kind,)*ampnorm;
146     Izders=Izdotffts(kind,)*ampnorm;
147     Eintders=Eintdotffts(kind,)*ampnorm;
148     Aks=Izamps(ki,:);
149     AEks=-Eintamps(kix,:);

150     %convert original data back to standard form
151     if ~isempty(protectaxes) & kdir<=3,
152         for iop=1:opnum,
153             if kdir==1,
154                 eval([deblank(namelist(iop,:)) 'sums=an_ztox(' ...
155                     deblank(namelist(iop,:)) ...
156                     'sums,dopsums(:,[2 3 4 1]));'])
157             elseif kdir==2,
158                 eval([deblank(namelist(iop,:)) 'sums=an_ztox(' ...
159                     deblank(namelist(iop,:)) ...
160                     'sums,dopsums(:,[3 4 1 2]));'])
161                 eval([deblank(namelist(iop,:)) 'sums=an_ztox(' ...
162                     deblank(namelist(iop,:)) ...
163                     'sums,dopsums(:,[2 3 4 1]);'])
164             elseif kdir==3,
165                 eval([deblank(namelist(iop,:)) 'sums=an_ytox(' ...
166                     deblank(namelist(iop,:)) ...
167                     'sums,dopsums(:,[4 1 2 3]);'])
168             end
169         end
170     end

171     %reshape diffusion constant and mean arrays
172     if kdir==1,
173         dopprocs=[1 2 3 4; 1 dopsums(2,2:4)];
174         Dks=reshape(Dks,nreps,nsig);
175         DEks=reshape(DEks,nreps,nsig);
176         Aks=reshape(Aks,nreps,nsig);
177         AEks=reshape(AEks,nreps,nsig);
178         for iop=1:opnum,
179             eval([deblank(namelist(iop,:)) 'means=reshape(' ...

```

```

180         deblank(namelist(iop,:)) 'means,nreps,nsig);'])
181     end
182 elseif kdir==2,
183     dopprocs=[2 3 4 1; 1 dopsums(2,[3 4 1])];
184     Dks=reshape(an_ztox(Dks,dopprocs),nreps,nsig);
185     DEks=reshape(an_ztox(DEks,dopprocs),nreps,nsig);
186     Aks=reshape(an_ztox(Aks,dopprocs),nreps,nsig);
187     AEks=reshape(an_ztox(AEks,dopprocs),nreps,nsig);
188     for iop=1:opnum,
189         eval([deblank(namelist(iop,:)) ...
190             'means=reshape(an_ztox(' ...
191             deblank(namelist(iop,:)) ...
192             'means,dopprocs),nreps,nsig);'])
193     end
194 elseif kdir==3,
195     dopprocs=[3 4 1 2; 1 dopsums(2,[4 1 2])];
196     [Dks,dopprocs]=an_ztox(Dks,dopprocs);
197     [DEks,dopprocs]=an_ztox(DEks,dopprocs);
198     [Aks,dopprocs]=an_ztox(Aks,dopprocs);
199     [AEks,dopprocs]=an_ztox(AEks,dopprocs);
200     Dks=reshape(an_ztox(Dks,dopprocs),nreps,nsig);
201     DEks=reshape(an_ztox(DEks,dopprocs),nreps,nsig);
202     Aks=reshape(an_ztox(Aks,dopprocs),nreps,nsig);
203     AEks=reshape(an_ztox(AEks,dopprocs),nreps,nsig);
204     for iop=1:opnum,
205         eval(['[' deblank(namelist(iop,:)) ...
206             'means,dopprocs]=an_ztox(' ...
207             deblank(namelist(iop,:)) ...
208             'means,dopprocs);'])
209         eval([deblank(namelist(iop,:)) ...
210             'means=reshape(an_ztox(' ...
211             deblank(namelist(iop,:)) ...
212             'means,dopprocs),nreps,nsig);'])
213     end
214 end

215 %evaluate amplitude and diffusion constant
216 %averages and statistics
217 Dk=mean(Dks);
218 DEk=mean(DEks);
219 Ak=mean(Aks);
220 AEk=mean(AEks);
221 Dkstd=std(Dks);
222 DEkstd=std(DEks);
223 Akstd=std(Aks);

```

```

224     AEkstd=std(AEks);

225     %average magnetization profiles, etc.
226     Izfft=real(fft(Izsum(xyzax,:)));
227     Izdotfft=real(fft(Izdotsum(xyzax,:)));
228     Eintfft=real(fft(eintsum(xyzax,:)));
229     Eintdotfft=real(fft(eintdotsum(xyzax,:)));
230     Dmat=-Izdotfft(kind,)./Izfft(kind,)./...
231         ((kax(:).^2)*ones(1,nsig));
232     DEmat=-Eintdotfft(kind,)./Eintfft(kind,)./...
233         ((kax(:).^2)*ones(1,nsig));
234     % Dkmean=Dmat(ki,:);           % NOT the same as Dk
235     % DEkmean=DEmat(kix,:);       %           DEk
236     Izamp=Izfft(kind,)*ampnorm;
237     Eintamp=Eintfft(kind,)*ampnorm;
238     Izder=Izdotfft(kind,)*ampnorm;
239     Eintder=Eintdotfft(kind,)*ampnorm;
240     % Akmean=Izamp(ki,:);         % equivalent to Ak
241     % AEkmean=Eintamp(kix,:);     %           AEK

242     end

243     %set title and axis strings
244     Nstring=[num2str(max(lsz*(~any(protectaxes==1)),1)) 'x' ...
245             num2str(max(msz*(~any(protectaxes==2)),1)) 'x' ...
246             num2str(max(nsz*(~any(protectaxes==3)),1))];
247     if isempty(protectaxes) | kdir>3,
248         repstring='';
249     else
250         repstring=[' ', num2str(nreps) ' samples'];
251     end
252     if jflag,
253         hamstring='Heisenberg Exchange Coupling';
254     elseif ~tflag,
255         hamstring='Zero-Field Dipole Coupling';
256     else
257         hamstring='Truncated Dipole Coupling';
258     end

259     tstring=[ostring ' lattice, N=' Nstring repstring ...
260             ', lambda=' num2str(lambda) ', ' hamstring];

261     if kdir <= 3
262         dstr=dirstring(kdir);
263     else

```

```

264     dstr=num2str(kdir);
265     end
266     dstring=['D' dstr ostring];
267     destring=['DE' dstr ostring];
268     lstring=['l = ' num2str(lambda)];

269 end

270 %plot data
271 if pflag,
272     clf
273     subplot(211)
274     if diffflag,
275         plot(xax,AEk)
276         ylabel('AE(t)')
277     else
278         plot(xax,Ak)
279         ylabel('A(t)')
280     end
281     grid
282     title(tstring)
283     xlabel('time')
284     subplot(212)
285     if diffflag,
286         plot(xax,DEk)
287         ylabel(destring)
288     else
289         plot(xax,Dk)
290         ylabel(dstring)
291     end
292     grid
293     xlabel('time')

294     contflag=1;
295     if contflag,
296         if ~isempty(protectaxes),
297             disp('Press any key to continue...')
298             pause
299             clf
300             subplot(211)
301             if diffflag,
302                 plot(xax,AEks)
303                 ylabel('AE(t)')
304             else
305                 plot(xax,Aks)

```

```

306     ylabel('A(t)')
307     end
308     grid
309     title(tstring)
310     xlabel('time')
311     subplot(212)
312     if diffflag,
313         plot(xax,DEks)
314         ylabel(destring)
315     else
316         plot(xax,Dks)
317         ylabel(dstring)
318     end
319     grid
320     xlabel('time')
321 end

322 disp('Press any key to continue...')
323 pause
324 clf
325 subplot(211)
326 plot(xax,eintmean)
327 grid
328 title(tstring)
329 xlabel('time')
330 ylabel('average internal energy')
331 subplot(212)
332 plot(xax,Izmean)
333 grid
334 xlabel('time')
335 ylabel('average z magnetization')

336 if ~isempty(protectaxes),
337     disp('Press any key to continue...')
338     pause
339     clf
340     subplot(211)
341     plot(xax,eintmeans)
342     grid
343     title(tstring)
344     xlabel('time')
345     ylabel('average internal energy')
346     subplot(212)
347     plot(xax,Izmeans)
348     grid

```

```

349     xlabel('time')
350     ylabel('average z magnetization')
351 end

352 disp('Press any key to continue...')
353 pause
354 clf
355 subplot(211)
356 if diffflag,
357     mesh(xax,fullax,eintsum);
358     xlabel(['Eint(' dstr ',t)'])
359 else
360     mesh(xax,fullax,Izsum);
361     xlabel(['Iz(' dstr ',t)'])
362 end
363 grid
364 xlabel('time')
365 ylabel([dstr ' position index'])
366 title(tstring)
367 subplot(212)
368 if diffflag,
369     mesh(xax,fullax,eintdotsum);
370     xlabel(['Eint(' dstr ',t) first derivative'])
371 else
372     mesh(xax,fullax,Izdotsum);
373     xlabel(['Iz(' dstr ',t) first derivative'])
374 end
375 grid
376 xlabel('time')
377 ylabel([dstr ' position index'])
378 disp('Press any key to continue...')
379 pause

380 clf
381 subplot(211)
382 if diffflag,
383     mesh(xax,kax,Eintamp)
384     xlabel('Eint(k,t)')
385 else
386     mesh(xax,kax,Izamp)
387     xlabel('Iz(k,t)')
388 end
389 grid
390 title(tstring)
391 xlabel('time')

```

```

392     ylabel('k')
393     subplot(212)
394     if diffflag,
395         mesh(xax,kax,Eintder)
396         zlabel('Eint(k,t) first derivatives')
397     else
398         mesh(xax,kax,Izder)
399         zlabel('Iz(k,t) first derivatives')
400     end
401     grid
402     xlabel('time')
403     ylabel('k')

404 end
405 end

406 procflag=0;

```

Throughout the code listed above, extensive use has been made of the A_FILES multidimensional matrix package developed for Matlab by G. Brunthaler. Since Matlab has no intrinsic method for representing arrays of dimension greater than two, such arrays are stored as extended 2D matrices according to a fixed ordering scheme. The contents file for the A_FILES package is included below, along with several sample routines. Source code (m-files) for the package may be found on the MIT Athena system in /mit/matlab/Matlab4.1/toolbox/contrib/Ndim or in /usr/people/dsodicks/matlab/Ndim on rumpleteazer.mit.edu.

```

1  % Files for n-dimensional arrays (A_FILES package, G. Brunthaler)
2  %
3  %
4  %           GENERAL Functions:
5  %
6  % a3_fft:      3D Fourier-Transformation (FT)
7  % a3_ifft:    inverse 3D Fourier-Transformation (FT)
8  % a3_zeros:   create empty 3D array
9  % a3_coord:   create 3D x-, y- and z-coordinate arrays
10 % a3_indi:    create 3D test array with indices as elements
11 %
12 % an_fft:     nD Fourier-Transformation (FT)
13 % an_ifft:    inverse nD Fourier-Transformation (FT)
14 % an_zeros:   create empty nD array
15 % an_indi:    create nD test array with indices as elements
16 %
17 %
18 %           SHIFT functions:

```

```

19 %
20 % a3_ztox: cyclic x-y-z permutation of elements for 3D array
21 % a3_ytox: anti-cyclic x-y-z permutation of elements " " "
22 % a3_xyper: permutation of x and y coordinates only
23 %
24 % an_ztox: cyclic x-y-z permutation of elements for nD array
25 % an_ytox: anti-cyclic x-y-z permutation of elements " " "
26 % an_xyper: permutation of x and y coordinates only
27 %
28 %
29 % INPUT functions:
30 %
31 % a3i_el: input of a single element into a 3D array
32 % a3i_vec: input of an x-vector into a 3D array
33 % a3i_mat: input of a 2D xy-matrix into a 3D array
34 %
35 % ani_el: input of a single element into an nD array
36 % ani_vec: input of an x-vector into an nD array
37 % ani_mat: input of a 2D xy-matrix into an nD array
38 %
39 %
40 % OUTPUT functions:
41 %
42 % a3o_el: output of a single element from a 3D array
43 % a3o_vec: output of an x-vector from a 3D array
44 % a3o_mat: output of a 2D xy-matrix from a 3D array
45 %
46 % ano_el: output of a single element from an nD array
47 % ano_vec: output of an x-vector from an nD array
48 % ano_mat: output of a 2D xy-matrix from an nD array
49 % ano_arr: output of an mD array from an nD array
50 %
51 %
52 % DKS additions:
53 %
54 % a3_ffts: 3D FFT with octant swapping (0 offset to center).
55 % a3_iffts: Inverse 3D FFT with octant swapping.
56 % a3_fftxyz: <=3D FFT (safety conditions when any dim=1).
57 % a3_ifftxyz: Inverse <=3D FFT with safety conditions.
58 % a3_fftsxyz: <=3D FFT with octant swapping .
59 % a3_ifftsxyz: Inverse <=3D FFT with octant swapping.
60 % a4_fftxyz: <=4D FFT (safety conditions for when any dim=1).
61 % a4_ifftxyz: Inverse <=4D FFT .
62 % a4_fftsxyz: <=4D FFT with octant swapping .
63 % a4_ifftsxyz: Inverse <=4D FFT with octant swapping.

```

```

64 % a3_to_m:      Convert 3D arrays in A_FILES format to matlab
65 %              meshgrid format.
66 % convolve3d:  <=3D convolution of two real arrays

1 function [C,dC]=an_ytox(A,dA),

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 % [C,dC]=an_ytox(A,dA),                                     %
4 % Array operation: anti-cyclic permutation of elements    %
5 %   in nD array! This function is ident to an_acyc.m.     %
6 % Input:                                                  %
7 %   A ... nD array                                        %
8 %   dA ... dimension matrix of array A                    %
9 % Output:                                                 %
10 %   C ... nD array                                       %
11 %   dC ... dimension matrix of new array C                %
12 % G.Brunthaler, 90-Sep-05                                 %
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

14 % assignments:
15 [t1,t2] = size(A);
16 [mA,dim] = size(dA);          % dim of A

17 % check array:
18 if t1 ~= dA(2) | t2 ~= prod(dA(2,:))/dA(2)
19     disp(['Size of array and content of dim. matrix ' ...
20         'are not consistent!']),
21     disp('Press any key to continue or <ctrl C> to break!'),
22     pause,
23 end

24 % do operation:
25 C=zeros(dA(4),prod(dA(2,:))/dA(4));
26 A=A';
27 C(:)=A;
28 dC=dA([2*[2:dim,1]-1;2*[2:dim,1]]);
29 return

1 function [FA,dFA] = a3_fftxyz(A,dA),

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 % [FA,dFA] = a3_fftxyz(A,dA),                                     %
4 % Array operation: 3-dim Fourier transf.                       %
5 %                                                                 %
6 % Input:                                                         %

```

```

7 %      A ... 3-dim array %
8 %      dA ... dimension vector of array A %
9 % Output: %
10 %      FA ... 3-dim Fourier transform of A %
11 %      dFA ... dimension vector of new array FA (equal to A)%
12 % %
13 % G.Brunthaler, 89-Nov-04 %
14 % %
15 % Adapted from a3_fft.m %
16 % DKS 10/29/93: safety conditions for when any dimension = 1 %
17 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

18 if dA(2,1)==1,
19     [A,dA]=a3_ztox(A,dA);
20 else
21     [A,dA]=a3_ztox(fft(A),dA);
22 end
23 if dA(2,1)==1,
24     [A,dA]=a3_ztox(A,dA);
25 else
26     [A,dA]=a3_ztox(fft(A),dA);
27 end
28 if dA(2,1)==1,
29     [FA,dFA]=a3_ztox(A,dA);
30 else
31     [FA,dFA]=a3_ztox(fft(A),dA);
32 end

33 return,

1 function [IFA,dIFA] = a3_ifftxyz(A,dA),

2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 % [IFA,dIFA] = a3_ifftxyz(A,dA), %
4 % Array operation: inverse 3-dim Fourier transf. with %
5 %      octant swapping      (inverse of a3_fftxyz) %
6 % %
7 % Input: %
8 %      A ... 3-dim array %
9 %      dA ... dimension vector of array A, with dA=[xA,yA,zA] %
10 % Output: %
11 %      FA ... inverse 3-dim Fourier transform of A %
12 %      dFA ... dimension vector of new array FA (equal to A) %
13 % %
14 % G.Brunthaler, 89-Nov-04 %

```

```

15 %
16 % Adapted from a3_ifft.m
17 % DKS 10/29/93: safety conditions for when any dimension = 1
18 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

19 % check array:
20 [n,m]=size(A);
21 if n ~= dA(2) | m ~= prod(dA(2,:))/dA(2)
22     disp(['Size of array and content of dimension vector ' ...
23         'are not consistent!']),
24     disp(['Press any key to continue or <ctrl C> ' ...
25         'to break the program!']),
26     pause,
27 end

28 % do operation:
29 % IFA=fft(a_acyc(fft(a_acyc(fft( ...
30 %     conj(a_acyc(A, [1,m,n]))), [m,n,1])), [n,1,m]));
31 % IFA=conj(IFA)/l/m/n;

32 [A,dA]=a3_ytox(conj(A),dA);
33 if dA(2,1)==1,
34     [A,dA]=a3_ytox(A,dA);
35 else
36     [A,dA]=a3_ytox(fft(A),dA);
37 end
38 if dA(2,1)==1,
39     [A,dIFA]=a3_ytox(A,dA);
40 else
41     [A,dIFA]=a3_ytox(fft(A),dA);
42 end
43 if dIFA(2,1)==1,
44     IFA=conj(A)/dA(2)/dA(4)/dA(6);
45 else
46     IFA=conj(fft(A))/dA(2)/dA(4)/dA(6);
47 end

48 return,

```

Finally, the code used for calculating Redfield-Yu diffusion constants and fitting parameters is included below:

```

1 function [chi1,chi2conv,chi2full,Dspin]=redfieldspin(...
2     loopflag,spins)

```

```

3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 %                                REDFIELDSPIN.M                                %
5 %                                                                            %
6 % Calculate Redfield-Yu spin diffusion constant for a vector of %
7 % spin quantum numbers. Diffusion constant at concentration c=1 %
8 % based on Redfield & Yu's moment arguments: D=chi1/sqrt(1+chi2) %
9 %                                                                            %
10 % Global variables used:                                                  %
11 %     lsz,msz,nsz: x,y,z dimensions of spin lattice                       %
12 %     lunit,munit,nunit: unit cell x,y,z lengths                          %
13 %     alpha,beta,gamma: unit cell angles                                  %
14 %     isz,jsz,ksz: x,y,z dimensions of interaction region                 %
15 %     qtheta,qphi: lattice orientation angles                             %
16 %                                                                            %
17 % Input:                                                                    %
18 %     loopflag: 1 --> do full loop calculation for chi2full               %
19 %               0 --> omit full calculation: use only approx.            %
20 %               convolution formula for chi2conv                           %
21 %               (chi2full=0)                                               %
22 %     spins: vector of spin quantum numbers                               %
23 %                                                                            %
24 % Output:                                                                    %
25 %     chi1: first nonlinear fit parameter                                  %
26 %     chi2conv: convolution approximation to 2nd fit param.                %
27 %     chi2full: exact evaluation of second fit parameter                  %
28 %     Dspin: vector of diffusion constants vs spin quantum #              %
29 %                                                                            %
30 % DKS 3/7/94                                                                %
31 %                                                                            %
32 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

33 t0=clock;

34 %global variable declarations
35 % (several statements to stay under max line length)
36 global lsz msz nsz lunit munit nunit ...
37     alpha beta gamma isz jsz ksz ...
38     rfact qtheta qphi ostring ...
39     cset ...
40     twopi vbis wbis v_iso w_iso ...
41     dw nsig;
42 global pflag dflag aterm bterm corrflag setflag ...
43     jflag tflag diffflag convflag swapflag;
44 global nderivmax nderivmin nnorm ...
45     err0 terr scalefac ...

```

```

46     enint c0 ...
47     lambda lambdax kdir Mamp;
48     global protectaxes;

49     %initialize variables
50     [Cmat,dCmat]=a3_zeros([1 2 3; lsz msz nsz]);
51     Xmat=Cmat; dXmat=dCmat;

52     %make accurate magic angle
53     if qtheta < 54.8 & qtheta > 54.6,
54         qtheta=atan(sqrt(2))*180/acos(-1);
55     end
56     %initialize constants
57     drfac=pi/180;
58     sinalpha=sin(alpha*drfac);
59     sinbeta=sin(beta*drfac);
60     singamma=sin(gamma*drfac);
61     if qtheta ~= 0,
62         qcttheta=cos(qtheta*drfac);
63         qsttheta=sin(qtheta*drfac);
64         qcphi=cos(qphi*drfac);
65         qsphi=sin(qphi*drfac);
66     end

67     %choice of center location in lattice
68     lcell=floor(lsz/2)+1;
69     mcell=floor(msz/2)+1;
70     ncell=floor(nsz/2)+1;
71     icell=floor((isz-1)/2);
72     jcell=floor((jsz-1)/2);
73     kcell=floor((ksz-1)/2);
74     iend=1-rem(isz,2);
75     jend=1-rem(jsz,2);
76     kend=1-rem(ksz,2);

77     %calculate Cmat
78     disp('making coupling matrix...')
79     nc=[-(kcell+kend); 1 ; kcell];
80     mc=[-(jcell+jend); 1 ; jcell];
81     lc=[-(icell+iend); 1 ; icell];
82     [C,dC]=a3_coord([lc mc nc]);
83     Xc=lunit*sinalpha*ano_arr(C,dC,1);
84     Yc=munit*sinbeta*singamma*ano_arr(C,dC,2);
85     Zc=nunit*ano_arr(C,dC,3);
86     R2c=Xc.*Xc+Yc.*Yc+Zc.*Zc;

```

```

87 Rc=sqrt(R2c);
88 [iind,jind]=find(abs(Rc)<rfact);
89 Rc(iind,jind)=Rc(iind,jind)+Inf;
90 Costhc=Zc./Rc;
91 if abs(qtheta)>eps,
92     Costhc=((Xc*qcphi+Yc*qsphi)*qsttheta+Zc*qcttheta)./Rc;
93     %^coordinate transform for lattices other than [001]
94 end

95 %form a coupling matrix "centered" at (lcell,mcell,ncell)
96 lind=lcell+(-(icell+iend):icell);
97 mnind=kron(ones(1,2*kcell+1+kend),mcell+(-(jcell+jend):jcell));
98 nmind=kron(ncell+(-(kcell+kend):kcell),ones(1,2*jcell+1+jend))-1;

99 Cmat(lind,nmind*msz+mnind)=-(1-3*Costhc.*Costhc)./(Rc.^3)/2;
100 Xmat(lind,nmind*msz+mnind)=Xc;

101 %rearrange indices so that coupling matrix is "centered"
102 %at (1,1,1) for correct convolution with spin vector array
103 lind=[lcell:lsz 1:(lcell-1)];
104 mnind=kron(ones(1,nsz),[mcell:msz 1:(mcell-1)]);
105 nmind=kron([ncell:nsz 1:(ncell-1)],ones(1,msz))-1;

106 Cmat=Cmat(lind,nmind*msz+mnind);
107 Xmat=Xmat(lind,nmind*msz+mnind);

108 foursum=sum(sum(Xmat.*Xmat.*(Cmat.^4)));
109 foursum=foursum*(1-(7/16)./(spins.*(spins+1)));

110 disp('calculating chi1...')
111 chi1=sqrt((5*pi*sum(sum(Xmat.*Xmat.*Cmat.*Cmat))^3)./foursum)/24;
112 disp(['      chi1=' num2str(chi1)])

113 disp('calculating chi2...')
114 chi2conv=((5/48)./foursum)*sum(sum(...
115     13*convolve3d(Xmat.*Xmat.*Cmat.*Cmat,Cmat.*Cmat,dCmat)-...
116     6*convolve3d(Xmat.*Cmat.*Cmat,Xmat.*Cmat.*Cmat,dCmat)-...
117     3*convolve3d(Cmat.*Cmat,Xmat.*Xmat.*Cmat.*Cmat,dCmat) ))...
118     -5/3;
119 disp(['      chi2=' num2str(chi2conv) ...
120 ' (convolution approximation)'])

121 chi2full=0;
122 if loopflag,
123     for il=1:lsz,

```

```

124     disp(['il=' num2str(il)])
125     ldiff=1-il;
126     lind=cshift(1:lsz,ldiff);
127     for im=1:msz,
128         disp([' im=' num2str(im)])
129         mdiff=1-im;
130         mind=cshift(1:msz,mdiff);
131         mnind=kron(ones(1,nsz),mind);
132         for in=1:nsz,
133             ndiff=1-in;
134             nind=cshift(1:nsz,ndiff);
135             nmind=kron(nind,ones(1,msz))-1;

136         Cmshft=Cmat(lind,nmind*msz+mnind);

137         lama=-3*Cmshft.*Cmshft*Cmshft(1,1)*Cmshft(1,1);
138         lamb= 4*Cmat.*Cmshft*Cmshft(1,1).*...
139             (Cmshft(1,1)+Cmshft);
140         lamc= 8*Cmat.*Cmat.*(Cmshft(1,1)-Cmshft).^2;
141         chi2full=chi2full+...
142             sum(sum(Xmat.*Xmat.*(lama+lamb+lamc)));
143     end
144 end
145 end

146 chi2full=((5/48)*chi2full./foursun) - 5/3;

147 disp(' ')
148 disp([' chi2=' num2str(chi2full) ' (full calculation)'])

149 Dspin=chi1./sqrt(1+chi2full);

150 else
151     Dspin=chi1./sqrt(1+chi2conv);
152 end

153 disp('')
154 telapsed=etime(clock,t0);
155 disp(['Elapsed time ' num2str(telapsed) ' seconds'])

```

Bibliography

- [1] D.K. Sodickson, M.H. Levitt, S. Vega, and R.G. Griffin. Broad band dipolar recoupling in the nuclear magnetic resonance of rotating solids. *Journal of Chemical Physics*, 98(9):6742, May 1993.
- [2] K. Wüthrich. *NMR of Proteins and Nucleic Acids*. Wiley, 1986.
- [3] D. Suter and R.R. Ernst. Spin diffusion in resolved solid-state nmr spectra. *Physical Review B*, 32(9):5608, November 1985.
- [4] D. Suter and R.R. Ernst. Spectral spin diffusion in the presence of an extraneous dipolar reservoir. *Physical Review B*, 25(9):6038, May 1982.
- [5] M.H. Levitt, D.P. Raleigh, F. Creuzet, and R.G. Griffin. Theory and simulations of homonuclear spin pair systems in rotating solids. *Journal of Chemical Physics*, 92(11):6347, June 1990. Contains an extensive list of references to historical and contemporary work on rotational resonance.
- [6] D.P. Raleigh, M.H. Levitt, and R.G. Griffin. Rotational resonance in solid state nmr. *Chemical Physics Letters*, 146(1.2):71, April 1988.
- [7] R. Tycko and G. Dabbagh. Measurement of nuclear magnetic dipole-dipole couplings in magic angle spinning nmr. *Chemical Physics Letters*, 173(5,6):461, October 1990.
- [8] S. Vega. Presentation at the 32nd Experimental NMR Conference, Asilomar, CA, April 1991.

- [9] T. Gullion and S. Vega. A simple magic angle spinning nmr experiment for the dephasing of rotational echoes of dipolar coupled homonuclear spin pairs. *Chemical Physics Letters*, 194(4-6):423, 1992.
- [10] P. Caravatti, G. Bodenhausen, and R.R. Ernst. Selective pulse experiments in high-resolution solid-state nmr. *Journal of Magnetic Resonance*, 55:88, 1983.
- [11] T. Gullion, D. B. Baker, and M. S. Conradi. New, compensated carr-purcell sequences. *Journal of Magnetic Resonance*, 89:479, 1990.
- [12] J.H. Ok, R.G.S. Spencer, A.E. Bennett, and R.G. Griffin. Homonuclear correlation spectroscopy in rotating solids. *Chemical Physics Letters*, 197(4-5):389, 1992.
- [13] A. Bennett, J.H. Ok, S. Vega, and R.G. Griffin. Chemical shift correlation spectroscopy in rotating solids: Radio frequency-driven dipolar recoupling and longitudinal exchange. *Journal of Chemical Physics*, 96(11):8624, June 1992.
- [14] S. Vega. Fictitious spin 1/2 operator formalism for multiple quantum nmr. *Journal of Chemical Physics*, 68(12):5518, June 1978.
- [15] A. Wokaun and R.R. Ernst. Selective excitation and detection in multilevel spin systems: Application of single transition operators. *Journal of Chemical Physics*, 67(4):1752, August 1977.
- [16] M.H. Levitt. Heteronuclear cross polarization in liquid-state nuclear magnetic resonance: Mismatch compensation and relaxation behavior. *Journal of Chemical Physics*, 94(1):30, January 1991.
- [17] L. Braunschweiler and R.R. Ernst. Coherence transfer by isotropic mixing: Application to proton correlation spectroscopy. *Journal of Magnetic Resonance*, 53:521, 1983.
- [18] T. Gullion and J. Schaefer. Rotational-echo double-resonance nmr. *Journal of Magnetic Resonance*, 81(1):196, 1989.

- [19] D.P. Raleigh, F. Creuzet, S.K. Das Gupta, M.H. Levitt, and R.G. Griffin. Measurement of internuclear distances in polycrystalline solids: Rotationally enhanced transfer of nuclear spin magnetization. *Journal of the American Chemical Society*, 111:4502, 1989.
- [20] A.E. McDermott, F. Creuzet, R.G. Griffin, L.E. Zawadzke, Q. Ye, and C.T. Walsh. Rotational resonance determination of the structure of an enzyme-inhibitor complex: Phosphorylation of an (aminoalkyl)phosphinate inhibitor of d-alanyl-d-alanine ligase by atp. *Biochemistry*, 29(24):5767, 1990.
- [21] F. Creuzet, A. McDermott, R. Gebhard, K. van der Hoef, M.B. Spijker-Assink, J. Herzfeld, J. Lugtenberg, M.H. Levitt, and R.G. Griffin. Determination of membrane protein structure by rotational resonance nmr: Bacteriorhodopsin. *Science*, 251:783, February 1991.
- [22] L.K. Thompson, A.E. McDermott, J. Raap, C.M. van der Weilen, J. Lugtenburg, J. Herzfeld, and R.G. Griffin. Rotational resonance nmr study of the active-site structure in bacteriorhodopsin – conformation of the schiff-base linkage. *Biochemistry*, 31(34):7931, 1992.
- [23] R.G.S. Spencer, K.J. Halverson, M. Auger, A.E. McDermott, R.G. Griffin, and P.T. Lansbury. An unusual peptide conformation may precipitate amyloid formation in alzheimer’s disease: Application of solid-state nmr to the determination of protein secondary structure. *Biochemistry*, 30(43):10382, 1991.
- [24] N. Bloembergen. On the interaction of nuclear spins in a crystalline lattice. *Physica (Utrecht)*, 15(3-4):386, May 1949.
- [25] A.G. Redfield. Spatial diffusion of spin energy. *Physical Review*, 116(2):315, October 1959.
- [26] A.G. Redfield and W.N. Yu. Moment method calculation of magnetization and interspin-energy diffusion. *Physical Review*, 169(2):443, May 1968.

- [27] A.G. Redfield and W.N. Yu. Erratum: Moment method calculation of magnetization and interspin-energy diffusion. *Physical Review*, 177:1018, 1969.
- [28] I.J. Lowe and S. Gade. Density-matrix derivation of the spin-diffusion equation. *Physical Review*, 156(3):817, April 1967.
- [29] I.J. Lowe and S. Gade. Erratum: Density-matrix derivation of the spin-diffusion equation. *Physical Review*, 166:934, 1968.
- [30] P. Borckmans and D. Walgraef. Irreversibility in paramagnetic spin systems: Free induction decay and spin diffusion. *Physical Review*, 167(2):282, March 1968.
- [31] T. Morita. Spin diffusion in the heisenberg magnets at infinite temperature. *Physical Review B*, 6(9):3385, November 1972.
- [32] G.R. Khutsishvili. *Proc. Inst. Acad. Sci. Georgia (USSR)*, 4:3, 1956.
- [33] P.G. de Gennes. Sur la relaxation nucleaire dans les cristaux ioniques. *Journal of Physics and Chemistry of Solids*, 7:345, 1958.
- [34] W.E. Blumberg. Nuclear spin-lattice relaxation caused by paramagnetic impurities. *Physical Review*, 119(1):79, July 1960.
- [35] E.R. Andrew, K.M. Swanson, and B.R. Williams. Angular dependence of nuclear spin-lattice relaxation time for several alkali halide crystals. *Proceedings of the Physical Society*, 77:36, 1961.
- [36] H.E. Rorschach. Nuclear relaxation in solids by diffusion to paramagnetic impurities. *Physica*, 30:38, 1964.
- [37] A.G. Akhmedov and R.A. Dautov. Relaxation and spin diffusion of F^{19} nuclei in CaF_2 . *Soviet Physics - Solid State*, 6(2):415, August 1964.
- [38] S.M. Day, E. Otsuka, and B. Josephson. Spin-lattice relaxation of F^{19} in CaF_2 at low temperatures. *Physical Review*, 137(1A):108, January 1965.

- [39] G.W. Leppelmeier and J. Jeener. Measurement of the nuclear spin diffusion coefficient in CaF_2 . *Physical Review*, 175(2):498, November 1968.
- [40] Changguo Tang. *Computational Dynamics of Classical Nuclear Spins in Solids*. PhD dissertation, Massachusetts Institute of Technology, Department of Chemistry, September 1990.
- [41] Changguo Tang and John S. Waugh. Dynamics of classical spins on a lattice: Spin diffusion. *Physical Review B*, 45(2):748, January 1992.
- [42] Irmgard Nolden. unpublished, 1993.
- [43] J.H. Van Vleck. The dipolar broadening of magnetic resonance lines in crystals. *Physical Review*, 74(9):1168, November 1948.
- [44] C. Kittel and E. Abrahams. Dipolar broadening of magnetic resonance lines in magnetically diluted crystals. *Physical Review*, 90(2):238, April 1953.
- [45] Dietrich Stauffer. *Introduction to Percolation Theory*. Taylor and Francis, 1985.
- [46] J.M. Radcliffe. Some properties of coherent spin states. *Journal of Physics A: General Physics*, 4:313, 1971.
- [47] F.T. Arecchi, E. Courtens, R. Gilmore, and H. Thomas. Atomic coherent states in quantum optics. *Physical Review*, A6:2211, 1972.
- [48] J.R. Klauder and B.S. Skagerstam. *Coherent States*. World Scientific, 1985.
- [49] A. Perelomov. *Generalized Coherent States and Their Applications*. Springer-Verlag, 1986.
- [50] R.J. Glauber. Photon correlations. *Physical Review Letters*, 10:84, 1963.
- [51] R.J. Glauber. The quantum theory of optical coherence. *Physical Review*, 130:2529, 1963.
- [52] R.J. Glauber. Coherent and incoherent states of the radiation field. *Physical Review*, 131:2766, 1963.

- [53] W.M. Zhang, D.H. Feng, and R. Gilmore. Coherent states: Theory and some applications. *Reviews of Modern Physics*, 62(4):867, October 1990.
- [54] Y. Takahashi and F. Shibata. Spin coherent state representation in non-equilibrium statistical mechanics. *Journal of the Physical Society of Japan*, 38(3):656, March 1975.
- [55] Y. Takahashi and F. Shibata. Generalized phase space method in spin systems – spin coherent state representation. *Journal of Statistical Physics*, 14(1):49, 1976.
- [56] E.H. Lieb. The classical limit of quantum spin systems. *Communications in Mathematical Physics*, 31:327, 1973.
- [57] D.V. Kapur, M.J. Skrinjar, and S.D. Stojanovic. Relation between spin-coherent states and boson-coherent states in the theory of magnetism. *Physical Review*, B44(5):2227, August 1991.
- [58] J.R. Klauder. Path integrals and stationary-phase approximations. *Physical Review D*, 19(8):2349, April 1979.
- [59] H. Kuratsuji and T. Suzuki. Path integral in the representation of $su(2)$ coherent state and classical dynamics in a generalized phase space. *Journal of Mathematical Physics*, 21(3):472, March 1980.
- [60] H. Takano. Monte carlo method for quantum spin systems based on the bloch coherent state representation. *Progress of Theoretical Physics*, 73(2):332, February 1985.
- [61] H. Takano. Monte carlo method for quantum spin systems with use of the bloch coherent state representation. In M. Suzuki, editor, *Quantum Monte Carlo Methods in Equilibrium and Nonequilibrium Systems*, page 144. Springer-Verlag, 1987.
- [62] M. Bergeron. Coherent state path integral for the harmonic oscillator and a spin particle in a constant magnetic field. *Fortschritte der Physik*, 40(2):119, March 1992.

- [63] M.V. Berry. Quantal phase factors accompanying adiabatic changes. *Proceedings of the Royal Society of London A*, 392:45, 1984.
- [64] J. Anandan and L. Stodolsky. Some geometrical considerations of berry's phase. *Physical Review D*, 35(8):2597, April 1987.
- [65] S.K. Bose and B. Dutta-Roy. Geometry of quantum evolution and the coherent state. *Physical Review A*, 43(7):3217, April 1991.
- [66] Y. Aharonov and J. Anandan. Phase change during a cyclic quantum evolution. *Physical Review Letters*, 58(16):1593, April 1987.
- [67] D. Suter, G.C. Chingas, R.A. Harris, and A. Pines. Berry's phase in magnetic resonance. *Molecular Physics*, 61(6):1327, 1987.
- [68] J. Anandan and Y. Aharonov. Geometry of quantum evolution. *Physical Review Letters*, 65(14):1697, October 1990.

Dawn slopes through doorways and slips into the corners of the room,
Sunlight piled in drifts along its path.
Out of indistinctness one by one
The planes and edges of a chair are made,
The pattern of a rug, a mantelpiece, a windowshade.
Atoms of tumultuous dust
Congeal into a corridor,
And a creaking fanfare of floorboards
Lifts the veil of shadow from yesterday's bread.
What awakening was enacted here?
What revelation? What experiment?
The day spins resolutely on its axis
While an ancient revolution hurls us inward toward the sun.