Institut für Organische Chemie und Biochemie
der Technischen Universität München

Anwendung der Steuerungstheorie auf die kernmagnetische Resonanzspektroskopie

Von der Entwicklung computergestützter Optimierungsmethoden bis zur experimentellen Umsetzung

Timo O. Reiss

Vollständiger Abdruck der von der Fakultät für Chemie der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. H. Kessler
Prüfer der Dissertation:	1. Univ.-Prof. Dr. St. J. Glaser
	2. Univ. Prof. Dr. F. H. Köhler

Die Dissertation wurde am 20.11.2002 bei der

Technischen Universität München

eingereicht und durch die

Fakultät für Chemie

am 22.01.2003 angenommen.

when I was 4 years old
they tried to test my i.q.
they showed me this picture
of 3 oranges and a pear
they asked me
which one is different
and does not belong?
they taught me different is wrong.

Ani D.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Die kernmagnetische Resonanzspektroskopie (NMR-Spektroskopie, *Nuclear Magnetic Resonance*) hat seit ihrer Entdeckung [1, 2] eine Entwicklung vollzogen, die nur wenigen spektroskopischen Methoden beschieden war. Inwischen reicht ihre Anwendung von der Routinespektroskopie über Anwendungen in der Medizin (Imaging [3]) bis hin zur Untersuchung komplexer chemischer Systeme (*in situ* Spektroskopie [4], Strukturaufklärung biochemischer Systeme [5]). Darüberhinaus eignet sich die Methode sehr gut, um quantenmechanische Modelle zu prüfen, z.B. aus der Quanten-Informationsverarbeitung (*Quantum Computing* [6]).

Die Vielseitigkeit der Methode führte dazu, daß die verfügbare Hardware stetig verbessert wurde. Doch Spektrometer-Hardware trägt nur teilweise zum Erfolg eines Experimentes bei. Daher gab es zusätzlich eine starke Enwicklung auf dem Sektor der Pulsprogramm-Entwicklung. Inzwischen gibt es eine Reihe von wichtigen Standard-Experimenten, welche aus dem spektroskopischen Alltag nicht mehr wegzudenken sind. Die Entwicklung des Produktoperator-Formalismus [7, 8] eröffnete die Möglichkeit, unter vereinfachten Annahmen vorhandene Pulssequenzen zu analysieren bzw. neue Experimente zu entwickeln. Allerdings vernachlässigen unter solch vereinfachten Bedingungen entwickelte Experimente bestimmte quantenmechanische Effekte. So stellt sich stets die Frage, ob eine vorhandene Implementierung eines Experimentes die best mögliche ist, d.h. ob die durch die Quantenmechanik gegebene Grenze des erwünschten Transfers in der kürzest möglichen Zeit erreicht wird. Kenntnis über diese sog. unitäre Grenze eines Experimentes ist insofern wichtig, weil (nahezu) maximaler Kohärenztransfer z.B. eine

Verkürzung der benötigten Gesamtmeßzeit durch besseres Signal/Rausch-Verhältnis ermöglicht. Ebenso wichtig ist es zu wissen, ob man den gewünschten Kohärenz-Transfer in möglichst kurzer Zeit erreicht, weil die spektrale Information eines Experimentes exponentiell zu seiner Dauer durch Relaxation verringert wird.

Neben der Information über die Zeitoptimalität eines Experimentes ist es ebenso wünschenswert, NMR-Experimente unter Berücksichtigung inhärenter quantenmechanischer Effekte zu entwickeln. Zu diesem Effekten zählen z.B. Resonanzfrequenz-Offsets oder der dirkete Einschluß von Relaxation. Die Berücksichtigung von Resonanzfrequenz-Offsets ist wichtig, weil man durch eine größere Bandbreite eines Experimentes die generelle Anwendbarkeit einer Pulssequenz erhöht. Kernmagentische Relaxation ist ein Effekt, welcher während der analytischen Entwicklung von NMR-Experimenten schwierig zu handhaben ist. Zum einen eröffnet dieser quantenmechanische Effekt neue Transfer-Wege (durch Kreuzrelaxation und kreuzkorrellierte Relaxation), welche durch Standard-Betrachtungsweisen unberücksichtigt bleiben, bzw. verschlechtert die Eigenschaften bestimmter Transfer-Wege dramatisch, wenn bestimmte schnell-relaxierende Operatoren erzeugt werden. Inzwischen versucht man Experimente zu entwickeln, welche bestimmte Relaxations-Pfade berücksichtigen (z.B. TROSY [9], crINEPT [10]), jedoch bleibt auch hier die Frage der Optimalität weitgehend unbeantwortet. Führt man sich die Komplexität der zu berücksichtigenden Effekte vor Augen, so ist es nicht verwunderlich, daß vermehrt numerische Computer-Methoden in der NMR eingesetzt werden, um optimierte Experimente zu entwickeln. Grundlage der meisten hierbei verwendeten Optimierungs-Verfahren sind Kombinationen aus statistischen und auf Gradienten-Methoden beruhenden Algorithmen.

Das Ziel, Verfahren zu entwickeln, welche zu optimalen Lösungen verschiedener Probleme führen, beschäftigt die Wissenschaft seit längerem. Bereits um 1910 beschäftigte sich der dänische Mathematiker Erlang mit der Modellierung des Auslastungsgrades von Telefonnetzen, um die Ausbaukosten des Netzes unter bestimmten Anforderungen zu minimieren. Kurz darauf verwendete Edison die Methode des *tactical game board* um verschiedene Routen für Handelsschiffe zu bewerten. Aus diesen Anfängen entwickelte sich ein eigenständiges wissenschaftliches Feld, welches das Problem der Optimierung immer besser zu beschreiben vermochte.

Jahr	Entdeckung
1868	Stabilitäts-Anlyse des Zentrifugal-Reglers nach Watt (Maxwell)
1877	Stabilitäts-Kriterium (Routh)
1890	Nichtlineare Stabilität (Lyapunov)
1910	Gyroskop und Autopilot (Sperry)
1927	Rückkopplung elektronischer Verstärker (Black)
	Differentialmesser (Bush)
1932	Stabilitäts Kriterium (Nyquist)
1938	Frequenzgang-Methoden (Bode)
1942	Design optimaler Filter (Wiener)
1947	Abtastdaten-Systeme (Hurewicz)
	Nichols-Diagramm (Nichols)
1948	Wurzelortskurve (Evans)
1950	Nichtlineare Analyse (Kochenburger)
1956	Maximum Prinzip (Pontryagin)
1957	Dynamische Programmierung (Bellman)
1960	Trägheits-Navigation (Draper)
	Optimale Bewertung (Kalman)
1969	Mikroprozessor (Hoff)

TABELLE 1.1: Historische Entwicklung der Steuerung-Technik [11]. Stets sind wichtige mechanische Erfindungen die Triebfeder der Weiterentwicklung der Steuerung-Theorie gewesen. So war es zu Mitte des 19. Jahrhunderts die Dampfmaschine und im 20. Jahrhundert die Luft- und Raumfahrt, die die mathematische Formulierung der Steurungs-Theorie vorantrieb.

Ein Teil dieses Feldes setzt sich mit der optimalen automatisierten Steuerung von Systemen finiter Freiheitsgrade auseinander, die sog. optimale Steuerung dynamischer Systeme. Wie man Tablle 1.1 entnehmen kann, folgte die theoretische Entwicklung der Steuerungs-Theorie stets der industriellen Entwicklung. Hierbei war es zunächst Mitte des 19. Jahrhunderts die Dampfmaschine, welche die Entwicklung automatisierter mechanischer Steuerungssysteme vorantrieb[1]. Zu Beginn des 20. Jahrhunderts wurden durch die Einführung der motorisierten Luftfahrt neue Anwedungsbereiche für Steuerungssysteme erschlossen. In der Mitte des 20. Jahrhunderts waren es vor allem die sowjetischen und amerikanischen Weltraumprogramme, welche die

[1]In diesem Zusammen ist interessant, daß die Dampfmaschine von Watt um 1790 die Dampfmaschinen Newcomens fast vollständig verdrängt hatte, weil letzteren u.a. die automatisierte Kontrollmöglichkeit fehlten. Dieser Zentrifugal-Regler war zunächst ein empirisch entwickeltes Kontrollsystem. Es dauerte fast 80 Jahre bis dieses System mathematisch untersucht wurde.

Steuerungs-Theorie zu neuen Erkenntnissen anspornte.

Betrachtet man nun die in der NMR verwendeten Systeme, so stellt man fest, daß eine Pulssequenz auch als Formulierung der Kontrollgesetze für die Bewegung eines Spinsystems aufgefaßt werden kann. Unter diesem Gesichtspunkt stellte sich nun die Frage, ob man die Steuerungstheorie auch zur Erzeugung solcher Kontrollgesetzte, d.h. Pulssequenzen verwenden kann. Die Bewegungsgleichungen eines Spinsystems sind für die NMR bekannt, somit verblieb die Formulierung der entsprechenden steuerungstheoretischen Gesetze als Voraussetzung, um NMR-Experimente innerhalb dieses Rahmenwerkes formulieren zu können.

Diese Arbeit widmet sich der Anwendung der Steuerungstheorie auf Fragestellungen der NMR-Spektroskopie, speziell der Anwendung des Pontryaginschen Maximum-Prinzips. In Teil I (Theorie) werden die für diese Arbeit notwendigen theoretischen Prinzipien und Bergriffe erläutert werden. In Teil II (Numerik) werden Anwendungen auf numerisch untersuchte Fragestellungen unter Verwendung der im Rahmen dieser Arbeit entwickelten Optimierungs-Software vorgestellt werden. Hierbei wird der Einfluß von Störtermen graduell erhöht werden. In Kapitel 5 werde Optimierungen unter idealen Bedingungen beschrieben. In Kapitel 6 werden Optimierungen unter Einschluß von Relaxation diskutiert, während die in Kapitel 7 vorgestellte Optimierung unter Berücksichtung von sowohl B_1-Inhomogenität als auch Offset-Effekten durchgeführt worden sind. In Teil III (Experimente) werden an Modell-Systemen vollzogenen NMR-Experimente besprochen werden, welche auf Prinzipien der Steuerungsthorie beruhen. In Teil IV (Software) wird die auf den Prinzipien der Steuerungstheorie beruhende Optimierung-Software vorgestellt werden.

Teil I

Theorie

Kapitel 2

Dichtematrix Theorie

Die Dynamik von Kernspinsystemen kann durch einen quantenmechanischen Formalismus beschrieben werden, welcher als Dichtematrixtheorie bezeichnet wird [12]. Die Dichtematrixtheorie ermöglicht eine vollständige Beschreibung eines Spinsystems. Im Gegensatz zum klassischen Bloch-Modell [13], welches die Gesamtmagnetisierung des Spinsystems vektoriell erfaßt und deren Entwicklung anhand dieses Vektors darstellt, kann man durch die Entwicklung der Dichtematrix alle Informationen über das Spinsystem zu jedem Zeitpunkt eines beliebigen NMR-Experimentes erhalten.

Nach einer Darstellung der quantenmechanischen Grundlagen soll in diesem Kapitel die Dichtematrix-Theorie und deren Anwendung auf das Phänomen der Kernresonanz dargestellt werden [14].

2.1 Grundlagen der Quantenmechanik

Bevor die Dichtematrix eingeführt werden kann, sollen einige grundlegende Postulate und Schreibweisen der Quantenmechanik [15, 16] eingeführt werden, welche für die darauffolgende Darstellung der Theorie wichtig sind.

2.1.1 Die Schrödinger-Gleichung

Generell wird die zeitliche Entwicklung eines quantenmechanischen Systems durch die zeitabhängige Schrödinger-Gleichung beschrieben:

$$i\hbar \frac{d\Psi(t)}{dt} = \hat{\mathcal{H}}\Psi(t). \tag{2.1}$$

Der Hamilton-Operator $\hat{\mathcal{H}}$ enthält die physikalischen Vorgaben, die die zeitliche Entwicklung eines Spinsystems bestimmen. Er kann hierbei zeitabhängig

oder zeitunabhängig sein. Die Lösung der Schrödinger-Gleichung bezeichnet man als Wellenfunktion des Systems, $\Psi(t)$; sie enhält alle möglichen Informationen über den Zustand des Systems. Die Wahrscheinlichkeit, daß sich ein System zum Zeitpunkt t im Zustand $\Psi(t)$ befindet, wird durch die Wahrscheinlichkeitsdichte

$$P(t) = \Psi^*(t)\Psi(t) \tag{2.2}$$

ausgedrückt. Wenn die normierte Wellenfunktion nun bekannt ist, kann man alle beobachtbaren Eigenschaften durch Anwendung entsprechender mathematischer Operationen bestimmen.

2.1.2 Die Eigenwertgleichung

Um den Wert einer Observablen berechnen zu können, ordnet man einer Observablen A einen hermiteschen Operator[1] $\hat{A}$ zu [17, 18]. Hierbei gilt die Eigenwertgleichung

$$\hat{A}\psi(t) = \lambda\psi(t), \tag{2.3}$$

wobei λ den zu dem Operator $\hat{A}$ gehörenden Eigenwert darstellt.

In der Quantenmechanik von Spinsystemen definiert Gleichung (2.3) einen Satz von N Eigenfunktionen ψ und N Eigenwerten λ. Da die Eigenfunktionen zu einem hermiteschen Operator gehören, bilden diese wiederum einen orthonormalen Satz von Funktionen. Ein vollständiger Satz orthonormaler Funktionen ist ein Satz Basisfunktionen eines N-dimensionalen Vektorraumes, des Hilbert-Raumes.

2.1.3 Der Erwartungswert

Gleichung (2.3) macht nur Aussagen darüber, welche Meßwerte bei einer Messung auftreten können, aber nicht darüber, welcher der möglichen Meßwerte bei einer Messung zu erwarten ist. Für eine Einzelmessung kann im allgemeinen nicht vorhergesagt werden, welcher der Eigenwerte des Operators $\hat{A}$ als Meßwert auftritt. Aussagen können aber über den Erwartungswert gemacht werden, der bei einer Folge von Messungen zu erwarten ist [19]. Mathematisch formuliert man den Erwartungswert $\langle A\rangle$ einer Observablen A durch

$$\langle A\rangle = \int \Psi^*(t)\hat{A}\Psi(t)d\tau \tag{2.4}$$

[1] Ein hermitescher Operator hat reelle Eigenwerte.

als Skalarprodukt von Ψ und $\hat{A}\Psi$. Ist die Wellenfunktion des Systems eine Eigenfunktion des Operators $\hat{A}$, $\Psi = \psi_n$, dann kann man für jede Messung nur ein Ergebnis erhalten, nämlich den Eigenwert λ_n. Häufig ist die Gesamt-Wellenfunktion des Systems aber keine Eigenfunktion des Operators. Diese Wellenfunktion läßt sich allerdings als Linearkombination aus Basisfunktionen darstellen:

$$\Psi = \sum_{n=1}^{N} c_n \psi_n. \tag{2.5}$$

Mit dieser Linearkombination ergibt sich aus Gleichung (2.4) der Erwartungswert zu [14]

$$\langle A \rangle = \sum_{j=1}^{N} c_j^* c_j \lambda_j \tag{2.6}$$

unter Verwendung der Orthonormalität der Eigenfunktionen. Gleichung (2.6) kann man nun wie folgt interpretieren: Würde man für ein System, dessen Wellenfunktion eine Eigenfunktion ist, den Erwartungswert messen, so würde dieser stets einer der Eigenwerte des Systems sein. Betrachtet man jedoch ein Ensemble, wird ein Eigenwert proportional zum Koeffizientenquadrat $c_j^* c_j$ beobachten, d.h. man kann diesen Faktor als die Wahrscheinlichkeit ansehen, einen Eigenwert λ_j in einer einzelnen Messung zu erhalten. Dieser Wahrscheinlichkeits-Faktor fhrt dazu, daß der für ein Ensemble meßbare Erwartungswert $\langle A \rangle$ aus einem kontinuierlichen Wertebereich stammen kann, obwohl die für A erlaubten Werte durch die diskreten Menge der Eigenwerte gebildet wird.

2.2 Die Dichtematrix

Die Berechnung von Skalarprodukten und Erwartungswerten sind in der Quantenmechanik häufig auftretende Operationen. Berechnungen dieser Art werden durch eine Formulierung der Quantenmechanik von Spinsystemen erleichtert, die nicht mehr die Wellenfunktion des Systems in den Vordergrund stellt, sondern die Dichtematrix des Systems verwendet. Die Darstellung dieses Formalismus und die damit zusammenhängenden symbolischen Operationen werden zusätzlich durch die Dirac-Schreibweise [20] vereinfacht.

2.2.1 Die Dirac-Schreibweise

Die Dirac-Schreibweise ist ein kompakter Formalismus, um Skalarprodukte darzustellen. Eine Wellenfunktion $\psi(t)$ wird durch einen Ket-Vektor $|\psi(t)\rangle$ dargestellt. Die hierzu komplex-konjugierte Wellenfunktion $\psi^*(t)$ wird durch einen Bra-Vektor $\langle\psi(t)|$ dargestellt. Ein Skalarprodukt aus diesen beiden wird schließlich durch Zusammenziehen von Bra und Ket formuliert:

$$\int \psi^*(t)\varphi(t)d\tau \equiv \langle\psi(t)|\varphi(t)\rangle. \tag{2.7}$$

Eine beliebige Wellenfunktion $\Psi(t)$ kann man in der Dirac-Schreibweise als Superposition eines Satzes orthomormaler zeitunabhängiger Kets durch

$$|\Psi(t)\rangle = \sum_{n=1}^{N} c_n(t)\ |n\rangle \tag{2.8}$$

darstellen. Die Kets $|n\rangle$ werden auch als Eigenkets oder Basiskets bezeichnet, $c_n(t)$ sind zeitabhängige komplexe Zahlen und N ist die Dimension des zugehörigen Vektorraumes. Man kann die Koeffizienten $c_n(t)$ als Amplitudenfaktoren auffassen, welche den Anteil eines jeden Basiskets an der Gesamtwellenfunktion zu einer bestimmten Zeit angeben. Wenn die Basiskets nun zeitunabhängig sind, dann ist jegliche Zeitabhängigkeit der Wellenfunktion in den komplexen Koeffizienten enthalten.

Der Erwartungswert $\langle A\rangle$ einer Observablen A kann in der Dirac-Schreibweise unter Verwendung von Gleichung (2.8) als

$$\langle A\rangle = \sum_{m,n} c_m^*(t)c_n(t)\langle m|\hat{A}|n\rangle \tag{2.9}$$

formuliert werden. Die Skalarprodukte $\langle m|\hat{A}|n\rangle$ sind für einen gegebenen Basissatz konstant. Die Zeitabhängigkeit der Observablen A für einen bestimmten Zustand des Systems wird damit durch das Koeffizienten-Produkt $c_m^*(t)c_n(t)$ bestimmt. Sowohl $\langle m|\hat{A}|n\rangle$ als auch $c_m^*(t)c_n(t)$ können als Matrix formuliert werden. Es gilt

$$\langle A\rangle = \sum_{m,n} c_m^*(t)c_n(t)A_{mn}, \tag{2.10}$$

wobei A_{mn} das (mn)-te Matrixelement der $N \times N$-Matrix-Darstellung des Operators $\hat{A}$ in einer gegebenen Basis ist. Sobald nun die Matrixdarstellung der Koeffizienten bekannt ist, kann der Erwartungswert berechnet werden.

Die Produkte $c_m^*(t)c_n(t)$ können als Elemente einer Matrixdarstellung eines Operators $\hat{P}$ aufgefaßt werden

$$P_{mn} = \langle m|\hat{P}|n\rangle = c_m^* c_n. \tag{2.11}$$

Setzt man nun Gleichung (2.11) in Gleichung (2.10) ein, so erhält man für den Erwartungswert folgende einfache Gleichung

$$\langle A\rangle = \mathrm{Sp}\{\hat{P}\hat{A}\}. \tag{2.12}$$

Gleichung (2.12) bedeutet, daß man den Erwartungswert einer Observablen, z.B. die x-Magnetisierung, $\hat{A} = \hat{I}_x$, durch Berechnung der Spur des Matrixproduktes der Operatoren $\hat{P}$ und $\hat{A}$ erhalten kann.

2.2.2 Dichtematrix und Dichteoperator

Der Gültigkeitsbereich der bisher dargestellten Theorie ist auf Systeme begrenzt, die ausschließlich durch eine einzige Wellenfunktion vollständig beschrieben werden. Eine solche Theorie ist für die Beschreibung der Kernresonanz-Spektroskopie ungeeignet, weil man es dort mit einem System-Ensemble zu tun hat. Die vollständige Wellenfunktion eines solchen Systems wird eine sehr komplizierte Funktion sein, daher ist ein statistischer Ansatz vorzuziehen.

Jedes in der Probe enthaltene System kann durch eine Wellenfunktion Ψ beschrieben werden, deren Beitrag zum Ensemble-Zustand durch eine Wahrscheinlichkeitsdichte $\mathcal{P}(\Psi)$ ausgedrückt werden kann. Den statistischen Wert des Eigenwertes für einen solchen gemischten Zustand erhält man dann, indem man über die Wahrscheinlichkeitsverteilung mittelt

$$\begin{aligned}\overline{\langle A\rangle} &= \int \mathcal{P}(\Psi)\ \langle\Psi|\hat{A}|\Psi\rangle \\ &= \sum_{m,n} \overline{c_m^* c_n}\ \langle m|\hat{A}|n\rangle.\end{aligned} \tag{2.13}$$

Die Mittelung über alle Systeme der NMR-Probe zeigt, daß nur die Koeffizientenprodukte $c_m^*(t)c_n(t)$ von Spinsystem zu Spinsystem unterschiedlich sind. Die Matrixelemente A_{mn} sind für die einzelnen Spinsysteme der Probe gleich. Das Mittel der Koeffizentenprodukte, $\overline{c_m^*(t)c_n(t)}$, bildet nun eine Matrix, die man als Dichtematrix σ bezeichnet. Die Dichtematrix wiederum ist die Matrixdarstellung eines Operators, des sog. Dichteoperators $\hat{\sigma}$. Es gilt

somit

$$\begin{aligned}\overline{c_m^*(t)c_n(t)} &= \overline{\langle m|\hat{P}|n\rangle} \\ &= \langle m|\hat{\sigma}|n\rangle \\ &= \sigma_{mn} \end{aligned} \tag{2.14}$$

Man kann die Matrixform des Dichteoperators innerhalb der gegebenen Basis direkt auf Eigenschaften des Spinsystems zurückführen. Die Diagonalelemente der Dichtematrix entsprechen den Besetzungen der verschiedenen Energieniveaus, alle Außerdiagonalelemente stellen sog. Kohärenzen zwischen verschiedenen Zuständen dar.

Um nun einen Erwartungswert für ein System im gemischten Zustand berechnen zu können, setzt man anstelle des Operators $\hat{P}$ in Gleichung (2.12) den Dichteoperator $\hat{\sigma}$ ein. Es ergibt sich[2]

$$\overline{\langle A\rangle} = \mathrm{Sp}\{\hat{\sigma}\hat{A}^\dagger\}. \tag{2.15}$$

Kennt man nun die Matrixdarstellung des Dichteoperators und die des zur Observablen gehörenden Operators, so kann man mit Gleichung (2.15) den Erwartungswert der Observablen einfach bestimmen.

2.3 Die Liouville-von Neumann-Gleichung

Mit dem Dichteoperator hat man nun eine Beschreibung für reale Spinsysteme, die es erlaubt, einen beliebigen Erwartungswert zu jeder Zeit zu bestimmen. Da der Dichteoperator eine Funktion der Zeit ist, reicht es aus, eine exakte Darstellung des Dichteoperators zu einer Zeit $t = 0$ s zu kennen, wenn man eine Differentialgleichung finden kann, die die Entwicklung des Dichteoperators in Abhängigkeit der Zeit beschreibt. Diese Beschreibung ist mit der Liouville-von Neumann-Gleichung möglich [22]

$$\frac{d\hat{\sigma}(t)}{dt} = i[\hat{\sigma}(t), \hat{\mathcal{H}}]. \tag{2.16}$$

Die Lösung der Liouville-von Neumann-Gleichung (2.16) ist einfach, wenn der Hamilton-Operator zeitunabhängig ist. Wie sich durch Differenzieren

[2]Der in Gleichung (2.15) noch enthaltene Überstrich, der die Mittelung andeutet, wird im Folgenden nicht mehr verwendet. Dennoch ist von nun an immer die Mittelung gemeint, es sei denn, es wird explizit darauf hingewiesen.

leicht zeigen läßt, ist

$$\begin{aligned}\hat{\sigma}(t) &= \exp(-i\hat{\mathcal{H}}t)\ \hat{\sigma}(0)\ \exp(i\hat{\mathcal{H}}t) \\ &= \mathcal{U}\hat{\sigma}(0)\mathcal{U}^{-1} \end{aligned} \tag{2.17}$$

eine Lösung von Gleichung (2.16), wobei $\mathcal{U} = \exp(-i\hat{\mathcal{H}}t)$ als Propagator bezeichnet wird.

2.3.1 Lösung für zeitabhängige Hamilton-Operatoren

Eine NMR Pulssequenz besteht in der Regel aus zwei unterschiedlichen Teilen: Radiofrequenzpulsen und Wartezeiten, in denen kein Radiofrequenzpuls wirkt. Ein Radiofrequenzpuls wirkt direkt auf das Spinsystem als eine zeitabhängige Störung. Auch in den Wartezeiten spielen Prozesse eine Rolle, die das Spinsystem zeitabhängig verändern. In beiden Fällen wird der Hamilton-Operator zeitabhängig sein, so daß die Lösung der Liouville-von Neumann-Gleichung, wie sie in Gleichung (2.17) angegeben ist, nicht richtig ist. Man muß nun einen Weg finden, die Liouville-von Neumann-Gleichung auch für zeitabhängige Hamilton-Operatoren lösen zu können. Eine einfache Möglichkeit ist es, eine Transformation durchzuführen, die einen zeitabhängigen Hamiton-Operator in einen zeitunabhängigen überführt. Eine solche Transformation kann man sich als quantenmechanisches Äquivalent zu der Überführung in das rotierende Koordinatensystem im Vektormodell nach Bloch vorstellen.

Eine auf den Labor-Dichteoperator $\hat{\sigma}(t)$ angewendete Ähnlichkeits-Transformation [23] führt zu einem transformierten Dichteoperator $\hat{\sigma}^T(t)$, so daß

$$\sigma^T(t) = \hat{U}_T\ \sigma(t)\ \hat{U}_T^{-1}, \tag{2.18}$$

wobei $\hat{U}_T$ ein unitärer Operator ist. Analog zu Gleichung (2.16) kann man für die zeitliche Entwicklung des transformierten Dichteoperators formulieren

$$\frac{d\hat{\sigma}^T(t)}{dt} = i[\hat{\sigma}^T(t), \hat{\mathcal{H}}^T], \tag{2.19}$$

wobei $\hat{\mathcal{H}}^T$ für den transformierten Hamilton-Operator steht. Der transformierte Hamilton-Operator ergibt sich ebenfalls durch eine unitäre Transformation zu [14]

$$\hat{\mathcal{H}}^T = \hat{U}_T\ \hat{\mathcal{H}}\ \hat{U}_T^{-1} - i\hat{U}_T\frac{d\hat{U}_T^{-1}}{dt}. \tag{2.20}$$

Findet man nun eine solche unitäre Transformation, die einen transformierten Hamilton-Operator liefert, der zeitunabhängig ist, dann kann man für die Liouville-von Neumann-Gleichung eine zu Gleichung (2.17) analoge Lösung angeben

$$\begin{aligned}\hat{\sigma}^T(t) &= \exp(-i\hat{\mathcal{H}}^T t)\ \hat{\sigma}^T(0)\ \exp(i\hat{\mathcal{H}}^T t). \\ &= \mathcal{U}\hat{\sigma}^T(0)\mathcal{U}^{-1} \end{aligned} \tag{2.21}$$

Hierbei ist $\mathcal{U}$ ein unitärer Operator, den man als Propagator bezeichnet.

2.4 Quantenmechanische Beschreibung der NMR-Spektroskopie

Die quantenmechanische Beschreibung der NMR-Spektroskopie kann nun als Grundlage zur Simulation von NMR-Spektren dienen, denen eine beliebig komplizierte Pulsfolge vorausgegangen ist. Das Spinsystem wird zu Beginn eines Experimentes durch seine Gleichgewichts-Dichtematrix beschrieben, welche dann mit der Liouville-von Neumann-Gleichung (2.16) zeitlich entwickelt wird. Der Hamilton-Operator, der hierbei die Entwicklung der Dichtematrix bestimmt, besteht aus Zeeman- und skalaren Kopplungstermen

$$\hat{\mathcal{H}} = \sum_{i=1}^{N} \omega_i \hat{I}_{iz} + 2\pi \sum_{i>j}^{N} J_{ij} \hat{\mathbf{I}}_i \hat{\mathbf{I}}_j. \tag{2.22}$$

Den Erwartungswert bestimmt man nach Gleichung (2.15) als Spur des Produktes von Dichteoperator und Beobachtungsoperator.

Nimmt man von dem Gitter an, daß es eine unendliche Wärmekapazität besitzt, d.h. es befindet sich im thermischen Gleichgewicht mit der Umgebung bei der Temperatur T, dann befinden sich auch die Spinzustände im thermischen Gleichgewicht mit dem Gitter. Daraus folgt, daß die Wahrscheinlichkeitsdichte $\mathcal{P}(\Psi)$ (vergl. Abschnitt 2.2.2) so beschränkt ist, daß die Spinzustände, welche den Diagonalelementen der Dichtematrix entsprechen, gemäß der Boltzmann-Verteilung besetzt sind (sog. *ensemble average*). Darüberhinaus ist die Dichtematrix dann eine Diagonalmatrix, denn es bestehen keine Kohärenzen zwischen den einzelnen Zuständen. Ein Dichteoperator der Form

$$\hat{\sigma}^{eq} = \frac{\exp(\frac{-\hat{\mathcal{H}}}{k_B T})}{\mathrm{Sp}\{\exp(\frac{-\hat{\mathcal{H}}}{k_B T})\}} \tag{2.23}$$

erfüllt diese Forderungen. Man kann nun zwei Vereinfachungen einführen: (1) Die Zeeman-Terme des Hamilton-Operators (2.22) sind erheblich größer als die bilinearen Terme; J-Kopplungen sind in Größenordnungen von Hz, während Larmorfrequenzen Größenordnungen von MHz aufweisen. (2) Die zu den Larmorfrequenzen gehörenden Energien sind wesentlich kleiner als die thermische Energie. Daher kann man die Tylor-Reihenentwicklung der Exponentialfunktion nach dem ersten Glied abbrechen. Somit ergibt sich als Näherung

$$\hat{\sigma}^{eq} = \frac{\hat{1} - \frac{1}{k_{\mathrm{B}}T} \sum_i \omega_i \hat{I}_{iz}}{\mathrm{Sp}\{\hat{1}\}}. \tag{2.24}$$

Wenn man nun beachtet, daß der Einheitsoperator invariant unter unitären Transformationen ist, so kann man ihn vernachlässigen und gelangt zu

$$\hat{\sigma}^{eq} = -\frac{1}{k_{\mathrm{B}}T} \sum_{i=1}^{N} \omega_i \hat{I}_{iz}. \tag{2.25}$$

Für einen einzelnen Spin ist somit $\hat{\sigma}^{eq} \propto \hat{I}_z$.

Die komplexe Magnetisierung, die während der Akquisitions-Zeit eines NMR-Experimentes gemessen wird, ist durch

$$\hat{M}^- = N\gamma\hbar \ \mathrm{Sp}\{\hat{\sigma}(t) \ \hat{F}^+\} \tag{2.26}$$

gegeben [22]. N entspricht hierbei der Anzahl Spins pro Einheitsvolumen. Für den Beobachtungsoperator gilt

$$\hat{F}^+ = \sum_{k=1}^{K} \hat{I}_k^+ = \sum_{k=1}^{K} (\hat{I}_{kx} + i\hat{I}_{ky}), \tag{2.27}$$

wobei K die Anzahl der Spins gleicher Art bezeichnet. Eine Behandlung von Spinsystemen im Dichtematrix-Formalismus führt somit zu einer Grundlage, welche zur Berechnung beliebig komplizierter NMR-Experimente genutzt werden kann.

2.5 Quantenmechanik von N-Spinsystemen

In diesem Abschnitt soll der Nutzen des Dichteoperatoren verwendenden Ansatzes für skalar gekoppelte N-Spinsysteme gezeigt werden. Die zeitliche Entwicklung solcher Spin-Systeme kann durch eine alternative Beschreibung, z.B.

das Bloch-Modell, nicht mehr korrekt wiedergegeben werden. Hierbei ist das Kernproblem, eine geeignete Matrixdarstellung von Wellenfunktionen und Operatoren eines allgemeinen N-Spinsystems zu finden. Im Rahmen dieser Arbeit wurde eine direkte Produktbasis verwendet; allerdings ist es durch Ähnlichkeitstransformationen möglich, auch andere Basissätze zu verwenden [25].

2.5.1 Direkte Produktbasis

Um N-Spinsysteme mit $N > 1$ beschreiben zu können, muß eine entprechende Basis der Operator-Algebra formuliert werden. Eine mögliche Darstellung ist die sog. direkte Produktbasis. Operatoren können direkt aus dem Ein-Spinraum durch das Bilden des direkten Produktes mit $N-1$ Einheitsoperatoren, $\hat{1}$, dargestellt werden

$$\hat{I}_{\eta k}^{N\ \mathrm{Spin}} = \hat{1}_1 \otimes \hat{1}_2 \otimes \cdots \otimes \hat{1}_{k-1} \otimes \hat{I}_{\eta k}^{1\ \mathrm{Spin}} \otimes \hat{1}_{k+1} \otimes \cdots \otimes \hat{1}_N, \tag{2.28}$$

wobei $\eta = x, y$ oder z ist. Durch diese Vorgehensweise erhält man z.B. aus der 1-Spinbasis für Spin-$\frac{1}{2}$-Teilchen 2^N neue Basisoperatoren, mit denen die Algebra des N-Spinraumes vollständig darstellbar ist.

2.5.2 Skalarer Kopplungs-Hamilton-Operator

Der Laborsystem-Hamilton-Operator, welcher die freie Präzession beschreibt, ist durch Gleichung (2.22) gegeben. Die Eigenfunktionen dieses Hamilton-Operators werden als Basisfunktionen verwendet, um den Dichteoperator in der Matrixdarstellung zu erhalten. An dieser Stelle soll noch auf die Einflüsse einer starken Kopplung eingangen werden. Die in der direkten Produktbasis definierten Wellenfunktionen sind nur dann Eigenfunktionen des Hamiltonoperators, wenn die Bedingung

$$\frac{|2\pi J_{nm}|}{|\omega_n - \omega_m|} \ll 1 \tag{2.29}$$

erfüllt ist. Diese Bedingung beschreibt den Bereich der sog. schwachen Kopplung. Ist sie nicht erfüllt, so befindet man sich im Bereich der starken Kopplung. In diesem Fall sind die Wellenfunktionen mit derselben magnetischen Quantenzahl innerhalb der Produktbasis nicht mehr vollständig von einander unabhängig. Einen geeigneten Basissatz kann man erhalten, indem man

Linearkombinationen aus der Teilmenge von Wellenfunktionen mit derselben magnetischen Quantenzahl bildet. Gruppentheoretische Methoden [25] erleichtern dies im Falle stark gekoppelter Spinsystemen mit mehr als zwei Spins. Diese Betrachtungen werden notwendig, wenn beispielsweise äquivalente Kerne[3] in dem zu beschreibenden Molekül enthalten sind.

[3] z.B. die drei Protonen einer Methylgruppe.

Kapitel 3

Relaxation

Bisher wurde der Dichtematrixformalismus dazu verwendet, den Einfluß verschiedener Spin-Hamilton-Operatoren auf den Dichteoperator eines Spinsystems zu beschreiben. Doch darüber hinaus gibt es noch andere Vorgänge, die die Dichtematrix beeinflussen. In diesem Kapitel soll das Phänomen der Kernspin-Relaxation vorgestellt werden. Hierbei kehrt eine vom thermodynamischen Gleichgewichtszustand verschiedene Dichtematrix in diesen Gleichgewichtszustand zurück. Es soll gezeigt werden, wie man das Phänomen der Relaxation in den Dichtematrixformalismus einschließen kann.

3.1 Allgemeine Einführung

Voraussetzung für Relaxation ist, daß sich das Spinsystem in einem Zustand befindet, der nicht dem thermodynamischen Gleichgewichtszustand entspricht. In der Sprache des Dichtematrixformalismus bedeutet dies, daß zum einen die Diagonalelemente der Dichtematrix nicht mehr die Gleichgewichtspopulationen der Spinzustände gemäß einer Boltzmann-Verteilung darstellen. Darüberhinaus können in diesem vom Gleichgewichtszustand verschiedenen Zustand auch Kohärenzen vorhanden sein, welche durch Außerdiagonalelemente der Dichtematrix beschrieben werden. Relaxiert ein Spinsystem nun, so müssen die Außerdiagonalelemente der Dichtematrix wieder auf Null abklingen und die Diagonalelemente müssen wieder die einer Boltzmann-Verteilung folgenden Populationen der Spinzustände darstellen, wie sie durch Gleichung 2.25 beschrieben werden.

Man kann zeigen [21], daß sowohl spontane als auch induzierte Emission von Photonen keinesfalls einen Übergang aus einem energetisch angeregten

Zustand in den Grundzustand eines magnetischen Dipols mit genügend hoher Wahrscheinlichkeit erlauben. Somit können diese beiden Mechanismen nicht die Ursache für Relaxation sein.

Die Ursache für Relaxation liegt nicht in Übergängen von Energiezuständen unter Aussenden von Photonen, sondern darin, daß das Spinsystem an das Gitter gekoppelt ist. Das Gitter besteht sowohl aus allen Freiheitsgraden der beteiligten Moleküle als auch aus anderen Molekülen, die das zu untersuchende Spinsystem beeinflussen (vergl. Abschnitt 2.2.2). Die Energiezustände, die das Gitter einnehmen kann, sind quasi-kontinuierlich; daher wird zu jedem bei Relaxation auftretenden Energieübergang innerhalb des Spinsystems auch ein Energieübergang innerhalb des Gitters vorhanden sein, so daß diese Energie in einem Austauschprozeß aufgenommen werden kann. Darüberhinaus nimmt man an, daß das Gitter zu jeder Zeit im thermischen Gleichgewicht mit der Umgebung steht. Dadurch, daß die Moleküle einer Brownschen Bewegung gehorchen, beeinflussen sie das lokale Magnetfeld am Ort des zu beobachtenden Spins, d.h. die lokalen magnetischen Felder werden zeitabhängig[1]. Diese lokalen Felder kann man nun mittels Fourier-Analyse zerlegen. Zusätzlich kann man diese Felder durch zum Hauptmagnetfeld parallel und rechtwinkling verlaufende Anteile darstellen.

Die transversalen Komponenten sind für die nicht-adiabatischen Anteile der Relaxation verantwortlich. Enthält das Fourier-Spektrum der fluktuierenden transversalen lokalen Felder Anteile mit Frequenzen, welche den Übergängen zwischen Eigenzuständen des Spinsystems entsprechen, dann werden solche Übergänge durch Energieaustausch zwischen Gitter und Spinsystem möglich. Da man nun Relaxation beobachtet, muß der Prozess des Energietransfers aus dem Gitter in das Spinsystem weniger wahrscheinlich sein als der umgekehrte Vorgang. Dies liegt daran, daß das Gitter immer im thermischen Gleichgewicht mit der Umgebung ist und die Besetzung der Energieniveaus innerhalb des Gitters einer Boltzmann-Verteilung folgt. Ein Prozeß, der das Spinsystem über einen Energieaustausch an das Gitter koppelt, bringt das Spinsystem in thermisches Gleichgewicht mit dem

[1]Wenn die Molekülbewegung statistisch ist, werden diese durch die zufällige Störung modulierten lokalen magnetischen Felder auch als stochastische lokale Felder bezeichnet. Resultiert eine statistische Fluktuation der elementaren Dipolmomente aus der als statistisch aufgefaßten Molekülbewegung, dann kann man eine hieraus resultierende Relaxation im sog. Mechanismus der stochastischen Felder ausdrücken.

Gitter. Daher werden die Besetzungen innerhalb des Spinsystems auch einer Boltzmann-Verteilung folgen müssen, wenn sich das vollständig relaxierte System im thermischen Gleichgewichtszustand befindet. Nach einer Störung dieser Verteilung muß ein Vorgang, der die Gleichgewichtsverteilung wieder herstellt, wahrscheinlicher sein als ein Prozeß, welcher den stationären Zustand noch weiter von der Gleichgewichtslage entfernt. Daraus ergibt sich, daß durch einen Übergang aus einem angeregten Kernniveau des Spinsystems in einen energetisch tiefer liegenden im Gitter ein Übergang von einem niedrigeren Energieniveau zu einem höheren induziert wird, z.B. wird eine molekulare Rotation beschleunigt. Da diese Art von Relaxation auf einer energetischen Kopplung zwischen Spinsystem und Gitter beruht, bezeichnet man diese als Spin-Gitter-Relaxation (vergl. Abbildung 3.1.c).

Die zum äußeren Magnetfeld parallelen Anteile der fluktuierenden Felder sind nun für die adiabatischen Beiträge der Relaxation verantwortlich. Durch diese Felder werden Veränderungen in der z-Richtung des am Kernort wirkenden Gesamtfeldes verursacht, wodurch sich die energetische Aufspaltung der Energieniveaus ebenfalls statistisch ändert. Daraus folgt, daß sich die Larmorfrequenzen der Kernspins ebenso zufällig verändern. Über eine längere Zeit hinweg werden die Kernspins ihre Phasenkohärenz verlieren und die Außerdiagonalelemente der Dichtematrix werden auf Null abklingen. Dabei wird die Besetzung der Energieniveaus allerdings nicht verändert, d.h. es wird keine Energie zwischen dem Gitter und dem Kernspinssystem ausgetauscht. Da diese Art von Relaxation auf den Phasenverlust der Präzessionsbewegung der einzelnen Spins zurückzuführen ist, bezeichnet man sie auch als Spin-Spin-Relaxation (vergl. Abbildung 3.1.a).

3.2 Semiklassische Relaxationstheorie

Eine mikroskopische halbklassische Theorie, welche die Spinrelaxation beschreibt, wurde von Wangsness, Bloch und Redfield [26, 27] formuliert. Hierbei wird das Spinsystem quantenmechanisch behandelt, wohingegen das Gitter durch die Gesetze der klassischen Physik beschrieben wird. Diese Theorie birgt allerdings den Nachteil, daß das Spinsystem zu einem Endzustand entwickelt wird, in dem die Energiezustände gleichbesetzt sind. Daher ist diese Theorie eigentlich nur für unendliche Spintemperaturen korrekt; man kann

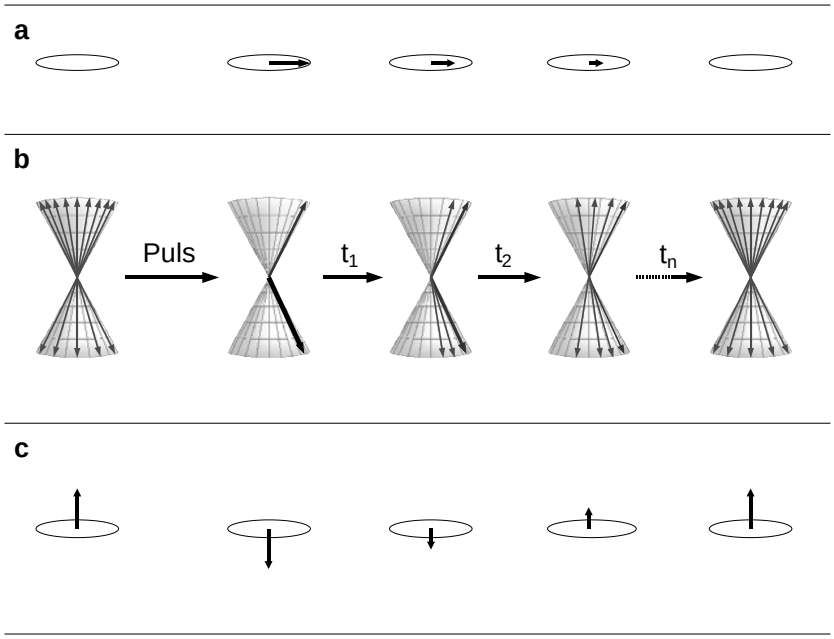

ABBILDUNG 3.1: Darstellung von Relaxation im Vektormodell. (a) Spin-Spin-Relaxation. Durch den Verlust der Phasenbeziehung der einzelnen Spins klingt der in die transversale Ebene projezierte Summenvektor auf Null im thermischen Gleichgewichtszustand ab. Wegen dieses Zusammenhanges bezeichnet man diese Art von Relaxation auch als transversale Relaxation. (b) Präzessionskegel eines Spin-$\frac{1}{2}$-Systems. Aus den den einzelnen Spins zugeordneten Vektoren ergibt sich ein Gesamtmagnetisierungsvektor, dessen Projektionen in a und c Verwendung finden. Zu Beginn befindet sich das System im thermischen Gleichgewichtszustand. Stört man diesen, z.B. durch einen Puls, so wird das System nach der Zeit t_n wieder den Gleichgewichtszustand erreicht haben. Die beiden unabhängigen Vorgänge sind getrennt in a und c gezeigt. (c) Spin-Gitter-Relaxation. Aufgrund von Energieübergängen im Spinsystem werden in zunehmendem Maße energetisch günstigere Kernzustände besetzt bis schließlich im thermischen Gleichgewicht die Besetzung wieder der Boltzmann-Verteilung gehorcht. Da hierbei der in die longitudinale Ebene projezierte Summenvektor wieder in seine Ausgangslage zurückkehrt, bezeichnet man diese Art von Relaxation auch als longitudinale Relaxation.

allerdings eine empirische Korrektur einführen, die auch die Beschreibung eines der Boltzmann-Verteilung gemäßen Gleichgewichtszustandes ermöglicht. Eine vollständig auf der Quantenmechanik beruhende Beschreibung [21, 27] der Spinrelaxation entbehrt dieses Fehlers und sagt die Erreichung des Gleichgewichtszustandes korrekt voraus. Diese Behandlung ist jedoch kompliziert und für diese Arbeit nicht relevant.

3.2.1 Erweiterte Liouville-von Neumann-Gleichung

In der semiklassischen Relaxationstheorie formuliert man den Hamilton-Operator des betrachteten Systems als Summe eines deterministischen quantenmechanischen Hamilton-Operators $\hat{\mathcal{H}}_{\mathrm{det}}(t)$, der nur auf das Spinsystem wirkt, und eines stochastischen Hamilton-Operators $\hat{\mathcal{H}}_{\mathrm{stat}}(t)$, welcher das Spinsystem an das Gitter koppelt. Darüberhinaus kann man den deterministischen Hamilton-Operator $\hat{\mathcal{H}}_{\mathrm{det}}(t)$ wiederum als eine Summe eines zeitunabhängigen Hamilton-Operators $\hat{\mathcal{H}}_0$ und eines zeitabhängigen Hamilton-Operators $\hat{\mathcal{H}}_{\mathrm{rf}}(t)$ darstellen

$$\begin{aligned}\hat{\mathcal{H}}(t) &= \hat{\mathcal{H}}_{\mathrm{det}}(t) + \hat{\mathcal{H}}_{\mathrm{stat}}(t) \\ &= \hat{\mathcal{H}}_0 + \hat{\mathcal{H}}_{\mathrm{rf}}(t) + \hat{\mathcal{H}}_{\mathrm{stat}}(t). \end{aligned} \tag{3.1}$$

Man kann die Hamilton-Operatoren $\hat{\mathcal{H}}_{\mathrm{rf}}(t)$ und $\hat{\mathcal{H}}_{\mathrm{stat}}(t)$ als zeitabhängige Störungen auffassen, die auf den zeitunabhängigen Haupt-Hamilton-Operator $\hat{\mathcal{H}}_0$ wirken. Die Entwicklung der Dichtematrix wird allgemein durch die Liouville-von Neumann-Gleichung

$$\frac{d\hat{\sigma}(t)}{dt} = i[\hat{\sigma}(t), \hat{\mathcal{H}}(t)] \tag{3.2}$$

beschrieben. Man kann den Einfluß des zeitunabhängigen Hamilton-Operators entfernen, wenn man die Liouville-von Neumann-Gleichung (3.2) in ein Wechselwirkungs-Koordinatensystem überführt. In Abwesenheit eines Radiofrequenzfeldes sind der Dichteoperator und der stochastische Hamilton-Operator durch

$$\hat{\sigma}^T(t) = \exp(i\hat{\mathcal{H}}_0 t)\ \hat{\sigma}(t)\ \exp(-i\hat{\mathcal{H}}_0 t) \tag{3.3}$$

$$\hat{\mathcal{H}}^T_{\mathrm{stat}}(t) = \exp(i\hat{\mathcal{H}}_0 t)\ \hat{\mathcal{H}}_{\mathrm{stat}}(t)\ \exp(-i\hat{\mathcal{H}}_0 t) \tag{3.4}$$

gegeben. Daraus ergibt sich die Liouville-von Neumann-Gleichung in ihrer transformierten Form zu

$$\frac{d\hat{\sigma}^T(t)}{dt} = i[\hat{\sigma}^T(t), \hat{\mathcal{H}}^T_{\text{stat}}(t)]. \tag{3.5}$$

Die Transformation in das Wechselwirkungs-Koordinatensystem ist zu der in das rotierende Koordinatensystem (vergl. Abschnitt 2.3.1) isomorph; es besteht jedoch einen wichtiger Unterschied. Die Transformation in das rotierende Koordinatensystem ergibt einen zeitunabhängigen Hamilton-Operator, indem durch die Transformation die Zeitabhängigkeit des Radiofrequenz-Hamilton-Operators entfernt wird. Der zeitunabhängige Haupt-Hamilton-Operator $\hat{\mathcal{H}}_0$ wirkt vollständig im rotierenden Koordinatensystem. Die Überführung in das Wechselwirkungs-Koordinatensystem hingegen entfernt ausdrücklich die Abhängigkeit von dem Haupt-Hamilton-Operator $\hat{\mathcal{H}}_0$. Die Zeitabhängigkeit des transformierten stochastischen Hamilton-Operators $\hat{\mathcal{H}}^T_{\text{stat}}(t)$ bleibt jedoch erhalten.

Die Liouville-von Neumann-Gleichung (3.5) kann man durch Integration umformulieren [21, 27]. Man erhält[2]

$$\frac{d\hat{\sigma}^T(t)}{dt} = -\int_0^\infty d\tau\, \overline{[\hat{\mathcal{H}}^T_{\text{stat}}(t),\, [\hat{\mathcal{H}}^T_{\text{stat}}(t-\tau)},\, \hat{\sigma}^T(t) - \hat{\sigma}(0)]]. \tag{3.6}$$

Hierbei gilt Gleichung (3.6) nur unter folgenden Annahmen [14, 21]: (1) Es muß erlaubt sein, die Korrelation von $\hat{\mathcal{H}}^T_{\text{stat}}(t)$ und $\hat{\sigma}^T(t)$ während der Mittelung zu vernachlässigen, damit man diese beiden Größen eigenständig mitteln kann. (2) Die Korrelationszeit $\tau_{\mathcal{H}}$ für $\hat{\mathcal{H}}^T_{\text{stat}}(t)$ ist erheblich kleiner als t und R_{ij}^{-1}, wobei R_{ij} die Relaxations-Geschwindigkeitskonstante des Dichtematrix-Elementes σ_{ij} darstellt[3]. (3) Der Term $\hat{\sigma}(0)$ kann in Gleichung (3.6) eingeführt werden, damit sichergestellt wird, daß der thermische Gleichgewichtszustand durch die Entwicklung des Dichteoperators nach Gleichung (3.6) erreicht werden kann[4].

[2]Der Überstrich bedeutet hierbei, daß über die Kernspins gemittelt wird. Gemäß der Vereinbarung in Kapitel 2.2.2 ist dies bei Dichteoperatoren bereits impliziert.

[3]In Flüssigkeiten beträgt $\tau_{\mathcal{H}}$ etwa den Wert einer Kollisionszeit, also 10^{-14} s bis 10^{-12} s. t und R_{ij}^{-1} sind um Größenordnungen größer. Die Kollisionszeiten sind allerdings stark von der Molekülgröße abhängig. Die hier durchgeführte Schätzung gilt für mittelgroße Moleküle; hierbei ist diese Bedingung immer erfüllt.

[4]Dies ist die zu Beginn angeführte empirische Korrektur, die bei der semiklassischen Relaxationstheorie nötig ist.

Wie man Gleichung (3.6) entnehmen kann, wäre der Dichteoperator zeitunabhängig, wenn der stochastische Hamilton-Operator verschwände. Da dieser Hamilton-Operator nur eine geringfügige Störung darstellt, wird die Veränderung der Dichtematrix langsam sein.

Man kann den stochastischen Hamilton-Operator und den transformierten stochastischen Hamilton-Operator (3.4) durch eine Linerarkombination von Tensor-Operatoren ausdrücken [21, 28]:

$$\hat{\mathcal{H}}_{stat}(t) = \sum_{q=-2}^{2} F^q(t)\hat{A}^q. \tag{3.7}$$

Hierbei sind $\hat{A}^q$ Tensor-Operatoren, die auf die Variablen des betrachteten Spinsystems wirken. Durch den Index q werden die irreduziblen Komponenten der Tensoren unterschieden. Darüberhinaus kann man die Tensor-Operatoren durch Basisoperatoren ausdrücken:

$$\hat{A}^q = \sum_p \hat{A}^q_p. \tag{3.8}$$

Mit den Zerlegungen (3.7) und (3.8) ergibt sich nun für die Liouville-von Neumann-Gleichung (3.6) in Operatorschreibweise

$$\frac{d\hat{\sigma}^T(t)}{dt} = -\frac{1}{2}\sum_q \sum_p [\hat{A}^{-q}_p,\ [\hat{A}^q_p,\ \hat{\sigma}^T(t) - \hat{\sigma}(0)]]\ j^q(\omega_p). \tag{3.9}$$

Die spektrale Dichtefunktion $j^q(\omega_p)$ ist durch

$$j^q(\omega_p) = \mathrm{Re}\left\{\int_{-\infty}^{\infty} \overline{F^q(t)\ F^{-q}(t+\tau)}\exp(-i\omega\tau)d\tau\right\} \tag{3.10}$$

gegeben. $F^q(t)$ sind zeitabhängige Zufallsfunktionen[5]. Transformiert man Gleichung (3.9) in das Labor-Koordinatensystem zurück, so erhält man die Liouville-von Neumann-Gleichung in Super-Operator-Schreibweise[6]

$$\frac{d\hat{\sigma}(t)}{dt} = -i\ \hat{\hat{\mathcal{H}}}_0\ \hat{\sigma}(t) - \hat{\hat{\Gamma}}\left\{\hat{\sigma}(t) - \hat{\sigma}(0)\right\}, \tag{3.11}$$

[5]Diese Zufallsfunktionen stammen aus der Zerlegung des stochastischen Hamilton-Operators (3.7). Sie treten hierbei als Koeffizienten der Basisoperatoren $\hat{A}^q$ auf und werden auf Kugelflächenfunktionen vom Rang 2 zurückgeführt (vergl. Gleichung (3.14)).

[6]Als Super-Operator bezeichnete man solche Operatoren, die auf Operatoren wirken. Z.B. kann man den folgenden Kommutator als Super-Operator formulieren:

$$[\hat{\mathcal{H}}_0, \hat{\sigma}(t)] = \hat{\hat{\mathcal{H}}}_0\ \hat{\sigma}(t).$$

Super-Operatoren werden durch ein doppeltes Operatorzeichen kenntlich gemacht [22].

mit dem Relaxations-Superoperator

$$\hat{\hat{\Gamma}} = \sum_q \sum_p [\hat{A}_p^{-q},\ [\hat{A}_p^q,\ \ ]]\ j^q(\omega_p). \tag{3.12}$$

Man erkennt an Gleichung (3.11), daß die ursprüngliche Liouville-von Neumann-Gleichung (2.16) nur um einen Summanden erweitert wurde, d.h. daß effektiv die Wirkung verschiedener Operatoren, hier der das Spinsystem beschreibende Hamilton-Operator und der Relaxations-Operator, linear gekoppelt wird.

3.2.2 Spektrale Dichtefunktionen

Gleichung (3.10) stellt eine allgemeine Formulierung der spektralen Dichtefunktion dar. Betrachtet man Relaxation isotroper Flüssigkeiten, so gilt im Hochtemperatur-Grenzfall [29]:

$$j^q(\omega) = (-1)^q\ j^0(\omega). \tag{3.13}$$

Diese Vereinfachung führt dazu, daß nur eine autospektrale Dichtefunktion $j^{q=0}(\omega)$ berechnet werden muß. Die räumlichen Zufallsfunktionen $F^0(t)$ aus Gleichung (3.10) kann man nun für $q = 0$ als

$$F^0(t) = c_0(t)\ Y^0(\theta(t),\ \varphi(t)) \tag{3.14}$$

formulieren. Hierbei stellt $Y^0(\theta(t),\ \varphi(t))$ eine normierte Kugelflächenfunktion dar (vergl. Tabelle 3.1) und $c_0(t)$ ist eine räumliche Funktion, die vom jeweiligen Relaxationsmechanismus bestimmt wird. Wenn man nun die statistische Korrelationsfunktion

$$C(\tau) = \overline{c_0(t)\ c_0(t+\tau)\ Y^0\left(\theta(t), \varphi(t)\right)\ Y^0\left(\theta(t+\tau), \varphi(t+\tau)\right)} \tag{3.15}$$

in Gleichung (3.10) einführt, so ergibt sich für die autospektrale Dichtefunktion

$$j^0(\omega) = \mathrm{Re}\left\{\int_{-\infty}^{\infty} C(\tau)\ \exp(i\omega\tau) d\tau\right\}. \tag{3.16}$$

Für ein starres sphärisches Molekül, dessen Bewegung durch die Brownsche Molekularbewegung beschrieben wird, ist $c_0(t) = c_0$ konstant. Somit ergibt sich die autospektrale Dichtefunktion zu

$$j^0(\omega) = c_0^2\ J(\omega). \tag{3.17}$$

q	Y^q	$Y^{q*} = Y^{-q}$
0	$\frac{1}{2}(3\cos^2\theta(t) - 1)$	$\frac{1}{2}(3\cos^2\theta(t) - 1)$
1	$\sqrt{\frac{3}{2}}\sin\theta(t)\cos\theta(t)\exp(i\varphi(t))$	$\sqrt{\frac{3}{2}}\sin\theta(t)\cos\theta(t)\exp(-i\varphi(t))$
2	$\sqrt{\frac{3}{8}}\sin^2\theta(t)\exp(2i\varphi(t))$	$\sqrt{\frac{3}{8}}\sin^2\theta(t)\exp(-2i\varphi(t))$

TABELLE 3.1: Normierte Kugelflächenfunktionen [30]

$J(\omega)$ ist hierbei die sog. reduzierte spektrale Dichtefunktion, die durch

$$J(\omega) = \mathrm{Re}\left\{\int_{-\infty}^{\infty} C_{00}^2(\tau)\exp(-i\omega\tau)d\tau\right\} \tag{3.18}$$

gegeben ist. Darin (3.18) bedeutet $C_{00}^2(\tau)$ wiederum die Orientierungs-Korrelationsfunktion, welche als

$$C_{00}^2(\tau) = \overline{Y_0\left(\theta(t),\varphi(t)\right)\ Y^0\left(\theta(t+\tau),\varphi(t+\tau)\right)} \tag{3.19}$$

formuliert wird. Im Falle isotroper Rotation eines starren Rotators kann man die Orientierungs-Korrelationsfunktion durch

$$C_{00}^2(\tau) = \frac{1}{5}\exp\left(\frac{-\tau}{\tau_c}\right) \tag{3.20}$$

ausdrücken. τ_c ist die Korrelationszeit, d.h. die Zeit, welche der betreffende Kern-Kern-Vektor im Mittel benötigt, um sich einen Radian zu drehen. Die Korrelationszeit hängt allgemein von der Molekülgröße, der Viskosität der Lösungsmittels und der Temperatur ab, sie liegt im Bereich von Picosekunden für kleine Moleküle bis Nanosekunden für Makromoleküle in wäßriger Lösung. Mit Gleichung (3.20) ergibt sich die spektrale Dichtefunktion zu

$$J(\omega) = \frac{2\tau_c}{5\left(1+\omega^2\tau_c^2\right)} \tag{3.21}$$

Aus Gleichung (3.21) ergibt sich eine Lorentz-Form für die spektrale Dichtefunktion eines starren Rotators (vergl. Abbildung 3.2). Man unterscheidet zwei Grenzfälle für die spektrale Dichtefunktion. Ist die molekulare Bewegung hinreichend langsam, damit $\omega^2\tau_c^2 \gg 1$ erfüllt ist, so befindet man sich im Spindiffusions-Grenzfall. Hierbei wird die spektrale Dichtefunktion als $J(\omega) \propto \omega^{-2}$ angenommen. Wenn die Molekülbewegung hingegen hinreichend

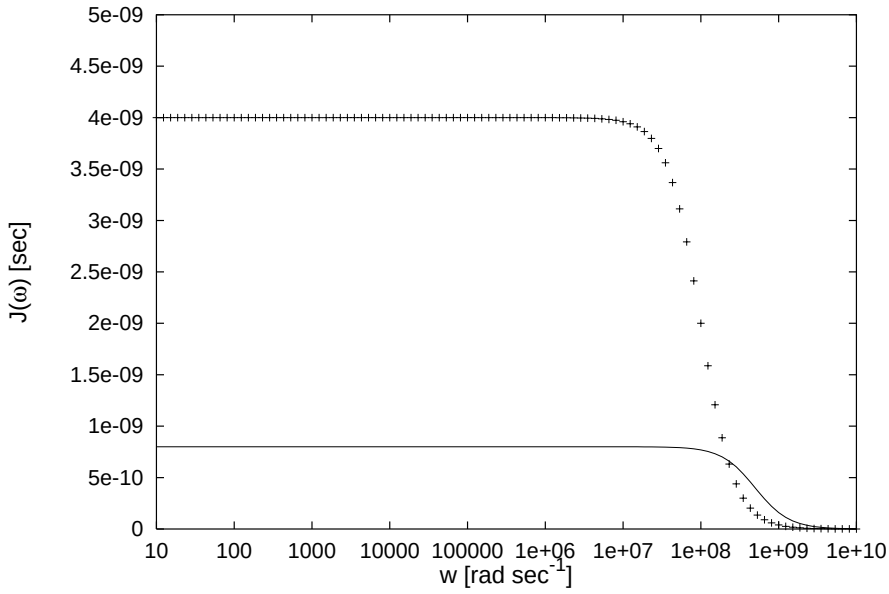

ABBILDUNG 3.2: Spektrale Dichtefunktionen eines starren Rotators. Die Graphen wurden nach Gleichung (3.21) unter Verwendung der Korrelationszeiten $\tau_c = 2$ ns (——) und $\tau_c = 10$ ns (+++) berechnet.

schnell ist, so daß $\omega^2\tau_c^2 \ll 1$ gilt, so befindet man sich im sog. *extreme narrowing limit*[7]. Für die spektrale Dichtefunktion wird hierbei $J(\omega) \approx J(0)$ angenommen (vergl. Abbildung 3.2).

3.2.3 Wechselwirkung von Relaxation

Oftmals ist mehr als nur ein statistischer Hamilton-Operator wirksam, welcher Relaxation verursachen kann. Wenn dies der Fall ist, dann kann man Gleichung (3.7) zu

$$\hat{\mathcal{H}}_{stat}(t) = \sum_m \sum_q F_m^q(t)\hat{A}_m^q \tag{3.22}$$

verallgemeinern, wobei sich die Summe über m auf die verschiedenen Relaxations-Wechselwirkungen bezieht. Aus Gleichung (3.12) ergibt sich unter dieser Verallgemeinerung ein Relaxations-Superoperator, welcher diese Wechselwirkungen berücksichtigt [14]

[7]Die Bedingung des *extreme narrowing limit* wird von kleinen Molekülen in nichtviskosen Lösungsmitteln erfüllt.

$$\begin{aligned}\hat{\hat{\Gamma}} &= \sum_m \sum_q \sum_p [\hat{A}^{-q}_{mp},\ [\hat{A}^{q}_{mp},\ ]]\ j_q(\omega_p) \\ &\quad + \sum_{m,n} \sum_q \sum_p [\hat{A}^{-q}_{mp},\ [\hat{A}^{q}_{np},\ ]]\ j^q_{mn}(\omega_p) \\ &= \sum_m \hat{\hat{\Gamma}}^m + \sum_{m,n} \hat{\hat{\Gamma}}^{mn}.\end{aligned} \tag{3.23}$$

Hierbei stellt die Summe über m den Relaxations-Superoperator dar, welcher die Relaxation auf Grund des m-ten Relaxationsmechanismus beschreibt; die Summe über mn steht für einen kreuzkorrelierten Relaxationsmechanismus, welcher durch Interferenzeffekte aus dem m-ten und n-ten Einzelmechanismus entsteht.

3.3 Relaxationsmechanismen

Wie in Abschnitt 3.1 herausgestellt wurde, kann Spinrelaxation mit ausreichender Wahrscheinlichkeit nur durch Prozesse verursacht werden, welche in irgendeiner Art und Weise ein fluktuierendes Feld mit dem zu relaxierenden Kern wechselwirken lassen. Hierbei muß zudem die Modulationsfrequenz mit der Energie des zu relaxierenden Energieüberganges übereinstimmen. Im folgenden soll ein kurzer Überblick über die wichtigsten Relaxationsmechanismen gegeben werden [14, 21, 22, 31, 32].

3.3.1 Dipol-Dipol Relaxation

Jeder magnetische Kern innerhalb eines Moleküls erzeugt unmittelbar ein zum magnetischen Moment des Kernes proportionales magnetisches Dipolfeld. Wenn sich ein Molekül in Lösung ungeordnet bewegt, wird dieses Feld fluktuieren[8] und somit einen geeigneten Relaxations-Übergang in benachbarten Dipolen anregen können. Die dipolare Kopplung der betrachteten Kerne liegt im Kilohertz-Bereich, sie kann aber aufgrund der schnellen isotropen Reorientierung nicht beobachtet werden; hieraus entsteht jedoch ein Relaxationsmechanismus. Hierbei ist für strukturelle Untersuchungen interessant,

[8]Diese Art von Fluktuation bezieht sich auf einen punktförmigen Beobachter (Kern), welcher einen rotierenden Dipol sieht. Da das Dipol-Feld somit verschiedene Orientierungen zum Beobachter einnimmt, erscheint dies als fluktuierend.

q	p	$\hat{A}_p^q$	$\hat{A}_p^{-q}$
0	0	$\frac{2}{\sqrt{6}}\hat{I}_{z1}\hat{I}_{z2}$	$\frac{2}{\sqrt{6}}\hat{I}_{z1}\hat{I}_{z2}$
0	1	$-\frac{1}{2\sqrt{6}}\hat{I}_1^+\hat{I}_2^-$	$-\frac{1}{2\sqrt{6}}\hat{I}_1^-\hat{I}_2^+$
1	0	$-\frac{1}{2}\hat{I}_{z1}\hat{I}_2^+$	$\frac{1}{2}\hat{I}_{z1}\hat{I}_2^-$
1	1	$-\frac{1}{2}\hat{I}_1^+\hat{I}_{z2}$	$\frac{1}{2}\hat{I}_1^-\hat{I}_{z2}$
2	0	$\frac{1}{2}\hat{I}_1^+\hat{I}_2^+$	$\frac{1}{2}\hat{I}_1^-\hat{I}_2^-$

TABELLE 3.2: Dipolare Spinoperatoren [14]

daß die Effizienz der dipolaren Relaxation nicht nur von den beteiligten kernmagnetischen Momenten abhängt, sondern auch umgekehrt proportional zur sechsten Potenz des Abstandes der betrachteten Kerne ist. Da Protonen ein großes gyromagnetisches Verhältnis besitzen, ist die dipolare Relaxation die für die ^{1}H-NMR-Spektroskopie wichtigste Relaxationsart.

Um nach Gleichung (3.12) den Relaxations-Superoperator für die Dipol-Dipol-Relaxation bestimmen zu können, muß man die räumliche Funktion c_0 aus Gleichung (3.17) kennen. Sie ergibt sich zu [14]

$$c_0 = -\frac{\sqrt{6}\mu_0\hbar}{4\pi}\,\frac{\gamma_1\gamma_2}{r_{12}^3}. \tag{3.24}$$

Darüberhinaus müssen die Spinoperatoren $\hat{A}_p^{\pm q}$ bekannt sein, um die entsprechenden Doppelkommutatoren in Gleichung (3.12) berechnen zu können. Sie sind in Tabelle 3.2 zusammengefaßt.

3.3.2 Relaxation durch anisotrope chemische Verschiebung

Eine generell bei der NMR-Spektroskopie auftretende Eigenschaft ist, daß die beobachtete Resonanzfrequenz eines Kernes von der lokalen Umgebung des betrachteten Kerns abhängt. Die auftretenden Unterschiede in den Resonanzfrequenzen bezeichnet man als chemische Verschiebung. Diese benutzt man dazu, um zwischen Kernen in verschiedenen chemischen Umgebungen unterscheiden zu können.

q	p	$\hat{A}_p^q$	$\hat{A}_p^{-q}$
0	0	$\frac{2}{\sqrt{6}}\hat{I}_{z1}$	$\frac{2}{\sqrt{6}}\hat{I}_{z1}$
1	0	$-\frac{1}{2}\hat{I}_1^+$	$\frac{1}{2}\hat{I}_1^-$

TABELLE 3.3: CSA-Spinoperatoren [14]

Das Phänomen der chemischen Verschiebung entsteht dadurch, daß die durch das äußere Magnetfeld induzierten dipolaren Felder der Elektronen zu sekundären Magnetfeldern führen. Dadurch wirkt am Kernort ein effektives Magnetfeld, welches sich aus der Summe des äußeren statischen und der sekundären Magnetfelder ergibt. Hierbei kann der Einfluß der sekundären Felder das äußere Magnetfeld verstärken oder abschwächen. Im allgemeinen ist die elektronische Ladungsverteilung anisotrop, was durch einen in einer 3×3 Matrix darstellbaren Abschirmungstensor κ beschrieben werden kann. Hierfür läßt sich eine Transformation in eine diagonale Matrixdarstellung mit den Komponenten κ_{xx}, κ_{yy} und κ_{zz} finden. Diese Komponenten haben folgende Bedeutung: Wenn ein Molekül so angeordnet ist, daß die k-te Achse längs der z-Achse des statischen Magnetfeldes ausgerichtet ist, dann ergibt sich das am Kernort wirkende Magnetfeld durch

$$\vec{B}_{\text{eff}} = (1 - \kappa_{kk})\vec{B}_0. \tag{3.25}$$

Chemische Verschiebungen spiegeln somit die elektronischen Einflüsse wider, welche die lokalen magnetischen Felder an einem Kernort beeinflussen. Da die chemische Verschiebung anisotrop[9] ist, verändern sich die Komponenten der lokalen Felder durch die Molekülbewegung. Diese variierenden lokalen Magnetfelder führen zu einem möglichen Relaxationsmechanismus.

Die räumliche Funktion c_0 aus Gleichung (3.17) ergibt sich für CSA-Relaxation zu [14]

$$c_0 = \sqrt{\frac{2}{3}}\,(\sigma_\parallel - \sigma_\perp)\,\gamma_1 B_0, \tag{3.26}$$

[9]Im englischen wird die Anisotropie der chemischen Verschiebung als „chemical shift anisotropy“bezeichnet. Daher kürzt man den darauf beruhenden Relaxationsmechanismus als CSA-Relaxation (*C*hemical *S*hift *A*nisotropy) ab.

falls der chemische Abschirmungstensor als axialsymmetrisch angenommen wird, d.h. $\kappa_{\parallel} = \kappa_{zz}$ und $\kappa_{\perp} = \kappa_{xx} = \kappa_{yy}$. Die benötigten CSA-Spinoperatoren $\hat{A}_p^{\pm q}$ sind in Tabelle 3.3 zusammengestellt.

Kapitel 4

Theorie der Steuerung dynamischer Systeme

Die Theorie der Steuerung dynamischer Systeme (*Optimum Control Theory*) beschäftigt sich allgemein damit, die Bewegung von dynamischen Systemen zu optimieren. In der NMR-Spektroskopie entspricht das Spinsystem dem zu steuernden dynamischen System, dessen Beschreibung im Dichtematrixformalismus möglich ist (vergl. Kapitel 2 und 3). Durch die Verbindung von Steuerungstheorie und Quantenmechanik erhält man einen Formalismus, der es ermöglicht, NMR-Experimente zu entwickeln, die unter Erfüllung des notwendigen Optimalitätskriteriums die Dichtematrix aus einem Startzustand in einen definierten Zielzustand überführen. In diesem Kapitel sollen die grundlegenden Ideen zur Anwendung der Steuerungstheorie auf die NMR-Spektroskopie dargestellt werden. Hierbei wird allerdings auf weiterführende mathematische Beweise verzeichtet, weil dies den Rahmen dieser Arbeit sprengen würde. An dieser Stelle sei auf die grundlegende Literatur zur Optimierungs-Theorie [33]-[85] verwiesen.

4.1 Die Steuerbarkeit

4.1.1 Allgemeine Definition

Damit die Bewegung eines Systems durch die Steuerungstheorie erfaßt werden kann, muß das System die Voraussetzung der Steuerbarkeit erfüllen. Grundsätzlich läßt sich dieser Begriff auf drei Kriterien zurückführen.

- **Existenz einer Bewegungsgleichung**: Die Steuerung eines Systems ist nur dann möglich, wenn es sich bei diesem um ein dynamisches System handelt. D.h. die Bewegung des Systems kann durch eine zeitabhängige Differentialgleichung der allgemeinen Form

$$\dot{x} = f(x, \ldots). \tag{4.1}$$

beschrieben werden. Gemäß (4.1) verfügt das System über verschiedene Systemzustände $x(t)$. Ist diese Voraussetzung erfüllt, so kann man das Steuerungsproblem prinzipiell als die Überführung des Systems aus einem Startzustand $x(t_0) = x^0$ in einen Zielzustand $x(t_1) = x^1$

$$x^0 \longrightarrow x^1 \tag{4.2}$$

formulieren.

- **Existenz von äußeren Kontrollvariablen**: Um das System nun steuern zu können, muß die Bewegungsgleichung (4.1) um äußere Kontrollvariablen $u_1(t), u_2(t), \ldots, u_n(t)$ erweiterbar sein[1]. Durch die Wahl dieser Terme ist es möglich, die Bewegung des Systems zu jeder Zeit t zu beeinflussen und so letzendlich auf einem kontrollierten Pfad $x(t)$ vom Start- zum Zielzustand zu steuern. Die Bewegungsgleichung des zu steuernden dynamischen Systems hat somit die Form

$$\dot{x} = f(x, u). \tag{4.3}$$

- **Existenz von Kontrollpfaden**: Damit die Bewegung des System nun durch die äußeren Kontrollvariablen u kontrolliert werden kann, müssen die Kontrollvariablen u wählbar sein, d.h. es müssen mehrere mögliche Kontrollpfade $x_i(t) = \mathcal{X}^i$ existieren. Die Wahl der Kontrollen legt die Bewegung auf einem Kontrollpfad $x(t) \in \mathcal{X}^i$ fest.

Ein einfaches Beispiel für ein kontrollierbares System stellt das in Abbildung 4.1 gezeigte Spiel dar. Die Spielebene kann kontrolliert um zwei orthogonale Achsen gekippt werden (äußere Kontrolle durch Beschleunigung

[1] Ohne Beschränkung der Allgemeinheit wird im weiteren Verlauf die Gesamtheit der Kontrollvariablen $\sum_{i=1}^{n} u_i(t)$ als u bezeichnet.

ABBILDUNG 4.1: Beispiel eines kontrollierbaren Systems. In diesem Geschicklichkeitsspiel wird die Bewegung der Spielkugel durch das Kippen der Ebene kontrolliert. Durch die Neigung der Ebene erfährt die Kugel variable Parallel- bzw. Seitenbeschleunigungen $a(t)$, wodurch die Geschwindigkeit und Richtung von außen steubar ist.

der Kugel im Gravitationsfeld der Erde), wodurch die Geschwindigkeit und Richtung der Spielkugel (beschrieben durch die Differentialgleichung der Kinematik) gesteuert werden kann. Durch geschicktes Einsetzten der äußeren Kontrollen soll es dem Spieler ultimativ gelingen, die Kugel auf einem Weg kontrolliert von einem Startpunkt x^0 zu einem Endpunkt x^1 zu bewegen, ohne daß die Kugel auf dem Weg in ein Loch fällt.

Aus dem bisher gesagten läßt sich der Begriff der Steuerbarkeit formalisieren [34]:

Defintion 4.1 *Gegeben sei ein dynamisches System, dessen Bewegung durch eine Differentialgleichung $f(x, u)$ beschrieben werden kann. Hierbei seien x die Zustands- und u die Kontrollvariablen.*

Man bezeichnet die Bewegung eines solchen Systems von einem Ausgangszustand x^0 zu einem Endzustand x^1 als steuerbar, wenn mehrere mögliche Wege zwischen diesen Punkten bestehen. Die Auswahl des benutzen Weges ist ausschließlich durch systemunabhängige (äußere) Kontrollen möglich.

4.1.2 Übertrag auf die NMR

Das dynamische System der NMR-Spektroskopie wird durch das Spinsystem dargestellt, d.h. die Zustandsvariable entspricht der Dichtematrix σ des Spin-

systems. Allgemein läßt sich das Ziel von Kohärenztansfer-Experimenten als die Überführung des Spinsystems ausgehend von einem (präparierten) Startzustand $\sigma(t_0) = \sigma^0$ in einen definierten Endzustand $\sigma(t_1) = \sigma^1$ formulieren:

$$\sigma^0 \longrightarrow \sigma^1. \tag{4.4}$$

Die Bewegung dieses Systems wird durch die Liouville-von Neumann Gleichung

$$\dot{\sigma} = -\imath[\mathcal{H}_{\text{tot}}, \sigma] \tag{4.5}$$

beschrieben. Hierbei wird die Bewegung durch den Gesamt-Hamilton-Operator $\mathcal{H}_{\text{tot}}$ bestimmt. Dieser läßt sich in einen systemabhängigen (*drift*: $\mathcal{H}_{\text{d}}$) und einen systemunabhängigen (*control*: $\mathcal{H}_{\text{c}}$) Anteil zerlegen. Der Drift-Anteil entspricht der intrinsischen Dynamik des Spinsystems (z.B. Kopplungs-Hamilton-Operator, Offset-Hamilton-Operator). Der steuerbare Anteil wird durch die RF-Felder dargestellt, durch die die Magnetisierung kontrolliert rotiert werden kann. Hierbei stellt die RF-Amplitude der einzelnen Felder (mit der Phase $\nu \in x, y$) die äußere Kontrollvariable[2] dar. Somit läßt sich unter Verwendung von

$$\begin{aligned} \mathcal{H}_{\text{tot}} &= \mathcal{H}_{\text{d}} + \mathcal{H}_{\text{c}} \\ &= \mathcal{H}_{\text{d}} + \sum_{i=1}^{n} 2\pi u_i(t) I_{\nu,i} \end{aligned} \tag{4.6}$$

die geforderte Form (4.3) der Bewegungsgleichung (4.5) zeigen:

$$\dot{\sigma} = -\imath \left[\mathcal{H}_{\text{d}} + \sum_{i=1}^{N} 2\pi u_i(t) I_{\nu,i}, \sigma \right] \equiv f(\sigma, u). \tag{4.7}$$

Somit erfüllt das NMR-System σ die allgemeinen Bedingungen der Steuerbarkeit.

[2]Die Verwendung von RF-Felder als Kontrollen hat einen Einfluß auf die Differnzierbarkeit. Die RF-Felder sollen unstetig sein können, d.h. die RF-Amplitude soll sich sprunghaft ändern dürfen. Darüberhinaus verfügen die Kontrollen aufgrund ihres physikalischen Erzeugungsprozesses über einen eingeschränkten Wertebereich. Die in diesem Kapitel gewählte Darstellung setzt allerdings stetig differenzierbare Funktionen voraus (klassischen Variationsrechnung). Daher können die gezeigten Berechnungen nicht als mathematischer Beweis gewertet werden, die gezeigten Ergebnisse sind dennoch korrekt. Auf die Beweisführung mittels nicht-klassischer Variationsrechnung wurde verzichtet, weil dies über den Rahmen dieser Arbeit hinausgeht.

4.2 Die Kostenfunktion

4.2.1 Allgemeine Definition

Wie in Abschnitt 4.1.1 beschrieben, wird für die Steuerbarkeit eines dynamischen Systems vorausgesetzt, daß für das allgemeine Steuerungsproblem (4.2) mehrere Kontrollpfade $x_i(t)$ exisitieren. Ziel der optimalen Steuerung ist es, den besten Pfad $x^\star(t)$ zu identifizieren. Es gilt also ein Kriterium zu entwickeln, durch das die Güte der verschiedenen Kontrollpfade ausdrückt werden kann, die sog. Kostenfuntkion $\mathcal{J}$. Durch die Minimierung eines solchen Kriteriums kann man schließlich Kontrollvariablen erhalten, die den kostengünstigsten Kontrollpfad wählen.

Prinzipiell kann man zwei Anteile für die Kosten eines Pfades definieren:

- **laufende Kosten** (*running cost*): Die Pfade unterscheiden sich durch ihre Eigenschaften während des Transfers. Die Formulierung ist stets vom betrachteten Problem abhängig. So kann ein Transferprozeß z.B. Energie benötigen und man möchte den Kontrollpfad identifizieren, der die geringste Energie verbraucht. Ein anderes Beispiel wäre die Minimierung der zurückgelegte Wegstreckte zwischen zwei Punkten. Eine Eigenschaft ist bei allen Kriterien identisch. Sie sind ein über die Transferzeit gebildetes Wegintegral. Somit läßt sich die entsprechende Funktion stets als

  $$\mathcal{J}_\mathrm{r} = \int_{t_0}^{t_1} \mathcal{L}(x(t), u(t))dt \tag{4.8}$$

  darstellen. Die spezielle Form der Funktion $\mathcal{L}(x(t), u(t))$ wird durch das Optimierungsziel bestimmt.

- **abschließende Kosten** (*final cost*): Durch einen Kontrollpfad $x(t)$ wird das System in einen Endzustand $x(t_1)$ überführt. Ein abschließendes Kostenkriterium ist z.B. die Nähe des erreichten zum geforderten Zielzustand x^1. Wie bereits bei den laufenden Kosten ist die spezielle Formulierung dieses Kriteriums von dem Optimierungsproblem abhängig. Einzig die Definition eines zu minimierenden Kriteriums für das Ende des Transfers ist ausreichend für eine abschließenden Kostenfuntkion, d.h. die Definition einer Funktion

  $$\mathcal{J}_\mathrm{f} = \mathcal{L}(x(t_1)) \tag{4.9}$$

führt die Minimierung der abschließenden Kosten in das Optimierungsproblem ein.

Aus diesen beiden Funktionen läßt sich nun eine Gesamt-Kostenfunktion

$$\mathcal{J} = \mathcal{J}_\mathrm{r} + \mathcal{J}_\mathrm{f} \tag{4.10}$$

bilden. Ein Kontrollpfad wird dann als optimal bezeichnet, wenn durch ihn die Kostenfunktion minimiert wird. Somit läßt sich das allgemeine Optimierugnsproblem (4.2) in

$$x^0 \overset{\min(\mathcal{J})}{\longrightarrow} x^1 \tag{4.11}$$

umschreiben.

4.2.2 Übertrag auf die NMR

Wie bereits festgestellt, ist die Kostenfunktion stets vom speziellen Optimierungsproblem abhängig. Hier werden die Kosten so beschrieben, wie sie generell im Rahmen dieser Arbeit verwendet wurden[3].

Für die Optimierung von NMR-Transferexperimenten stellt die Maximierung der Projektion des erreichten auf den Zielzustand ein wichtiges Kriterium dar, wodurch sich die abschließende Kostenfunktion z.B. als

$$\mathcal{L}(x(t_1)) = \langle \sigma(t_1) | \sigma^1 \rangle \tag{4.12}$$

darstellen läßt.

Die laufenden Kosten eines Transfers können z.B. durch die während des Experimentes eingestrahlte Energie formuliert werden, die minimiert werden soll. Im Rahmen dieser Arbeit wird die Energiebeschränkung nicht durch eine Kostenfunktion, sondern durch einen beschränkten Wertebereich der Kontrollen formuliert. Diese Formulierung hat den Vorteil, daß die erzeugten RF-Frequenzen eine bestimmte Grenze nicht überschreiten, was durch die integrale Darstellung nicht gewährleistet wird. Somit ergeben sich die laufenden Kosten zu

$$\mathcal{L}(x(t), u(t)) = 0. \tag{4.13}$$

[3]Im Falle der in Kapitel 7 vorgestellten Optimierung war es sinnvoll, eine für das Problem speziell angepaßte Kostenfunktion zu verwenden. Diese wird dort detailliert besprochen.

4.3 Das Pontryaginsche Maximum Prinzip

4.3.1 Allgemein

Um nun den optimalen Pfad zu identfizieren ist es nach (4.11) notwendig, die Kostenfunktion des Transfers zu minimieren. Die notwendige Bedingung für eine optimale Kontrolle $u^\star$ ist, daß die Variation

$$\delta\mathcal{J} = \frac{\partial\mathcal{J}}{\partial x}\delta x + \frac{\partial\mathcal{J}}{\partial u}\delta u = 0 \tag{4.14}$$

ist. Allerdings stellt sich nun das Problem, daß δx und δu nicht unabhängig voneinander sind, sondern durch die Bewegungsgleichung miteinander verknüpft sind. Formal bedeutet dies, daß es sich um ein durch eine Nebenbedingung beschränktes Minimierungsproblem handelt. Allgemein kann man ein solch beschränktes Problem $\min(f)$ durch die Einführung von Lagrange-Faktoren λ und einer Nebenbedingungs-Funktion g in ein unbeschränktes Problem $\min(f + \lambda g)$überführen. Um nun die Variation $\delta\mathcal{J}$ bestimmen zu können, muß zunächst die durch die Zustandsgleichung gegebene Nebenbedingung formuliert werden. Da die Bewegungsgleichung zu jeder Zeit t während des Transfers erfüllt sein muß, kann der Ausdruck

$$\Phi = \int_{t_0}^{t_1} \lambda(t)\left(f(x(t),u(t)) - \dot{x}(t)\right)dt \tag{4.15}$$

als Nebenbedinungs-Gleichung in (4.10) eingeführt werden, wodurch (4.14) in ein unbeschränktes Problem überführt wird. Die zu minimierende Funktion ergibt sich nun zu

$$\begin{aligned}\mathcal{J} &= \mathcal{J}_\mathrm{r} + \mathcal{J}_\mathrm{f} + \Phi \\ &= \mathcal{L}(x(t_1)) \\ &\quad + \int_{t_0}^{t_1} \mathcal{L}(x(t),u(t)) + \lambda(t)f(x(t),u(t)) - \lambda(t)\dot{x}(t)dt.\end{aligned} \tag{4.16}$$

Durch Einführen der Hamilton-Funktion

$$\mathsf{H}(x,u) = \mathcal{L}(x(t),u(t)) + \lambda(t)f(x(t),u(t)) \tag{4.17}$$

und partielle Integration von

$$\int_{t_0}^{t_1} \lambda(t)\dot{x}(t)dt = \lambda(t_1)x(t_1) - \lambda(t_0)x(t_0) - \int_{t_0}^{t_1} \dot{\lambda}(t)x(t)dt \tag{4.18}$$

ergibt sich aus (4.16) die unbeschränkt zu minimierende Kostenfunktion zu

$$\begin{aligned}\mathcal{J} &= \mathcal{L}(x(t_1)) - \lambda(t_1)x(t_1) + \lambda(t_0)x(t_0) \\ &\quad + \int_{t_0}^{t_1} \left\{ \mathsf{H}(x,u) + \dot{\lambda}(t)x(t) \right\} dt. \end{aligned} \tag{4.19}$$

Die in (4.14) benötigten partiellen Ableitungen lassen sich nun zu

$$\frac{\partial \mathcal{J}}{\partial x} = \frac{\partial \mathcal{L}(x(t_1))}{\partial x} - \lambda(t_1) + \int_{t_0}^{t_1} \left\{ \frac{\partial \mathsf{H}(x,u)}{\partial x} + \dot{\lambda}(t) \right\} dt \tag{4.20}$$

$$\frac{\partial \mathcal{J}}{\partial u} = \int_{t_0}^{t_1} \frac{\partial \mathsf{H}(x,u)}{\partial u} \tag{4.21}$$

bestimmen, woraus sich die Variation der Kostenfunktion unter Umordnen der Terme zu

$$\begin{aligned}\delta\mathcal{J} &= \left(\frac{\partial \mathcal{L}(x(t_1))}{\partial x} - \lambda(t_1) \right) \delta x \\ &\quad + \int_{t_0}^{t_1} \left\{ \left(\frac{\partial \mathsf{H}(x,u)}{\partial x} + \dot{\lambda}(t) \right) \delta x + \frac{\partial \mathsf{H}(x,u)}{\partial u} \delta u \right\} dt \end{aligned} \tag{4.22}$$

ergibt.

Aus der notwendigen Bedingung für ein Extremum

$$\delta\mathcal{J} = 0 \tag{4.23}$$

folgt, daß jeder Summand in (4.22) ebenfalls Null sein muß. Wertet man die einzelnen Terme aus, so erhält man die Lagrange-Faktoren zum Zeitpunkt t_1

$$\lambda(t_1) = \frac{\partial \mathcal{L}(x(t_1))}{\partial x}, \tag{4.24}$$

die Bewegungsgleichung der Lagrange-Faktoren

$$\dot{\lambda}(t) = -\frac{\partial \mathsf{H}(x,u)}{\partial x} \tag{4.25}$$

und ein Gradienten-Kriterium

$$\frac{\partial \mathsf{H}(x,u)}{\partial u} = 0. \tag{4.26}$$

Wenn ein Steuerungsprozeß diese Kriterien erfüllt, so ist auch das notwendige Kriterium der Optimalität erfüllt (Pontryaginsches Maximum Prinzip).

4.3.2 Übertrag auf die NMR

Aus der Standard-Kostenfunktion (4.12) und Gleichung (4.24) ergibt sich für hermitische Matrizen

$$\begin{aligned} \lambda(t_1) &= \frac{\partial\langle\sigma(t_1)|\sigma^1\rangle}{\partial\sigma(t_1)} \\ &= \sigma^1 \end{aligned} \tag{4.27}$$

die Zieldichtematrix für die Lagrange-Faktoren bei t_1.

Die Hamiltonfunktion ergibt sich aus (4.17) unter Verwendung von (4.13) zu

$$\mathsf{H}(\sigma, u) = \langle\lambda(t)|\dot{\sigma}(t)\rangle. \tag{4.28}$$

Unter Verwendung von (4.28) kann man aus (4.25) die Bewegungsgleichung der Lagrange-Faktoren zu

$$\begin{aligned} \dot{\lambda}(t) &= -\frac{\partial\mathsf{H}(\sigma, u)}{\partial\sigma} \\ &= [-\imath\mathcal{H}_{\mathrm{tot}}, \lambda] \end{aligned} \tag{4.29}$$

bestimmen. Man beachte die Identität der Bewegungsgleichung (4.29) zur Bewegungsgleichung des Spinsystems (4.5). Darüberhinaus erhält man aus (4.26) unter Verwendung der Hamilton-Funktion (4.28) und der Bewegungsgleichung (4.7) das Gradienten-Kriterium als

$$\begin{aligned} 0 &= \frac{\partial\mathsf{H}(x, u)}{\partial u_i} \\ &= \frac{\partial}{\partial u_i}\mathrm{tr}\left\{\lambda^\dagger(t)\left[-\imath(\mathcal{H}_{\mathrm{d}} + \sum_{i=1}^{N} 2\pi u_i(t) I_{\nu,i}), \sigma(t)\right]\right\} \\ &= \mathrm{tr}\left\{\lambda^\dagger(t)\left[-\imath 2\pi u_i(t) I_{\nu,i}, \sigma(t)\right]\right\}. \end{aligned} \tag{4.30}$$

Im Falle einer optimalen Kontrolle $u^\star$ beträgt der Gradient Null, andernfalls beschreibt der Gradient die Änderungsrichtung der i-ten Kontrolle u_i, durch deren Einführung

$$u_i'(t) = u_i(t) + \varepsilon\frac{\partial\mathsf{H}(x, u)}{\partial u_i} \tag{4.31}$$

die Kostenfunktion minimiert wird. Hierbei stellt der Faktor ε die Schrittweite der Änderung in die durch den Gradienten gegebene Richtung dar.

4.3.3 Übertrag in einen Algorithmus

Aus den in Abschnitt 4.3.1 bzw. 4.3.2 festgestellten Beziehungen kann das prinzipielle Vorgehen einer (numerischen) Optimierung ersehen werden.

- **Initialsierung:**
  1. Definition des Startzustandes $x^0 = \sigma(t_0)$
  2. Definition des Zielzustandes $x^1 = \sigma(t_1)$
  3. Erzeugung einer zufälligen Start-Kontrolle $u(t)$
- **Iteration**
  1. Propagation des Startzustandes $x^0 \xrightarrow{u(t)} x(t_1)$ gemäß Gleichung (4.3) bzw. (4.5)
  2. Bestimmen der Lagrange-Faktoren $\lambda(t_1)$ aus den abschließenden Kosten gemäß Gleichung (4.24) bzw. (4.27)
  3. Propagation der Lagrange-Faktoren $\lambda(t_1) \xrightarrow{u(t)} \lambda(t_0)$ gemäß Gleichung (4.25) bzw. (4.29)
  4. Anpassen aller Kontrollen $u_i(t) = u_i(t) + \varepsilon \frac{\partial \mathsf{H}(x,u)}{\partial u_i}$ unter Verwendung von Gleichung (4.26) bzw. (4.31)

Wenn die Gradienten Null betragen, ist das Pontryaginsche Maximum Prinzip erfüllt und der Algorithmus ist konvergiert. Aus der zufälligen Startkontrolle $u(t)$ wurde eine optimale Kontrolle $u^\star(t)$ erzeugt.

Teil II

Numerik

Kapitel 5

Optimierungen für ideale Spinsysteme

Jeder Kohärenztransfer wird in seinen Transfer-Eigenschaften durch die ihm zugrunde liegende Quantenmechanik beschränkt. Eine charakteristische Grösse stellt hierbei die maximal erreichbare Transfer-Amplitude dar, die sog. unitäre Grenze η_{uni} [58]. Diese ist für jeden Kohärenztransfer bestimmbar. Berechnet man nun für alle Zeiten t die optimale Transferamplitude, so exisitiert stets eine minimale Zeit $t^{\star}$, ab der die unitäre Transfergrenze durch eine theoretisch optimale Steuerung bzw. ein Experiment erreicht werden kann, d.h. $\{\eta_{\text{opt}}(t \geq t^{\star}) = \eta^{\star}\} \equiv \eta_{\text{uni}}$. Für Zeiten $t < t^{\star}$ folgt die optimale Transfer einer zeitabhängigen Funktion $\eta_{\text{opt}}(t)$ mit $\eta_{\text{opt}} < \eta_{\text{uni}}$. In Abbildung 5.1 sind diese allgemeinen Eigenschaften von Kohärenztransfer-Funktionen schematisch dargestellt.

Um solche allgemeinen Daten bezüglich beliebiger Kohärenztransfers zu ermitteln, kann man prinzipiell zwei Ansätze verfolgen. Eine Möglichkeit besteht darin, eine Transferfunktion analytisch zu bestimmen. Für einfache Spinsysteme ist dies mit noch vertretbarem Aufwand möglich. Werden die Systeme jedoch komplexer, so ist die Bestimmung der Transferfunktionen mit numerischen Computermethoden vorzuziehen. Wenn hierbei jeder Datenpunkt durch eine Pulssequenz-Optimierung gewonnen wird, so setzt dies voraus, daß die entsprechende Sequenz die für die jeweilige Mischzeit optimale ist. Daher bezeichnet man solche numerisch bestimmten Transferfunktion auch als TOP-Kurven (Time Optimal Pulses). Im Allgemeinen wird bei der Berechnung von TOP-Kurven der Einfluß von Störtermen bewußt vernachlässigt, weil man zu jeder Zeit das theoretische Maximum eines

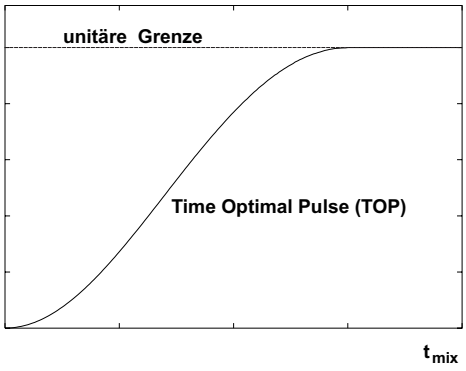

ABBILDUNG 5.1: Die Transfereigenschaften von Kohärenztransfers werden durch die Quantenmechanik begrenzt. Es gibt eine bestimmbare unitäre Grenze, die der maximal erreichbaren Transferamplitude entspricht. Diese kann durch Experimente ab einer Zeit $t^\star$ erreicht werden. Für Zeiten $t < t^\star$ ist die optimale Transferamplitude zeitabhängig. Die entprechende für alle Zeiten optimale Transferamplitude wird als TOP-Kurve bezeichnet.

Kohärenztransfers bestimmen möchte[1].

In diesem Kapitel werden numerisch bestimmte Transferfunktionen für zwei Grundtypen von Spinsystemen vorgestellt. In Abschnitt 5.1 werden TOP-Kurven für Spinsysteme des Typs I_nS mit $n \in [1, 3]$ vorgestellt, während in Abschnitt 5.2 entprechende Ergebnisse für den Transfer entlang eindimensionaler Spinketten I_n mit $n \in [2, 5]$ gezeigt werden.

Die numerische Bestimmung der TOP-Kurven folgt dabei diesem prinzipiellen Protokoll: Zunächst werden für den jeweiligen Kohärenztransfer 10 (Systeme mit $n \geq 4$ Spins) bzw. 20 (Systeme mit $n \leq 3$ Spins) Pulssequenzen bis zu einer Konvergenzschranke von $\varepsilon = 10^{-4}$ optimiert. Aus dieser Voroptimierung wird die Pulssequenz mit den besten Transfereigenschaften als Startkontrolle für die finale Optimierung ausgewählt, wobei die Konvergenzschranke bei $\varepsilon = 10^{-6}$ liegt. Durch diesen statistischen Ansatz senkt sich die Wahrscheinlichkeit, daß sich der Optimierungs-Algorithmus durch einen Hyperflächen-Bereich geringer Steigung auf das absolute Maximum bewegt, wodurch zwei Effekte vermieden werden können. Zum einen besteht die Möglichkeit, daß durch die geringe Steigung das Konvergenzkriterium

[1] Die Vernachlässigung von Störtermen bedeutet, daß ein nicht relaxierendes, vollständig *on-resonantes* Spinsystem verwendet wird. Die erzeugten RF-Felder sind darüberhinaus ohne Inhomogenitäts-Fehler. Erst durch Annahme dieser idealen Bedingungen können theoretische Transfergrenzen bestimmt werden.

viel zu früh erfüllt wird, d.h. das absolute Maximum wird nicht erreicht. Wird die Konvergenzschranke in diesem Bereich jedoch nicht unterschritten, so verringert sich die Konvergenzgeschwindigkeit, was wiederum zu einem größeren Bedarf an Rechenzeit führt. Durch die Bestimmung eines größeren Satzes an voroptimierten Sequenzen können solche Fälle mit einer größeren Wahrscheinlichkeit für die Hauptoptimierung ausgeschlossen werden, d.h diese wird mit größeren Wahrscheinlichkeit nahe an das absolute Transfermaximum konvergieren.

Darüberhinaus gilt zu beachten, daß die Kontrollen genügend Freiheitsgrade benötigen, damit die Optimierung gegen die maximale Transferamplitude einer Pulssequenz konvergieren kann. Um dies bei den hier vorgestellen Optimierungen zu gewährleisten, wurde jede Kontrolle in 400 Datenpunkte digitalisiert.

5.1 Kohärenztransfer in I_nS-Spinsystemen

5.1.1 Theorie

Bei den in diesem Abschnitt untersuchten Spinsystemen handelt sich in der Regel um heteronukleare Systeme, z.B. CH_n-Gruppen mit $n \in [1, 3]$. Daher ist es für die Beschreibung der systemischen Dynamik (*Drift*[2]) ausreichend, ausschließlich den Hamilton-Operator der schwachen Kopplung

$$\mathcal{H}_{\text{weak}} = 2\pi J S_z \sum_{1}^{n} I_{n,z} \tag{5.1}$$

zu berücksichtigen.

Die auf diese Spinsysteme anwendbaren Kontrollen können in zwei Klassen eingeteilt werden. Da es sich bei den I- und S-Spins eines Systems um unterschiedliche Kernspins handelt, werden diese stets durch getrennte RF-Kanäle angesprochen. In der Sprache der Steuerungstheorie bedeutet dies, daß die I- und S-Spins durch selektive Kontrollen gesteuert werden. Bei I_nS-Systemen mit $n > 1$ besteht nun zusätzlich die Möglichkeit, daß alle I-Kernspins ebenfalls durch selektive RF-Felder manipuliert werden können.

[2]Da die Optimierungen keine Offset-Effekte berücksichtigen, entspricht der Kopplungs-Hamiltonian dem Drift-Hamiltonian.

Der entsprechenden vollständig selektive Kontroll-Hamiltonian

$$\mathcal{H}_{\text{sel}} = 2\pi \left\{ \nu_{0,x}(t) S_x + \nu_{0,y}(t) S_y + \sum_1^n \nu_{n,x}(t) I_{n,x} + \sum_1^n \nu_{n,y}(t) I_{n,y} \right\} \quad (5.2)$$

verfügt dementsprechend über $2(1+n)$ Kontrollfelder $\nu_{n,\varphi}(t)$ mit $n \in [0,3]$ und $\varphi \in \{x,y\}$. Diesem Fall der maximalen Steuerbarkeit eines Spinsystems steht in der Realität häufig der Fall gegenüber, daß man diese spinselektiven RF-Pulse nicht oder nur unter sehr großem Aufwand realisieren kann. Der Fall vollständig I-unselektiver RF-Pulse wird durch einen Kontroll-Hamiltonian

$$\mathcal{H}_{\text{unsel}} = 2\pi \left\{ \nu_{0,x}(t) S_x + \nu_{0,y}(t) S_y + \nu_{1,x}(t) \mathrm{F}_x + \nu_{1,y}(t) \mathrm{F}_y \right\} \quad (5.3)$$

beschrieben. Dieser Term verfügt entsprechend nur über vier Kontrollfelder $\nu_{n,\varphi}(t)$ mit $n \in [0,1]$ und $\varphi \in \{x,y\}$.

Unter Verwendung der zwei möglichen Gesamt-Hamiltonians

$$\mathcal{H}_{\text{tot},1} = \mathcal{H}_{\text{weak}} + \mathcal{H}_{\text{sel}} \quad (5.4)$$

$$\mathcal{H}_{\text{tot},2} = \mathcal{H}_{\text{weak}} + \mathcal{H}_{\text{unsel}} \quad (5.5)$$

sollen nun jeweils TOP-Kurven für zwei in der mehrdimensionalen Kernresonanz-Spektroskopie häufig auftretenden heteronuklearen Transferschritte berechnet werden, speziell für den Transfer von Antiphase-Kohärenz

$$2\mathrm{F}_z S^- \xrightarrow{\tau} \mathrm{F}^- \quad (5.6)$$

und Inphase-Kohärenz

$$S^- \xrightarrow{\tau} \mathrm{F}^-. \quad (5.7)$$

5.1.2 Numerische Ergebnisse

Die Ergebnisse für das 2-Spinsystem IS sind in Abbildung 5.2 gezeigt (Tafel 5.2.A: Antiphase-Transfer, Tafel 5.2.B: Inphase-Transfer). Da in diesem System nur ein I-Kern enthalten ist, entfällt die Unterscheidung selektiver und unselektiver Kontrollen für diese Kernsorte. Die folgenden Abbildungen beziehen sich auf die höheren Spinsysteme $\mathrm{I}_n\mathrm{S}$ (Antiphase-Transfer: Abbildungen 5.3 ($n=2$) und 5.5 ($n=3$); Inphase-Transfer: Abbildungen 5.4 ($n=2$) und 5.6 ($n=3$)), wobei nun die Seletivität der I-Pulse unterschieden werden

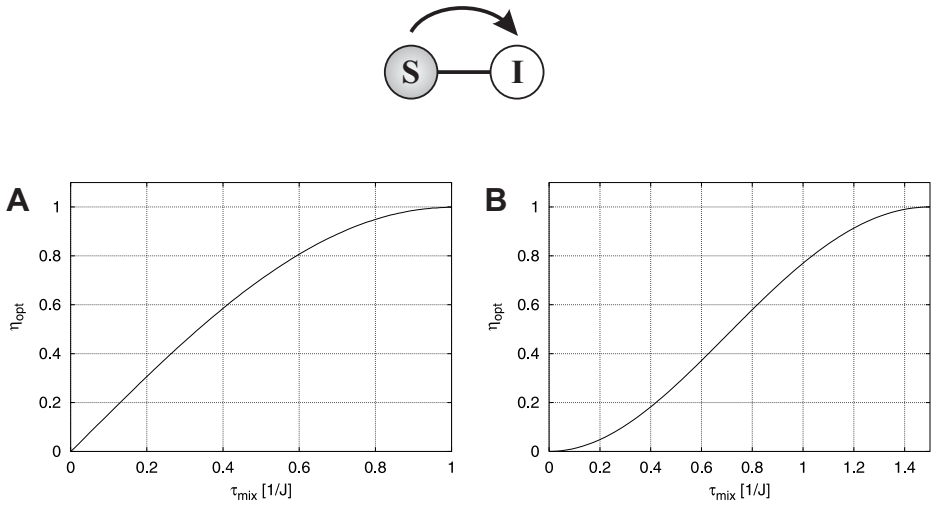

ABBILDUNG 5.2: Numerisch bestimmte Aufbaukurven für den Kohärenztransfer in gekoppelten 2-Spinsystemen. (A) Transfer von Antiphase-Magnetisierung $2I_zS^- \rightarrow I^-$, schwache Kopplung, selektive Pulse. (B) Transfer von Inphase-Magnetisierung $S^- \rightarrow I^-$, schwache Kopplung, selektive Pulse.

kann. Alle Abbildungen folgen der Konvention, daß zur Berechnung der in Tafel A gezeigten TOP-Kurve I-selektive Pulse verwendet wurden, während Tafel B auf der Annahme I-unselektiver Pulse beruht.

Die den TOP-Kurven zu entnehmenden charakteristischen Transfer-Daten sind in Tabellen 5.1 (Antiphase-Transfer) und 5.2 (Inphase-Transfer) zusammengestellt. Die unitären Grenzen für die hier numerisch untersuchten Kohärenztransfers wurden bereits in [48] bestimmt. Durch die Numerik werden im Rahmen der numerischen Genauigkeit Pulssequenzen generiert, die die unitäre Grenze erreichen (die maximale Abweichung $(\eta_{\text{uni}} - \eta^\star)$ liegt in der Größenordnung von 10^{-3}). Neben der Erkenntnis, daß es Pulssequenzen gibt, die als maximale Transferamplitude die unitäre Grenze besitzen, kann man den Kurven zusätzlich die minimale Zeit $t^\star$ entnehmen, in der diese Transfergrenze noch durch eine Sequenz realisiert werden kann[3]. Hierbei bestimmt sich die Genauigkeit durch die digitale Auflösung des Zeitvektors; bei den hier gezeigten Ergebnissen betrug diese $10^{-2}/J$.

[3]An dieser Stelle sei darauf hingewiesen, daß die numerisch gewonnenen Daten $\eta^\star$ und $t^\star$ ausschließlich für den durch die Optimierung betrachteten Zeitraum gelten.

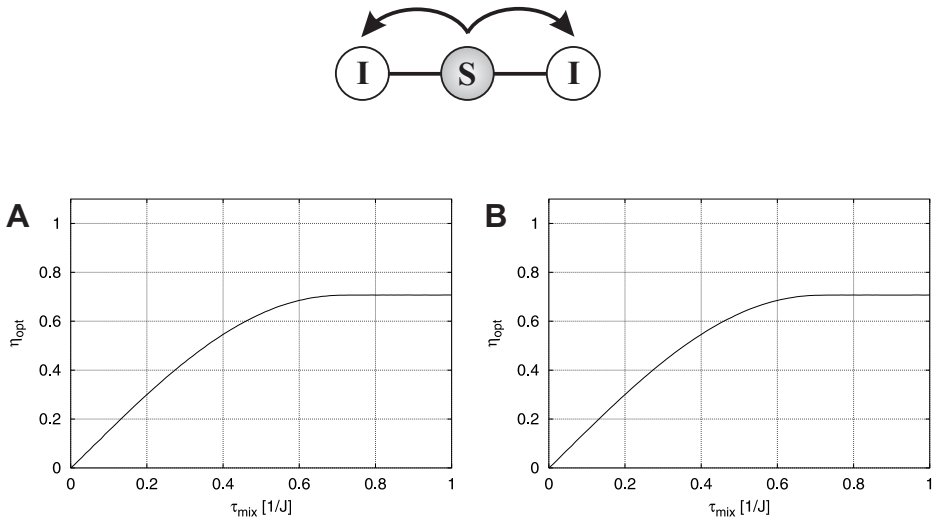

ABBILDUNG 5.3: Numerisch bestimmte Aufbaukurven für den Transfer von Antiphase-Magnetisierung $2F_zS^- \rightarrow F^-$ in einem gekoppelten 3-Spinsystem. (A) Schwache Kopplung, selektive Pulse. (B) Schwache Kopplung, unselektive Pulse.

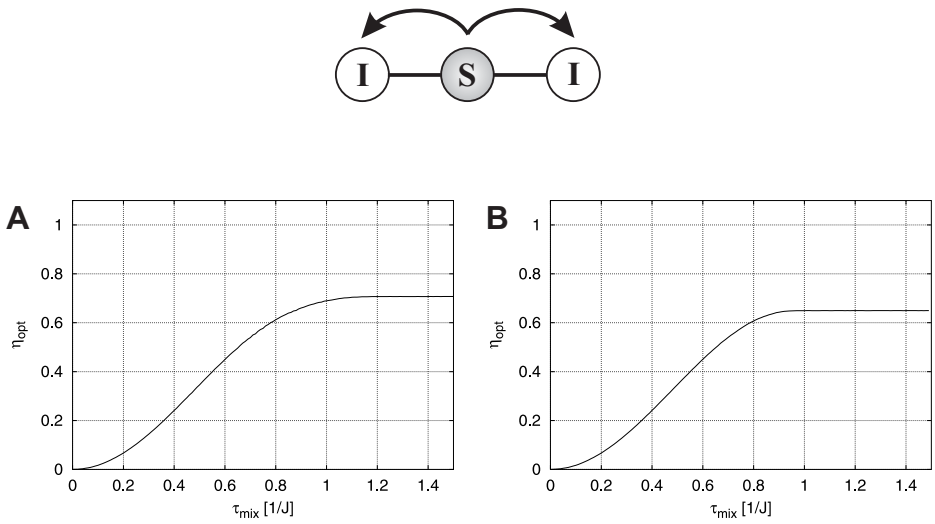

ABBILDUNG 5.4: Numerisch bestimmte Aufbaukurven für den Transfer von Inphase-Magnetisierung $S^- \rightarrow F^-$ in einem gekoppelten 3-Spinsystem. (A) Schwache Kopplung, selektive Pulse. (B) Schwache Kopplung, unselektive Pulse.

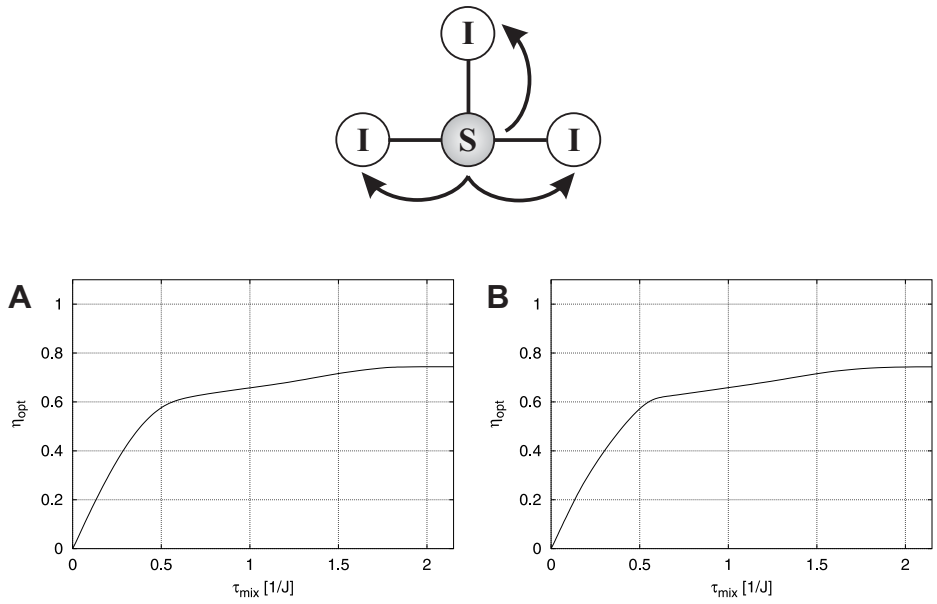

ABBILDUNG 5.5: Numerisch bestimmte Aufbaukurven für den Transfer von Antiphase-Magnetisierung $2F_zS^- \rightarrow F^-$ in einem gekoppelten 4-Spinsystem. (A) Schwache Kopplung, selektive Pulse. (B) Schwache Kopplung, unselektive Pulse.

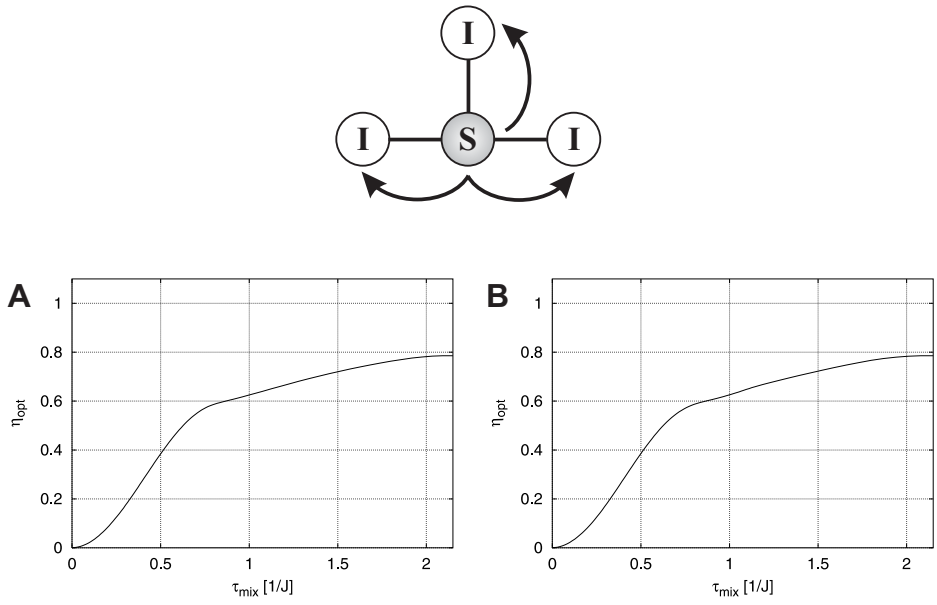

ABBILDUNG 5.6: Numerisch bestimmte Aufbaukurven für den Transfer von Inphase-Magnetisierung $S^- \rightarrow F^-$ in einem gekoppelten 4-Spinsystem. (A) Schwache Kopplung, selektive Pulse. (B) Schwache Kopplung, unselektive Pulse.

n	$\nu_{\mathrm{RF}}(I)$	$\eta^\star$ [-]	$t^\star$ $[1/J]$	η_{uni} [-]
1	selektiv	1	1	1
	unselektiv	-	-	-
2	selektiv	$0,707$	$0,707$	$\frac{1}{\sqrt{2}} = 0,707$
	unselektiv	$0,707$	$0,707$	$\frac{1}{\sqrt{2}} = 0,707$
3	selektiv	$0,744$	2	$\frac{1+2\sqrt{3}}{6} = 0,744$
	unselektiv	$0,744$	2	$\frac{1+2\sqrt{3}}{6} = 0,744$

TABELLE 5.1: Zusammenfasung der Daten bezüglich des Transfers von Antiphase-Kohärenz $2\mathrm{F}_z S^- \rightarrow \mathrm{F}^-$ in $\mathrm{I}_n\mathrm{S}$-Systemen. Aufgeführt sind die numerisch bestimmten Werte für die maximale Transferamplitude $\eta^\star$ und die minimale Zeit $t^\star$, die eine Pulssequenz für deren Umsetzung benötigt. Zum Vergleich sind die theoretisch bestimmte unitäre Grenze η_{uni} [48] angegeben.

n	$\nu_{\mathrm{RF}}(I)$	$\eta^\star$ [-]	$t^\star$ $[1/J]$	η_{uni} [-]
1	selektiv	1	$1,5$	1
	unselektiv	-	-	-
2	selektiv	$0,707$	$1,15$	$\frac{1}{\sqrt{2}} = 0,707$
	unselektiv	$0,649$	$0,95$	$\frac{3\sqrt{3}}{8} = 0,650$
3	selektiv	$0,787$	2	$\frac{3+\sqrt{3}}{6} = 0,789$
	unselektiv	$0,787$	2	$\frac{3+\sqrt{3}}{6} = 0,789$

TABELLE 5.2: Zusammenfasung der Daten bezüglich des Transfers von Inphase-Kohärenz $S^- \rightarrow \mathrm{F}^-$ in $\mathrm{I}_n\mathrm{S}$-Systemen. Aufgeführt sind die numerisch bestimmten Werte für die maximale Transferamplitude $\eta^\star$ und die minimale Zeit $t^\star$, die eine Pulssequenz für deren Umsetzung benötigt. Zum Vergleich sind die theoretisch bestimmte unitäre Grenze η_{uni} [48] angegeben.

5.2 Kohärenztransfer in Spinketten

5.2.1 Theorie

In diesem Abschnitt werden TOP-Kurven bezüglich des Kohärenztransfers in Spinketten vorgestellt. Hierunter versteht man aus n Spin-1/2-Teilchen aufgebaute Spinketten I_n, in denen Kopplungen nur zwischen den nächsten Nachbarn bestehen. Da hierbei keine Beschränkung in der Art der Kopplung besteht, werden im Folgenden die Grenzfälle der schwachen Kopplung

$$\mathcal{H}_{\mathrm{weak}} = 2\pi \sum_{i}^{n-1} J_{i,i+1} I_{i,z} I_{i+1,z} \tag{5.8}$$

und der isotropen Kopplung

$$\mathcal{H}_{\mathrm{iso}} = 2\pi \sum_{i}^{n-1} J_{i,i+1} \left(I_{i,x} I_{i+1,x} + I_{i,y} I_{i+1,y} + I_{i,z} I_{i+1,z} \right) \tag{5.9}$$

betrachtet. Das Optimierungsziel besteht gemäß

$$I_1^- \xrightarrow{\tau} I_n^- \tag{5.10}$$

darin, -1-Quatenkohärenz von einem Ende der Kette zum anderen zu transferieren. Wie bereits in Abschnit 5.1 sollen auch zwei Grenzfälle der Pulsselektivität betrachtet werden. Sind alle Spins der Kette selektiv bepulsbar, so erhält man einen Kontroll-Hamiltonian

$$\mathcal{H}_{\mathrm{sel}} = 2\pi \sum_{i}^{n} \left(\nu_{i,x}(t) I_{i,x} + \nu_{i,y}(t) I_{i,y} \right), \tag{5.11}$$

der über $2n$ Kontrollfelder verfügt. In dem Fall, daß kein Spin selektiv manipulierbar ist, reduziert sich dieser zu

$$\mathcal{H}_{\mathrm{unsel}} = 2\pi \left(\nu_x(t) \mathrm{F}_x + \nu_y(t) \mathrm{F}_y \right), \tag{5.12}$$

wobei nur 2 Kontrollen auf das System wirken können.

Die Kombination dieser Operatoren führt zu vier unterschiedlichen Gesamt-Hamiltonians

$$\mathcal{H}_{\mathrm{tot,A}} = \mathcal{H}_{\mathrm{weak}} + \mathcal{H}_{\mathrm{sel}} \tag{5.13}$$

$$\mathcal{H}_{\mathrm{tot,B}} = \mathcal{H}_{\mathrm{weak}} + \mathcal{H}_{\mathrm{unsel}} \tag{5.14}$$

$$\mathcal{H}_{\mathrm{tot,C}} = \mathcal{H}_{\mathrm{iso}} + \mathcal{H}_{\mathrm{sel}} \tag{5.15}$$

$$\mathcal{H}_{\mathrm{tot,D}} = \mathcal{H}_{\mathrm{iso}} + \mathcal{H}_{\mathrm{unsel}}, \tag{5.16}$$

unter deren Verwendung TOP-Kurven für den Transfer (5.10) in I_n-Spinketten mit $n \in [2, 5]$ berechnet werden sollen.

5.2.2 Numerische Ergebnisse

Die Ergebnisse der Berechnungen sind in Abbildungen 5.7 ($n = 2$) bis 5.10 ($n = 5$) gezeigt, wobei die Unterabbildungen stets dieser Konvention folgen. Tafeln A und B zeigen die Ergebnisse der Optimierung unter Verwendung der schwachen Kopplung. Hierbei entspricht Tafel A dem Fall vollständig spinselektiver RF-Felder gemäß Gleichung (5.11), während für Tafel B der unselektive Kontroll-Hamiltonian (5.12) verwendet wurde. Analog hierzu zeigen Tafeln C bzw. D die Ergebnisse unter Annahme selektiver bzw. unselektiver Kontrollfelder, jedoch wird in diesen Fällen gemäß Gleichung (5.9) eine isotrope Kopplung aller nächsten Nachbarn angenommen.

In Tabelle 5.3 sind die durch die Optimierung gewonnenen charakteristischen Daten $\eta^\star$ und $t^\star$ bezüglich des Kettentransfers von -1-Quantenkohärenz gemäß Gleichung (5.10) zusammengefaßt. Da die Werte durch in einem Intervall $\tau \in [0, t_{\max}]$ numerisch optimierte Pulssequenzen bestimmt wurden, ist ihre Gültigkeit nur auf diesen Bereich beschränkt. Z.B. können für die in Tafeln 5.7.D bis 5.10.D gezeigten Kohärenztransfers durch isotropes Mischen [40] für $\tau \geq t^\star$ eine Transferamplitude $\eta \geq \eta^\star$ erreicht werden; die hier bestimmten TOP-Kurven entsprechen dem Ausschnitt der jeweiligen Transferfunktion innerhalb des Optimierungsintervalls.

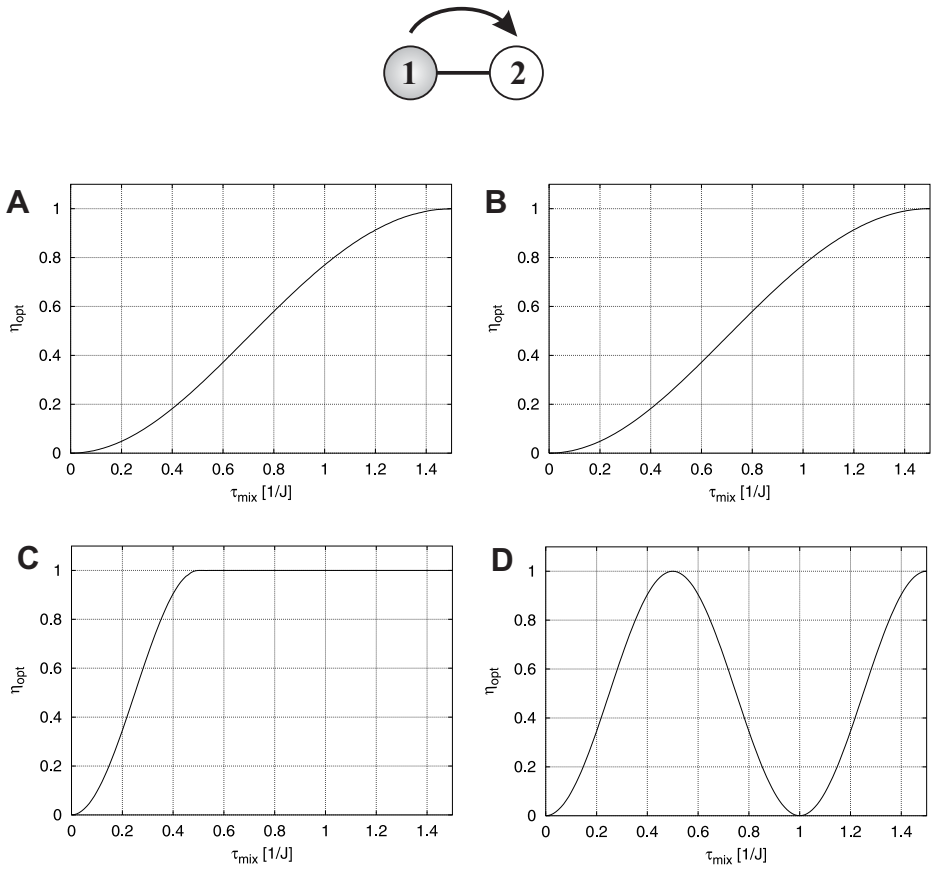

ABBILDUNG 5.7: Numerisch bestimmte Aufbaukurven für den Transfer von Inphase-Magnetisierung $S^- \rightarrow I^-$ in einem gekoppelten 2-Spinsystem. (A) Schwache Kopplung, selektive Pulse. (B) Schwache Kopplung, unselektive Pulse. (C) Isotrope Kopplung, selektive Pulse. (D) Isotrope Kopplung, unselektive Pulse.

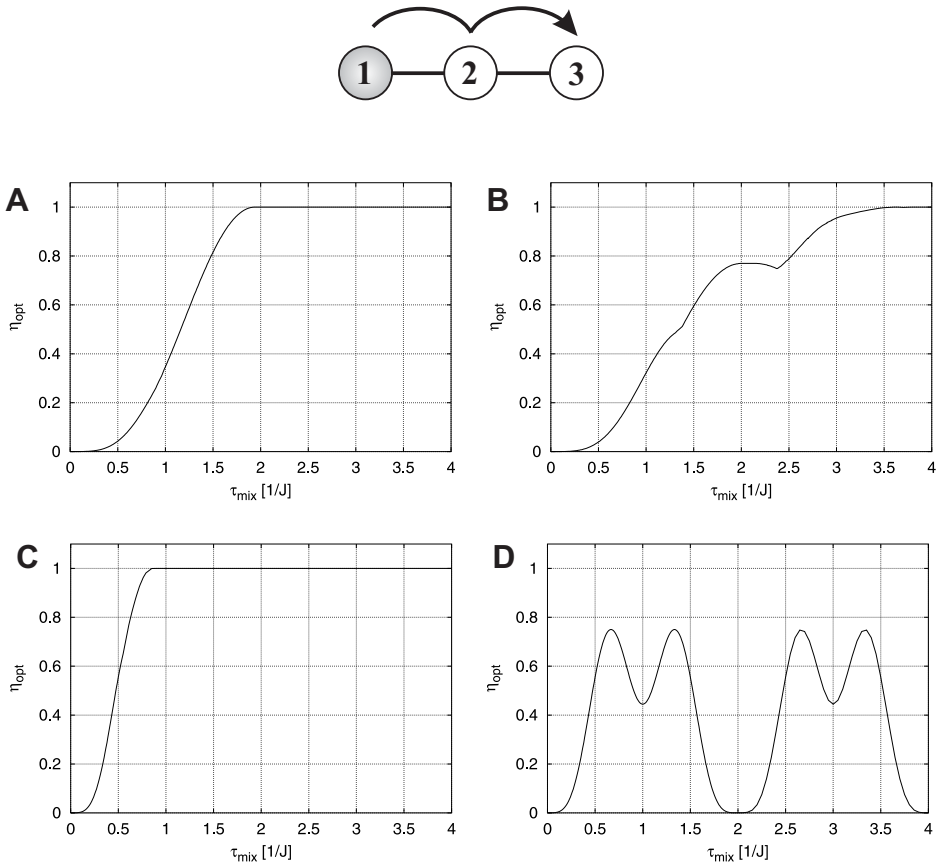

ABBILDUNG 5.8: Numerisch bestimmte Aufbaukurven für den Transfer von Inphase-Magnetisierung $I_1^- \rightarrow I_3^-$ in einem gekoppelten 3-Spinsystem. (A) Schwache Kopplung, selektive Pulse. (B) Schwache Kopplung, unselektive Pulse. (C) Isotrope Kopplung, selektive Pulse. (D) Isotrope Kopplung, unselektive Pulse.

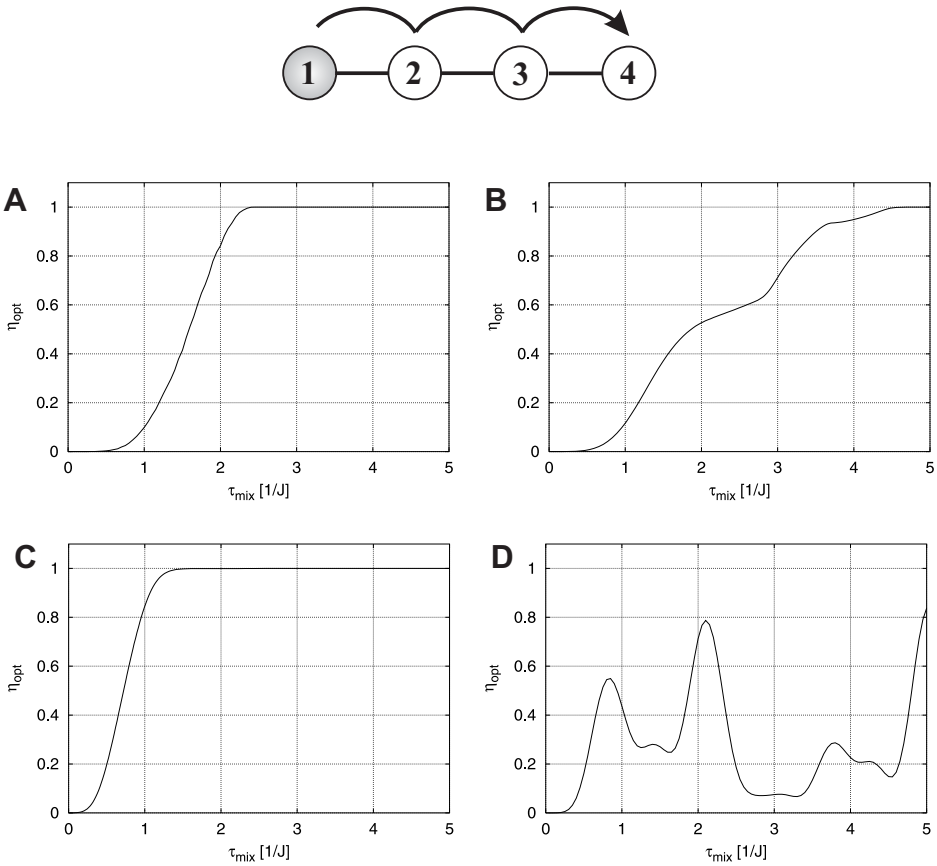

Abbildung 5.9: Numerisch bestimmte Aufbaukurven für den Transfer von Inphase-Magnetisierung $I_1^- \rightarrow I_4^-$ in einem gekoppelten 4-Spinsystem. (A) Schwache Kopplung, selektive Pulse. (B) Schwache Kopplung, unselektive Pulse. (C) Isotrope Kopplung, selektive Pulse. (D) Isotrope Kopplung, unselektive Pulse.

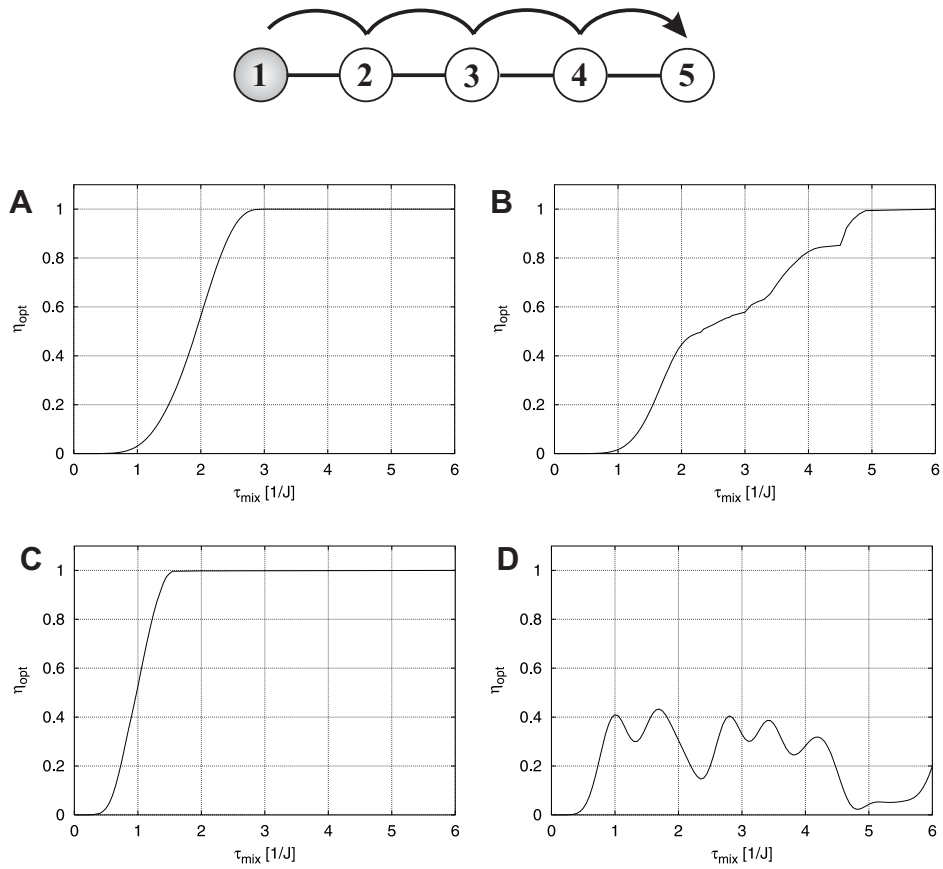

Abbildung 5.10: Numerisch bestimmte Aufbaukurven für den Transfer von Inphase-Magnetisierung $I_1^- \rightarrow I_5^-$ in einem gekoppelten 5-Spinsystem. (A) Schwache Kopplung, selektive Pulse. (B) Schwache Kopplung, unselektive Pulse. (C) Isotrope Kopplung, selektive Pulse. (D) Isotrope Kopplung, unselektive Pulse.

n	Kopplung	$\nu_{\mathrm{RF}}(I)$	$\eta^\star$ [-]	$t^\star$ $[1/J]$
2	schwach	selektiv	1	$1,5$
		unselektiv	1	$1,5$
	isotrop	selektiv	1	$0,5$
		unselektiv	1	$0,5$
3	schwach	selektiv	1	$1,9$
		unselektiv	1	$3,45$
	isotrop	selektiv	1	$0,8$
		unselektiv	$0,75$	$0,67$
4	schwach	selektiv	1	$2,35$
		unselektiv	1	$4,4$
	isotrop	selektiv	1	$1,2$
		unselektiv	$0,837$	5
5	schwach	selektiv	1	$2,7$
		unselektiv	1	$4,9$
	isotrop	selektiv	1	$1,55$
		unselektiv	$0,433$	$1,7$

TABELLE 5.3: Zusammenfasung der Daten bezüglich des Transfers von -1-Quantenkohärenz $I_1^- \rightarrow I_n^-$ in Ising-Spinkette I_n. Aufgeführt sind die numerisch bestimmten Werte für die maximale Transferamplitude $\eta^\star$ und die minimale Zeit $t^\star$, die eine Pulssequenz für deren Umsetzung benötigt.

5.3 Schlußfolgerung

Die in diesem Kapitel gezeigten Untersuchen zeigen, daß die numerische Optimierung von NMR-Experimenten unter Verwendung des in OCTANE implementierten steuerungstheoretischen Algorithmus möglich ist. Die korrekte Funktionsweise des Algorithmus wurde dadurch bestätigt, daß durch die Optimierung von Kohärenztranfers, deren unitäre Grenzen bereits durch theoretische Betrachtungen bekannt waren, im Rahmen der numerischen Genauigkeit identische Werte gefunden wurden.

Da die Optimierungen für jeden Zeitpunkt eine optimale Kontrolle (RF-Sequenz) erzeugen, kann man in einem gegebenen Zeitraum $\tau \in [0, t_{\max}]$ neben der maximalen Transfergrenze $\eta^\star$ auch die minimale Zeit $t^\star$ erhalten, die eine Pulssequenz benötigen wird, um den gewünschten Kohärenztransfer zu implementieren.

Kapitel 6

Optimierung unter Einschluß von Relaxations-Effekten

In diesem Kapitel werden Anwendungen von OCTANE zur Optimierung von Pulssequenzen unter dem Einfluß von Relaxation vorgestellt. Hierbei wird sich auf die Annahme von dipolar induzierter Relaxation beschränkt. Zunächst wird der einfachste Fall des Pseudo-1-Spin-Systems untersucht, das keine komplizierten Relaxationseigenschaften aufweist. Nachdem an diesem Fall die Prinzipien verdeutlicht wurden, wird die Optimierung eines in der mehrdimensionalen NMR-Spektroskopie häufig verwendeten Transferschritts vorgestellt.

6.1 Theorie

Wie in Kapitel 4 dargestellt, muß die Bewegung des Systems σ bzw. des konjugierten Systems λ durch Differentialgleichungen der Form $\dot{x} = f(x, u)$ zu beschreiben sein. Werden die Einflüsse von Relaxation vernachlässigt, so sind die entsprechenden Bewegungsgleichungen durch die Ausdrücke (4.5) und (4.29) gegeben. Um nun Relaxation während Optimierungen zu berücksichtigen, werden die entsprechenden Gleichungen des Optimierungsalgorithmus (vergl. Abschnitt 4.3.3) gemäß der in Kapitel 3 vorgestellten semiklassische Relaxationstheorie angepaßt.

Die Bewegung des Spinsystems wird durch die Liouville-von Neumann Gleichung

$$|\dot{\sigma}\rangle = -\imath\mathcal{L}_\sigma|\sigma\rangle \tag{6.1}$$

beschrieben. Durch den Übergang in den Liouville-Raum wird das Spinsystem durch einen Superket repräsentiert [41]. Hierbei wird die Gesamt-Dynamik des Systems durch den Superoperator

$$\mathcal{L}_\sigma = -\imath\mathcal{H}_{\text{tot}} - \Gamma \tag{6.2}$$

bestimmt. Dieser als Liouvillian bezeichnete Operator setzt sich aus dem Super-Hamiltonian $\mathcal{H}_{\text{tot}}$ (Beschreibung der Spindynamik) und der Redfield-Matrix Γ (Beschreibung der Relaxation) zusammen.

Die Bewegungsgleichung der Lagrange-Faktoren ergibt sich durch Auswerten des Ausdrucks (4.25) unter Beachtung von Beziehung (4.28) zu

$$|\dot{\lambda}\rangle = -\imath\mathcal{L}_\lambda|\lambda\rangle. \tag{6.3}$$

Wie bereits in Kapitel 4 ergibt sich im Liouville-Raum ebenfalls eine Identität der Bewegungsgleichgungen des Spinsystems und der Lagrange-Faktoren. Lediglich der Liouvillian

$$\mathcal{L}_\lambda = -\imath\mathcal{H}_{\text{tot}} + \Gamma \tag{6.4}$$

weist einen Unterschied auf. Wie dem Algorithmus 4.3.3 zu entnehmen ist, werden die Lagange-Faktoren ausgehend vom Endzeitpunkt t_1 zeitlich zurückentwickelt, d.h. die Entwicklungsperioden $\Delta = t_0 - t_1$ sind wegen $t_1 > t_0$ negativ. Hierdurch wird die Wirkung des Operators während der Propagation zeitlich umgekehrt. Da sich die Lagrange-Faktoren ausgehend von einem Startzustand $\lambda(t_1)$ zu einem Endzustand $\lambda(t_0)$ entwickeln, wirkt hierbei auch die Relaxation dämpfend entlang der inversen Zeitachse. Dies kommt dadurch zum Ausdruck, daß das Vorzeichen der Redfield-Matrix Γ in Gleichung (6.4) gegenüber Geichung (6.2) umgekehrt ist.

6.2 Das Pseudo-1-Spinsystem

Der einfachste Fall eines Spinsystems ist das 1-Spinsystem. Dieses verfügt jedoch nicht über die dipolaren Wechselwirkungen, die für den entsprechenden Relaxationsmechanismus notwendig sind. Daher muß eine Beschreibung gewählt werden, die zum einen die Spindynamik des 1-Spinsystems und zum anderen über die dipolaren Relaxationseigenschaften eines höheren Spinsystems verfügt. Eine mögliche Darstellung ist die eines nicht-gekoppeltes 2-Spinsystems. Wegen der fehlenden Kopplung verhalten sich die beiden Kerne

wie isolierte Teilchen, d.h. die Charakterstik des 1-Spinsystems bleibt gewahrt. Durch den zweiten Spin wird jedoch das benötigte Dipolmoment eingeführt, wodurch der gewünschte Relaxationsmechanismus ermöglicht wird. Die Beschränkung auf zwei Kernspins führt darüber hinaus dazu, daß das System über nur eine Dipol-Dipol-Kopplung verfügt, wodurch ausschließlich autokorrelierte Relaxation möglich ist. Dieses Modell legt die Bezeichnung des Pseudo-1-Spinsystems nahe. Hiermit kann nun das zeitabhängige Verhalten der Terme eines Spin-1/2-Teilchens unter dem Einfluß dipolarer Relaxation untersucht werden.

6.2.1 Die theoretische Erwartung

Dieses einfache System wurde wegen seiner einfache Beschreibbarkeit ausgewählt, um erste Optimierungs-Versuche relaxierender Systeme durchzuführen. Die Relaxation läßt sich vollständig durch zwei Relaxationsraten beschreiben, was in den bekannten Bloch-Gleichungen [13]

$$\begin{aligned} I_z(t) &= I_{z,0} - (I_{z,0} - I_z(0)) \exp(-t/T_1), \\ I_{\{x,y\}}(t) &= I_{\{x,y\}}(0) \exp(-t/T_2). \end{aligned} \tag{6.5}$$

resultiert. Hierbei ist $1/T_1$ die Relaxtionsrate der longitudinalen Komponente (Spin-Gitter-Relaxation) und $1/T_2$ entspricht der Rate der transversalen Relaxation (Spin-Spin-Relaxation).

In OCTANE wird Relaxation im Rahmen des in Kapitel 3 beschriebenen Wagnes-Bloch-Redfield-Modells beschrieben. Durch Verwendung der Methoden `setRedfield` bzw. `readRedfield` wird die zur Erzeugung des Liouville-Operators benötigte Redfield-Matrix an die Optimierungs-Bibliothek übergeben. Im Falle des Pseudo-1-Spinsystem entspricht die Redfield-Matrix der eines 2-Spinsystems, die durch Gleichung (3.12) unter Verwendung der dipolaren Spintensoren gemäß Tabelle 3.2 und einer geeigneten spektralen Dichtefunktion beschrieben wird.

Da im Pseudo-1-Spinsystem die Relaxationseigenschaften bekannt sind und sich alle Spinterme durch die entsprechenden Spinterme I_x, I_y und I_z darstellen lassen, eignet sich das System sehr gut dazu, die Fähigkeit der numerischen Bibliothek zu demonstrieren, unter dem Einfluß von Relaxation optimierte Lösungen zu bestimmen.

Durch die möglichen Spinterme sind zwar keine komplizierten Transfers als Optimierungsziel möglich. Jedoch weisen die vorhandenen Terme unterschiedliche Relaxationseigenschaften auf, die durch eine Pulssequenz ausgenutzt werden können. Betrachtet man nun z.B. den Fall

$$I_x \xrightarrow{\tau} I_x \tag{6.6}$$

als Optimierungsziel, so kann man aufgrund der größeren transversalen Relaxationsrate eine Pulssequenz als Ergebnis erwarten, die möglichst schnell die Magnetisierung entlang der longitudinalen Achse speichert, um sie erst bei der Zeit τ wieder in transversale Magnetisierung zu transformieren. Dies entspricht einem Transferschema

$$I_x \xrightarrow{(\frac{\pi}{2})_{\pm y}} [I_z] \xRightarrow{\tau} [I_z] \xrightarrow{(\frac{\pi}{2})_{\mp y}} I_x, \tag{6.7}$$

wobei die langsam relaxierenden Terme mit "[ ]" gekennzeichnet sind. Durch dieses Transferschema werden die Transfereffizienzen der transversalen und longitudinalen Komponenten vereinheitlicht, d.h.

$$\langle I_x(\tau)|I_x\rangle \equiv \langle I_z(\tau)|I_z\rangle. \tag{6.8}$$

6.2.2 Das numerische Ergebnis

Für die numerische Optimierung wird eine dipolare 2-Spin-Redfield-Matrix (s. Teil IV, Kapitel 13.1) mit einer Korreltionszeit $\tau_c = 5$ ns verwendet. Die Entwicklung der transversalen und longitudinalen Komponenten unter diesem Relaxationsoperator sind in Abbildung 6.1 zusammengefaßt. Sehr deutlich erkennt man die unterschiedlichen Relaxationsraten. Im betrachteten Zeitraum kling die transversale Komponente annähernd vollständig ab, während die longitudinale noch einen endlichen Erwartungswert aufweist. Um eine deutliche Diskriminierung des transversalen Zustandes zu erreichen, wird die Zeit der Optimierung $\tau = 0,25$ s gewählt. An diesem Punkt würde das Transferschema (6.6) praktisch keine Magnetisierung erwarten lassen. Wird jedoch das Schema (6.7) verwendet, so ist nach

$$\langle I_x(\tau = 0,25\text{s})|I_x\rangle = \langle I_z(\tau = 0,25\text{s})|I_z\rangle = 0,65 \tag{6.9}$$

noch ein hoher Anteil transversaler Magnetisierung zu erwarten. Für die numerische Optimierung wurde eine willkürliche RF-Pulsform mit einer maximalen Feldstärke von 1 Hz erzeugt, die in 5000 Punkte diskretisiert vorlag,

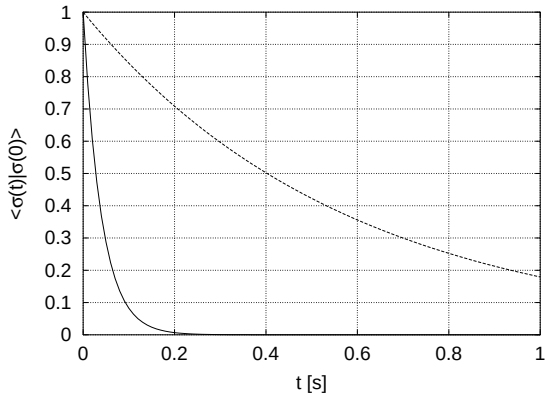

ABBILDUNG 6.1: Zeitabhängige Entwicklung der transversalen (durchgezogen) und longitudinalen (gestrichelt) Komponente eines Pseudo-1-Spinsystems unter dem Einfluß dipolar induzierter Relaxation. Die Daten entstammen einer Simulation gemäß des Wagness-Bloch-Redfield-Modells unter Verwendung der spektralen Dichtefunktion eines starren Rotors mit einer Korrelationszeit $\tau_c = 5$ ns. Man erkennt die deutlich unterschiedlichen Relaxationsraten für transversale und longiudinale Spinanteile.

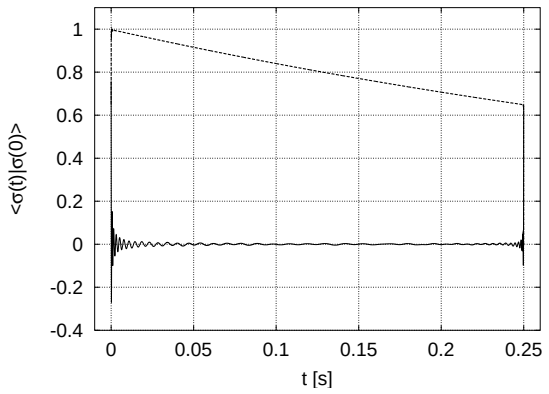

ABBILDUNG 6.2: Durch die optimierte Pulssequenz erzeugten 1-Spin-Basisoperatoren I_x (durchgezogen) und I_z (gestrichelt). Durch die Sequenz wird die schnell relaxierende transversale ohne Zeitverlust in die langsam relaxierende longitudinale Komponente umgewandelt. Am Ende erfolgt dazu analog die Rücktransformation in den Zielterm. Somit ist ausschlieslich die longitudinale Relaxationsrate die während der gesamten Pulsseuenz wirkwirksam, d.h. der Erwartungswert des Zieloperators stimmt mit dem des longitudinalen Operators überein.

d.h. der Abstand der RF-Werte beträgt 50 μs. Die Pulsform verfügt über keine Beschränkung der Phase, gemäß

$$\mathcal{H}_{\mathrm{RF}} = 2\pi \left(\nu_x(t) I_x + \nu_y(t) I_y\right) \tag{6.10}$$

werden dafür zwei orthogonale RF-Anteile verwendet. Neben der Spin-Relaxation wurden keine weiteren Störeffekte berücksichtigt. Die Optimierung wurde unter Verwendung der Methode konjugierter Gradienten (`optimizeCG`) durchgeführt, wobei die Konvergenzschranke mit $\varepsilon = 10^{-6}$ festgelegt wurde.

In Abbildung 6.2 sind die durch die optimierte Pulssequenz erzeugten Operatoren aufgetragen. Hierbei werden die gemachten Voraussagen vollständig erfüllt. Das Transferschema (6.7) wird durch die numerische Optimierung in einer Pulssequenz implementiert, so daß der Erwartungswert des Zieloperators mit $\langle I_x(\tau)\rangle = 0,6478$ den in Gleichung (6.9) vorhergesagten Wert sehr gut annähert.

Durch dieses Ergebnis wird demonstriert, daß die Optimierung von Pulssequenzen für relaxierende Spinsysteme prinzipiell möglich ist. Im Folgenden gilt es, diese Art von Optimierung auf komplexere Fälle anzuwenden.

6.3 Das 2-Spinsystem

Die bisher besprochenen Ergebnisse bezüglich relaxationsoptimierter NMR-Experimente in 1-Spinsystemen sind natürlich nicht von realem experimentellem Interesse. Sie machen jedoch deutlich, daß die numerische steuerungstheoretische Optimierung unter Einschluß von Relaxation möglich ist und relaxtionsstabilisierte Transferschemata identifiziert werden können.

Im Folgenden soll ein Fall untersucht werden, der einen direkten Bezug zu praktisch relevanten Kohärenztransfers aufweist. In vielen bei der Strukturaufklärung Anwendung findenden zweidimensionalen NMR-Experimenten ist ein Transferschritt zwischen zwei gekoppelten Kernspins ein wichtiges Element, der konventionell z.B. als INEPT-Schritt implementiert wird. Durch die Größe der hierbei verwendeten Moleküle ist die molekulare Rotationsgeschwindigkeit jedoch so langsam, daß die entstehenden fluktuierenden dipolaren Felder Relaxation effektiv ermöglichen. Daher ist es von großem Interesse, einen solchen Transferschritt unter Einschluß von Relaxation mittels der Steuerungs-Theorie zu untersuchen.

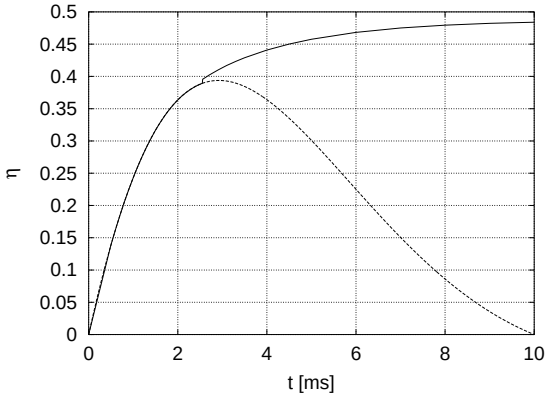

ABBILDUNG 6.3: Zeitabhängigkeit der Transferamplituden für $I_x \rightarrow 2I_yS_z$ unter der Wirkung eines diploaren Relaxationsoperators für INEPT (gestrichelt) und ROPE (durchgezogen). Die Kurven sind für ein mit $J = 100$ Hz gekoppeltes 2-Spinsystem berechnet. Die hierbei verwendete Redfield-Matrix wurde mit der spektralen Dichte eines starren Rotors mit einer Korrelationszeit $\tau_c = 50$ ns bestimmt.

6.3.1 Die numerische Untersuchung

Es wird ein gemäß

$$\mathcal{H}_c = 2\pi J I_z S_z \tag{6.11}$$

schwach gekoppeltes 2-Spinsystem als Basis für die Optimierung herangezogen. Zu untersuchen ist der Transferschritt

$$I_\alpha \xrightarrow{\tau} 2I_\beta S_\gamma, \tag{6.12}$$

wobei $\alpha, \beta, \gamma \in \{x, y, z\}$. Konventionell wird dieser Schritt durch INEPT-Elemente verwirklicht, wobei das Transferschema

$$I_\alpha \xrightarrow{\mathrm{RF}} I_x \xrightarrow{J} 2I_yS_z \xrightarrow{\mathrm{RF}} 2I_\beta S_\gamma \tag{6.13}$$

erfüllt wird. Da in diesem Schema ausschließlich schnell relaxierende Spinterme beteiligt sind, ist die Effizienz des Transfers durch das Verhältnis aus Relaxation und Kopplung bestimmt. So gilt für gleiche Relaxationseigenschaften, daß der Transfer durch eine größere Kopplung effizienter wird, weil die Mischzeit geringer wird.

In einem gegebenen Spinsystem kann man den zeitabhängigen Verlauf der Transferamplitude unter dem Einfluß einer Relaxationsmatrix für das

Transferschema (6.13) simulieren. In Abbildung 6.3 ist diese (gestrichelt) für ein mit $J = 100$ Hz gekoppeltes 2-Spinsystem gezeigt. Man erkennt sehr gut die gegensätzlichen Effekte des Aufbaus durch den Transfer und den Abbau durch Relaxation. Für kurze Zeiten ist der Aufbau effizienter als der Abbau, jedoch für längere Mischzeit kehrt sich dieses Verhältnis um, bis sogar durch das Überwiegen der Relaxation der maximale Transfer nicht mehr erreicht werden kann. Im Falle des INEPT-Transfers (6.13) wäre im Idealfall (keine Relaxation) der Transfer nach $t = 1/2J$ vollständig zu erreichen, d.h. $\eta(1/2J) = 1$. Im verwendeten Spinsystem entspricht dies einer Zeit von $t = 5$ ms. Wie man der Abbildung entnehmen kann, ist dies unter dem Einfluß von Relaxation nicht mehr die optimale Lösung. Das Maximum der INEPT-Transferamplitude hat sich durch die zusätzliche Dynamik zu kürzeren Zeiten verschoben und stellt nur noch einen Bruchteil des maximal möglichen Transfers dar. Der beste mögliche Transfer $\eta(t_{\text{opt}}^{\text{INEPT}}) = 0,3938$ wird nun für $t_{\text{opt}}^{\text{INEPT}} = 2,9$ ms erreicht.

Unter Verwendung desselben Spinsystems sollen nun steuerungs-theoretisch optimierte Pulssequenzen unter Verwendung der Bibliothek OCTANE numerisch bestimmt werden. Hierbei sollen die beiden Kerne selektiv bepulsbar sein, d.h. der RF-Hamiltonian hat die Form

$$\mathcal{H}_{\text{RF}} = 2\pi \left(\nu_{x,1}(t) I_x + \nu_{y,1}(t) I_y + \nu_{x,2}(t) S_x + \nu_{y,2}(t) S_y\right). \tag{6.14}$$

Unter dieser Voraussetzung sind Rotationen der einzelnen Spinterme durch harte selektive Pulse in vernachlässigbarer Zeit und somit frei von Relaxations-Einflüssen möglich. Da somit die Transformationen $I_\alpha \rightarrow I_x$, $I_y \rightarrow I_\beta$ und $S_z \rightarrow S_\gamma$ vor bzw. im Anschluß an die Mischsequenz ohne Zeitbedarf (d.h. auch ohne Relaxations-Verluste) darstellbar sind, läßt sich das Optimierungsziel (6.12) als

$$\{\mathcal{A} = I_y\} \xrightarrow{\tau} \{\mathcal{F} = 2 I_y S_z\} \tag{6.15}$$

formulieren. Neben der Relaxation werden in der Optimierung keine weiteren Störeffekte berücksichtigt. Die Optimierung wird mit der Methode der konjugierten Gradienten (`optimizeCG`) durchgeführt, wobei eine Konvergenzschranke $\varepsilon = 10^{-6}$ verwendet wird.

In Abbildung 6.3 sind zunächst die Transferamplituden dieser ROPE-Sequenzen (Relaxation Optimized Pulse Elements) aufgetragen (durchgezogene Kurve) und dem konventionellen Ansatz (INEPT, gestrichelte Kurve)

gegenübergestellt. In dem Bereich $t = [0, t_1],\ t_1 < t_{\text{opt}}^{\text{INEPT}}$ ist der Verlauf der Transferamplituden für den konventionellen und optimierten Ansatz identisch, d.h. INEPT stellt dort die optimale Lösung dar[1]. Für Zeiten $t > t_1$ erweist sich die Relaxation für einen reinen INEPT-Transfer als zu effizient, was sich vor allem darin manifestiert, daß die Dämpfung stärker ist als der Transfer. Die ROPE-Sequenzen zeigen in diesem Bereich bemerkenswerter Weise eine asymptotische Annäherung an eine maximale Transferamplitude, die deutlich über dem Maximum des INEPT-Transfers liegt.

Die Analyse der Pulssequenz gibt Aufschluß über den Grund dieses besseren Transferverhaltens. Zerlegt man die sich unter der Pulssequenz entwickelnde Dichtematrix des Spinsystems in die zeitlichen Anteile der Basisoperatoren (vergl. Abbildung 6.4), so erkennt man zunächst, daß die Sequenz in zwei prinzipiell symmetrische Anteile aufgeteilt ist. Die Grenze der Teilung liegt hierbei immer bei $t = \tau/2$.

Im Bereich $t = [0, \tau/2]$ ist das prinzipielle Vorgehen, den relaxationsanfälligen transversalen Startoperator durch einen harten Puls, d.h. in möglichst kurzer Zeit, in den gegenüber Relaxation unempfindlichen longitudinalen zu transformieren. Daraufhin wird dieser Zustand mit einer bestimmten Geschwindigkeit zurück in die transversale Ebene rotiert. Die Rotationsgeschwindigkeit ist hierbei nicht konstant; sie bestimmt sich durch das optimale Verhältnis zwischen Kopplungstransfer und Relaxation. Diese erste Phase ist beendet, wenn die longitudinale Magnetiserung vollständig zurückrotiert wurde. An diesem Punkt hat die transversale Magnetisierung I_x ein Maximum erreicht, ebenso wie der Zielterm $2I_yS_z$.

Im folgenden Bereich $t = [\tau/2, \tau]$ wird ausgehend von diesen lokalen Maxima wiederum ein mit dem Relaxationsoperator (in erster Näherung) kommutierender Spinoperator durch eine zu der ersten Phase analogen Rotation erzeugt. Hierbei entwickelt sich unter der Kopplung die verbliebene transversale Magnetisierung in den schnell relaxierenden bilinearen Term $2I_yS_z$,

[1] In Abbildung 6.3 erkennt man bei $\eta(t^\star)$ eine scheinbare Unstetigkeitsstelle. Diese stellt ein Artefakt des numerischen Algotithmus dar. Die analytische Lösung [42] zeigt, daß es einen definierten Zeitpunkt gibt, an dem der Wechsel von INEPT zu ROPE so erfolgt, daß die Kurve der Transferamplitude eine stetige Funktion bildet. Der numerische Algorithmus kann diesen Punkt nur im Rahmen einer gewissen Schwankungsbreite identifizieren, die durch die Konvergenzschranke ε bestimmt wird. Da der Zeitbedarf für eine Optimierung invers proportional zu ε steigt, definiert man in der Regel einen endlichen Wert $\varepsilon \gg 0$. Im speziellen Fall betrug $\varepsilon = 10^{-6}$.

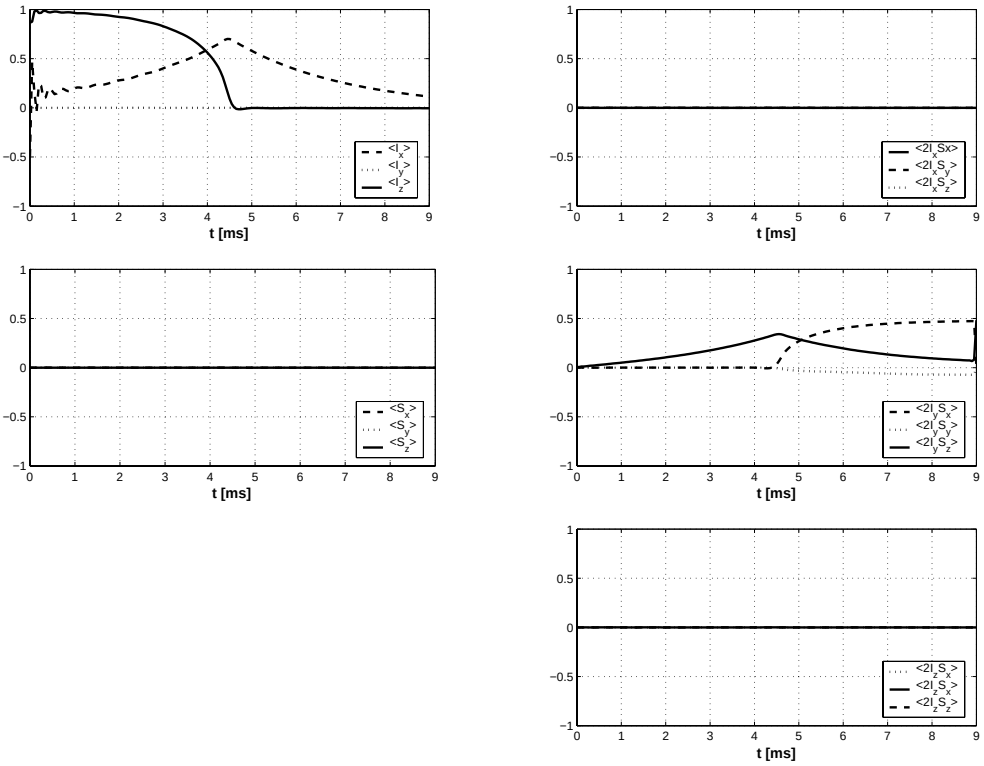

ABBILDUNG 6.4: Zeitliche Entwicklung des Spinsystems unter der optimierten Pulssequenz. Gezeigt ist der Anteil der kartesischen Basisoperatoren an der Dichtematrix des Spinsystems zu jedem Digitalisierungszeitpunkt am Beispiel der ROPE-Sequenz für $\tau = 9$ ms. Charakteristisch ist die Aufteilung in zwei symmetrische Teilsequenzen für $t = [0, \tau/2]$ bzw. $t = [\tau/2, \tau]$. Zu Beginn der Sequenz wird der für Relaxation anfällige Startoperator I_x sofort in relaxationsstabile longitudinale Magnetisierung I_z verwandelt. Diese wird langsam in die transversale Ebene zurückrotiert. Durch die Rotationsgeschwindigkeit wird ein für den Transfer optimales Verhältnis zwischen Kopplungsentwicklung und Relaxation erreicht. Diese Rotation ist vollständig am Ende der ersten Phase. Der dazu symmetrische Teil beruht auf den selben Prinzipien. Hier findet die Rotation ausgehend vom schnell relaxierenden Operator $2I_yS_z$ zu dem mit dem Relaxationsoperator (annähernd) kommutierenden Operator $2I_yS_y$ statt. Aus diesem relaxationsstabilen Term wird am Ende durch einen harten Puls der Zielzustand generiert.

der gleichzeitig durch die Rotation in einen langsam relaxierenden bilinearen Term $2I_yS_y$ verwandelt wird[2]. Am Ende der Phase wird letzterer wieder durch einen harten Puls in den Zieloperator zurückverwandelt.

Diese Erkenntnisse lassen sich in dem Transferschema

$$I_\alpha \xrightarrow{\text{RF}} [I_z] \Longrightarrow I_x \xrightarrow{J} 2I_yS_z \Longrightarrow [2I_\delta S_\varepsilon] \xrightarrow{RF} 2I_\beta S_\gamma \tag{6.16}$$

zusammenfassen. Hierbei werden relaxations-stabile Reservoirs durch "[ ]" dargestellt, die genannten langsamen Rotationen werden durch "$\Longrightarrow$" symbolisiert. Als Nebenbedingung für die als Reservoir dienenden Operatoren gilt, daß sie mit dem Relaxationsoperator Γ kommutieren[3].

In [42] konnten diese Erkenntnisse durch entsprechende analytische Lösungen bestätigt werden, insbesondere konnten analytische Beziehungen für die gegebenen Rotationen angegeben werden. In Kapitel 14 wird ein Verfahren vorgestellt, mit dem die analytische Lösung in eine Hartpuls-Näherung umgesetzt werden kann. Erste diese Näherung verwendende Experimente zeigen, daß sich diese Transferschemata auch in realen Spinsystemen implementieren lassen.

6.4 Schlußfolgerung

Es konnte in diesem Kapitel gezeigt werden, daß mit OCTANE Optimierungen unter dem Einschluß von Relaxationseffekten möglich sind. Durch die Analyse der Optimierungsergebnisse können Transferschemata entwickelt wer-

[2]Voraussetzung für die Stabilität eines bilinearen Operator gegenüber Relaxation ist, daß der entsprechende Term mit dem Relaxationsoperator kommutiert. Durch die numerische Optimierung wurde der Term $2I_yS_y$ als Reservoir gewählt, diese Wahl ist jedoch nicht die einzig mögliche. Ausgehend vom Term $2I_yS_z$ können durch selektive Rotationen vier Terme erzeugt werden: (1) Rotation um S_x: $2I_yS_y$, (2) Rotation um S_y: $2I_yS_x$, (3) Rotation um I_x: $2I_zS_z$, (4) Rotation um I_y: $2I_yS_z$. Hiebei kommutieren die Terme (1)-(3) mit dem Relaxationsoperator und stellen somit geeignete Zielterme dar.

[3]Betrachtet man den Grenzfall langsamer molekularer Rotation (*spin diffusion limit*, $\omega_I\tau_c \gg \omega_S\tau_c \gg 1$), so gilt für die spektralen Dichtefunktionen $J(\omega = 0) \gg J(\omega_S) \gg J(\omega_I) \approx J(\omega_I \pm \omega_S)$. D.h. der Relaxationoperator wird durch die spektrale Dichte $J(\omega = 0)$ bestimmt. In erster Näherung kann man den Beitrag aller weiteren Terme vernachlässigen und erhält als Relaxationsmatrix für diesen Grenzfall

$$\Gamma = \frac{2}{15}c_0^2\tau_c[2I_zS_z, [2I_zS_z,]],$$

wobei c_0 durch Gleichung (3.24) gegeben ist. Mit diesem einfachen Relaxationsoperator lassen sich die benötigten Kommutatoren relativ einfach bestimmen.

den, die gegenüber konventionellen Experimenten verbesserte Relaxationseigenschaften aufweisen. Durch die Verwendung des Wagness-Bloch-Redfield-Formalismus ist es möglich, die Relaxation beliebiger Moleküle zu modellieren. Dabei ist man nicht auf die in diesem Kapitel verwendete Relaxation aufgrund diplolarer Wechselwirkungen beschränkt.

Kapitel 7

Optimierung unter Einschluß von Offset-Effekten

Verwendet man ideale Bedingungen zur Optimierung von Pulssequenzen, so kann man die quantenmechanischen Grenzen für bestimmte Kohärenztransfers erforschen (vergl. Kapitel 5). So gewonnene Ergebnisse sind zwar theoretisch interessant, führen jedoch nicht zu in der Praxis direkt verwertbaren Pulssequenzen.

Um den erfolgreichen Einsatz der Optimierungsbibliotkek OCTANE zur Erzeugung einer praxisrelevanten Puls-Form zu verdeutlichen, werden in diesem Kapitel die Ergebnisse eines Projektes vorgestellt, dessen Ziel die Entwicklung eines breitbandigen und fehlertoleranten Anregungspulses war.

7.1 Die numerische Optimierung

Um in der Praxis allgemein einsetzbare RF-Formen zu erhalten, ist es notwendig, die real herrschenden Bedingungen durch ein entsprechendes Modell bereits während der Optimierung abzubilden. Die hierbei wichtigsten Einflüsse sind Resonanzfrequenz-Offsets und Radiofrequenz-Offsets (B_1-Feld-Inhomogenitäten). OCTANE stellt die zu dieser Modellierung benötigten Methoden bereit (`addOffset` und `setB1errGauss`).

Da die zu optimierende RF-Form einen großen Resonanzfrequenz-Offsetbereich abdecken soll (z.B. ± 20 kHz), besteht die Möglichkeit, daß der Optimierungsalgorithmus versucht, diesen Bereich durch entsprechende RF-Leistungen (z.B. 40 kHz) abzudecken. Nun können durch die RF-Bauelemte eines Spektrometers keine beliebig hohen Feldstärken erzeugt werden. Da-

her muß die erlaubte Leistung der RF-Felder bereits während der Optimierung auf ein durch die verwendete Spektrometer-Hardware begrenztes Maximum beschränkt werden. Von OCTANE wird hierzu die Methode `limitRF` zur Verfügung gestellt.

Das Optimierungsziel eines Anregungspulses läßt sich vollständig durch ein 1-Spinsystem darstellen. Somit können die in der Klasse `OCT_rho` für allgemeine N-Spinsysteme implementierten Algorithmen an diesen Spezialfall angepaßt werden. Dazu wurde die spezialisierte Klasse `OCT_rho_1sp` abgeleitet. Eine Verringerung der Laufzeit um den Faktor 7 ließ sich z.B. durch Quaternion-Propagation [43] der Dichtematrix erreichen. Weitere Änderungen in den algortihmischen Strukturen werden im Folgenden erläutert.

7.1.1 Die Kostenfunktion

Wie bereits in der Einleitung gesagt, ist das Optimierungsziel ein Anregeungspuls. Quantenmechanisch läßt sich dies als

$$[\mathcal{A} = I_z] \xrightarrow{T} [\mathcal{F} = I_x] \tag{7.1}$$

formulieren. Hierbei sei T die für die Einstrahlung der RF-Form gewünschte Zeit. Verwendet man die standardmäßig in OCTANE implementierte Minimerung der Kostenfunktion

$$\min\left(\Phi = 1 - \langle\sigma(T)|\mathcal{F}\rangle\right), \tag{7.2}$$

so wird ausschließlich gewährleistet, daß die Projektion des durch die Optimierung erreichten Zustandes auf die Zielachse maximal ist. Im Idealfall entspricht der erreichte Zustand vollständig dem Zielzustand, d.h. $\Phi = 0$. Werden jedoch Störeffekte in die Optimierung eingeschlossen, wird in der Regel ein Minimum der Kostenfunktion $\Phi = (\varepsilon > 0)$ durch die Optimierung erreicht. Für das durch Gleichung (7.1) gegebenen Optimierungsziel bedeutet dies, daß die erreichte Magnetisierung innerhalb eines Konus verteilt sein wird, dessen Hauptachse identisch zur Zielachse ist (vergl. Abbildung 7.1.A). Da es jedoch das Ziel der Optimierung ist, möglichst viel Magnetisierung mit einheitlicher Phase in die transversale Ebene zu überführen, sind alle Anteile am erreichten Zustand, die noch eine Projektion $\langle\sigma(T)|I_y\rangle > 0$ aufweisen, diesem entgegengesetzt. Daher ist es notwendig, die Kostenfunktion

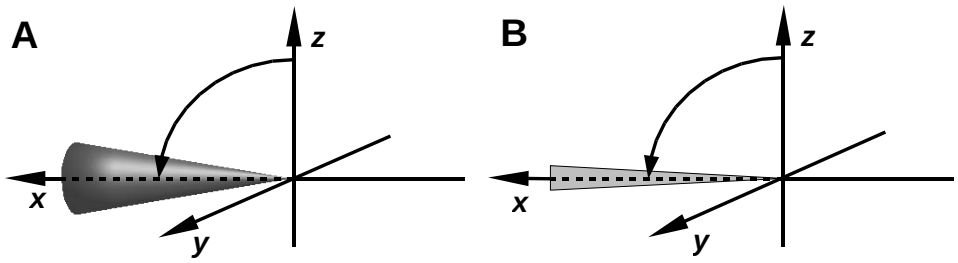

ABBILDUNG 7.1: Qualitative Darstellung des Einflusses der Kostenfunktion auf das Optimierungsziel. (A) Verwendet man die Minimierung gemäß Gleichung (7.2) als Maß der Konvergenz, so wird der Algorithmus nur die Projektion der Magnetisierung auf die Zielachse maximieren. Da keine Raumrichtungen unterschieden werden, ist das Ergebnis ein Konus mit minimierter Basisfläche, dessen Hauptachse mit der Zielachse übereinstimmt. (B) Durch Einführung von räumlichen Gewichtungsfaktoren $\alpha_{\{x,y,z\}}$ und der Formulierung des Konvergenzkriteriums als Minimierung gemäß Gleichung (7.3) können bestimmte räumliche Anteile diskriminiert werden. Im Falle des Anregungspulses soll die durch die Anregung erreichte Phasenlage der Magnetisierung möglichst maximal entlang der Zielachse (z.B. der x-Achse) liegen. Das Ergebnis ist im Extremfall die Schnittfläche eines Konus (z.B. die Schnittfläche des Konus mit der xz-Ebene).

so zu formulieren, daß diese Komponenten möglichst vollständig unterdrückt werden (vergl. Abbildung 7.1.B). Durch numerische Auswertung von

$$\min\left(\Phi = \sum_{\nu=\{x,y,z\}} \alpha_\nu \langle (\sigma(T) - \mathcal{F})^\dagger \, |I_\nu\rangle^2\right) \tag{7.3}$$

kann dieser Fall näherungsweise erreicht werden. Das generelle Optimierungsziel wird weiterhin erreicht, weil die in Gleichung (7.2) zu maximierende Projektion durch die zu minimierende Abweichung des optimierten Zustandes vom Zielzustand erreicht wird. Durch die Einführung der durch $\alpha_{\{x,y,z\}}$ parametrisierten Summe über die Projektion des optimierten Zustandes auf die orthogonalen Basiszustände kann man darüberhinaus die einzelnen Beiträge unterschiedlich gewichten. Da die Kostenfunktion (7.3) minimiert wird, besteht die Möglichkeit den Anteil mit der Phase y stärker zu gewichten und diesen so aus dem optimierten Zustand annähernd zu entfernen.

7.1.2 Die Parametrisierung der Optimierung

Im konkreten Fall soll der einem Anregungspuls entsprechende Transfer gemäß Gleichung (7.1) numerisch optimiert werden. Die durch die optimierte RF-

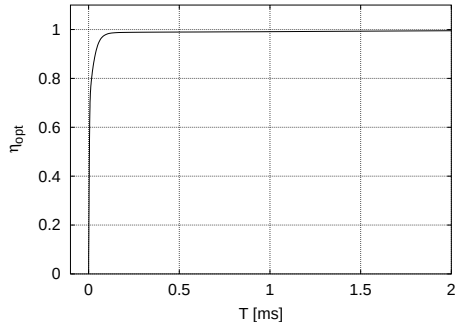

ABBILDUNG 7.2: TOP-Kurve für Anregungspulse, berechnet für einen Offsetbereich $\nu_{\text{off}} \in [-20\text{kHz}, 20\text{kHz}]$ und relative Fehler im erzeugten B_1-Feld $\text{Err}_{\text{rel}} = \pm 5\%$.

Form erreichte Anregung soll einen möglichst kleiner Phasenfehler aufweisen. Daher werden die räumlichen Gewichtungsfaktoren der Kostenfunktion (7.3) so gewählt, daß Anteile mit y-Phase diskreminiert werden. Da es sich bei dem zugrunde liegenden Spinsystem um ein 1-Spinsystem handelt, verfügt das System über keinen Kopplungs-Hamiltonian, d.h. der Fluß der Systems (*Drift*) wird einzig durch die Offset-Hamiltonians bestimmt. Die zu optimierende RF-Form wird durch einen Operator

$$\mathcal{H}_{\text{RF}} = 2\pi \left(\nu_x(t) I_x + \nu_y(t) I_y\right) \tag{7.4}$$

dargestellt, wodurch die effektive Phase des Pulses in $\varphi \in [0, 2\pi]$ liegen kann.

Die Gesamt-Dauer des Pulses wurde $T = 2$ ms gewählt. In dieser Zeit ist es möglich, die für diese Parameter maximale Anregung zu erreichen (vergl. Abildung 7.2). Die RF-Form wurde in 4000 Punkten pro Phasenanteil digitalisiert, d.h. das Zeitinkrement zwischen zwei RF-Werten beträgt 0,05 μs.

Um die Zielsetzung eines möglichst breiten Anregungsprofiles zu erreichen, wurden die Parameter der Methode `addOffset` so gewählt, daß ein Bereich von Offset-Frequenzen $\nu_{\text{off}} \in [-20\text{ kHz}, 20\text{ kHz}]$ berücksichtigt wurde. Dieser wurde in 81 Stützpunkten diskretisiert, d.h. $\Delta\nu_{\text{off}} = 500$ Hz. Zusätzlich wurde die maximal erlaubte Amplitude der RF-Form mit `limitRF` auf 20 kHz begrenzt.

Die Inhomogenität des B_1-Feldes wurde durch die Verwendung der Methode `setB1errGauss` für eine maximale Abweichung von $\pm$ 5% modelliert. Hierbei wird eine Fehlerverteilung nach Gauss mit einer Halbwertsbreite von 0,1 berechnet. Diese Fehlerverteilung wurde in 5 Datenpunkte digitalisiert.

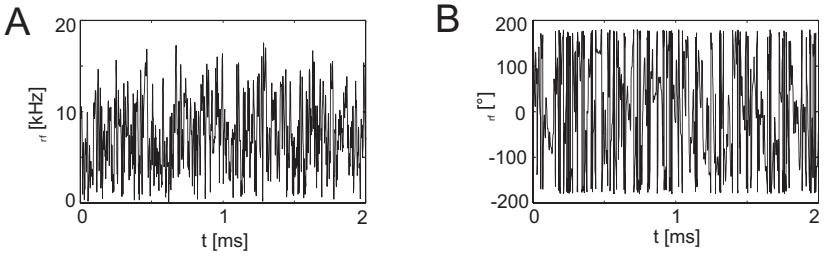

ABBILDUNG 7.3: Optimierte Pulsform BEBOP (Broadband Excitation By Optimized Pulses). Während der numerischen Optimierung wurden 8.000 Parameter berücksichtigt. Die RF-Form erreicht das Optimierungsziel $I_z \rightarrow I_x$ zu 99,5%. (A) Resultierende RF-Amplitude, berechnet aus den gemäß Gleichung (7.4) einzeln optimierten Anteilen $\nu_x(t)$ und $\nu_y(t)$. Die maximale Amplitude beträgt $B_{1,max}$ = 17,5 kHz. Über den gesamten Zeitraum gemittelt beträgt die Wurzel aus dem mittleren Leisungsquadrat $\sqrt{\overline{B_1^2}}$ = 7,7 kHz. (B) Resultierende Phase, berechnet aus den relativen Anteilen der orthogonalen Komponenten.

Der Algorithmus `optimizeCG` wird als konvergiert angesehen, wenn die Minimierung der Kostenfunktion (7.3) eine Verbesserung $\varepsilon < 10^{-8}$ erreicht.

7.1.3 Die numerische Optimierung

In Abbildung 7.3 ist die optimierte RF-Form BEBOP (Broadband Excitation By Optimized Pulses) gezeigt. Für die Abbildung wurde aus den getrennt optimierten RF-Anteilen $\nu_x(t)$ und $\nu_y(t)$ die für jeden Zeitpunkt effektive RF-Amplitude und Phase des Pulses berechnet (Abbildung 7.3.A bzw Abbildung 7.3.B). Hierbei ergibt sich die maximale RF-Amplitude zu $B_{1,max} = 17{,}5$ kHz. Mittelt man alle RF-Amplituden über die Dauer der RF-Form, so erhält man als Wurzel aus dem mittleren Leisungsquadrat $\sqrt{\overline{B_1^2}} = 7{,}7$ kHz.

Die optimierte RF-Form erreicht das Optimierungsziel der Anregung gemäß Gleichung (7.1) über alle in die Optimierung eingeschlossenen RF-Offsets und B_1-Inhomogenitäten gemittelt zu 99,5%. Dies entspricht der bereits in Abbildung 7.2 erkennbaren Grenze für diese Art von Optimierung.

Neben diesem charakterischen mittleren Qualitätsfaktor sind die generellen Transfereigenschaften einer Pulsequenz von Interesse. Um diese Güte der numerisch optimierten RF-Form zu überprüfen, wurden die durch die RF-Form erreichten Anregungamplituden für einen erweiterten Bereich von Offset-Frequenzen und B_1-Inhomogenoitäten simuliert. Für das in Abbildung 7.4 gezeigte Profil wurde ein Offset-Bereich $\nu_{\mathrm{RF}} \in [-25\mathrm{kHz}, 25\mathrm{kHz}]$ und eine

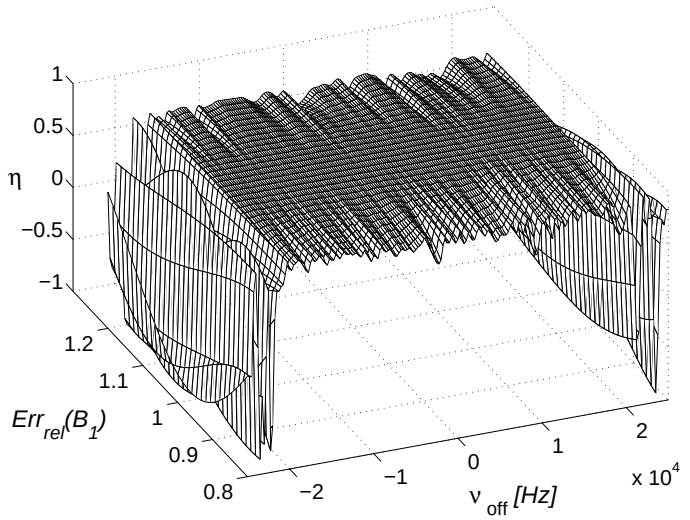

ABBILDUNG 7.4: Simuliertes Profil des Anregungspulses in Abhängigkeit der B_1-Inhomogenität $\mathrm{Err}_{\mathrm{rel}}(B_1)$ und der Offset-Frequenz ν_{off} (Offset-Bereich: $\nu_{\mathrm{RF}} \in [-25\ \mathrm{kHz}, 25\ \mathrm{kHz}]$, 201 Datenpunkte; Inhomogenitäts-Verteilung: $\pm 20\%$, 31 Datenpunkte). Man kann sehr deutlich das durch die Optimierungsparameter (Offset-Bereich: $\nu_{\mathrm{RF}} \in [-20\ \mathrm{kHz}, 20\ \mathrm{kHz}]$; Inhomogenitäts-Verteilung: $\pm 5\%$) bestimmte Plateau erkennen.

Inhomogenitäts-Verteilung von $\pm 20\%$ gewählt. Der Offsetbereich wurde hierbei in 201 Punkte digitalisiert (im Gegensatz zu 81 Datenpunkten während der Optimierung), die Inhomogenitäts-Verteilung wurde durch 31 Punkte dargestellt (im Gegensatz zu 5 Datenpunkten während der Optimierung).

Man erkennt sehr deutlich das durch die Optimierungsparameter begrenzte Plateau. In diesem Bereich ist die Anregung durch die RF-Form nahezu uniform. Die Stabilität gegenüber B_1-Inhomogenitäten ist auch über den optimierten Bereich hinaus noch erstaunlich gut. Dahingegen ist die Sequenz erheblich empfindlicher gegenüber Offset-Frequenzen, die außerhalb des optimierten Bereiches liegen.

7.2 Die experimentelle Überprüfung

Um die numerisch erzielten Ergebnisse experimentell zu überprüfen, wurde die optimierte Pulsform zur Erzeugung von Anregungsspektren verwendet. Hierbei wurde [U-$^{13}C_6$]-Lysin als Probe verwendet.

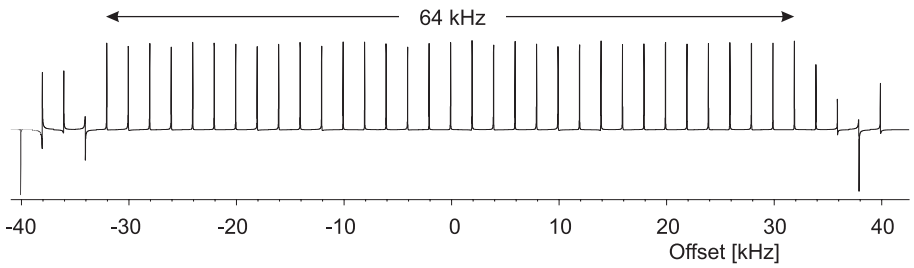

ABBILDUNG 7.5: Experimentelles Anregungsprofil der numerisch optimierten RF-Form. Im Experiment wurde der Puls so kalibriert, daß die Einstrahldauer 1,25 ms beträgt. Die erwartete Bandbreite beträgt 64 kHz (s. Text). Durch diese Kalibrierung erreicht man aufgrund höherer absoluter RF-Amplituden eine höhere Bandbreite. Die Anregung beträgt mehr als 93% über die gesamte Bandbreite von 64 kHz, wobei die Phasenfehler kleiner als 8° sind.

Zunächst wurde der Einfluß realer Offset-Frequenzen auf die Güte der Anregung getest. Die hierbei experimentell verwendete RF-Form wurde für eine Einstrahldauer von 1,25 ms kalibriert. Durch diese relative Verkürzung auf 62,5% (Skalierungsfaktor 5/8) der ursprünglichen Dauer von 2 ms steigt die erwartete Anregungsbandbreite von 40 kHz um 62,5% (Skalierungsfaktor 8/5) auf 64 kHz. Um nun die Abhängigkeit der Offset-Frequenz zu ermitteln, wird das NMR-Signal eines Kohlenstoff-Atoms bei einer bestimmten relativen Trägerfrequenz beobachtet. Durch systematische Variation der Trägerfrequenz erhält man das in Abbildung 7.5 gezeigte Profil. Man sieht eine über den gesamten Offset-Bereich stabile Anregung, die 93% nicht unterschreitet. Der hierbei maximal auftretende Phasenfehler beträgt 8°. Außerhalb des in der Optimierung eingeschlossenen Offset-Bereichs ist kein systematisches Anregungsverhalten zu beobachten. Wie bereits in den Simulationen (vergl. Abbildung 7.4) ist auch hier die Grenze zwischen optimiertem und nicht optimiertem Bereich sehr scharf.

Neben der Bandbreite der Anregung war die Stabilität gegenüber Inhomogenitäten der erzeugten B_1-Felder ein weiteres Optimierungskriterium. Um dieses experimentell zu überprüfen wurde die maximale RF-Amplitude durch Variationen der Dämpfungsfaktoren systematisch verändert. Die numerisch optimierte RF-Form wurde in diesem Fall auf eine Einstrahldauer von 4 ms kalibiriert (Skalierungsfaktor 2 gegenüber der ursprünglichen Dauer), wodurch die erwartete Anregungsbandbreite auf 20 kHz sinkt (Skalierungs-

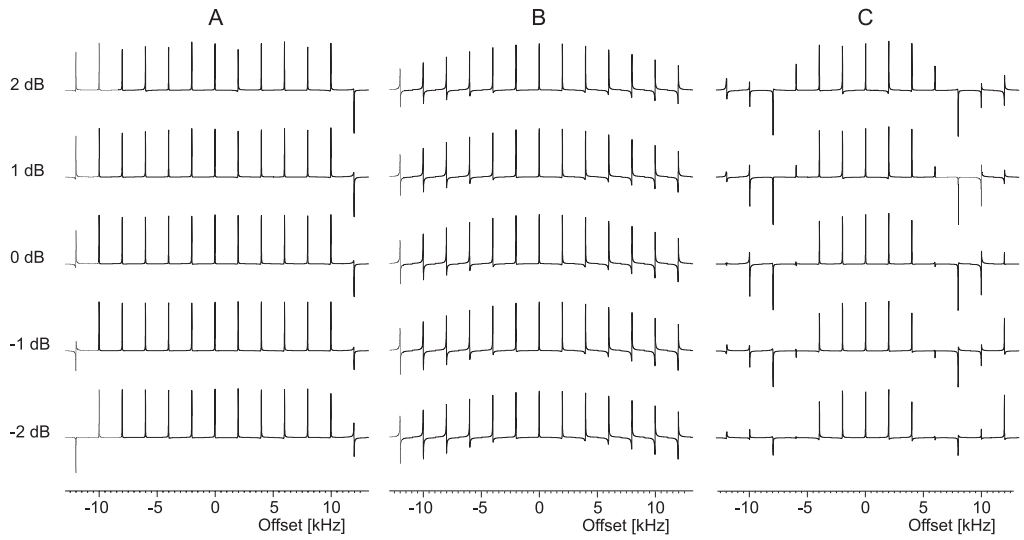

ABBILDUNG 7.6: Vergleich der Anregungsprofile verschiedener Anregungspulse. Zum einen wurde die Abhängigkeit von der Offset-Frequenz ermittelt (Abszisse), zum anderen die Stabilität gegenüber B_1-Inhomogenitäten (Ordinate). Letztere wird durch eine gewollte Mißkalibrierung der maximalen RF-Amplitude bei gleicher Einstrahldauer erreicht. (A) Numerisch optimierte RF-Form mit einer auf eine Einstrahldauer von 4 ms kalibrierten Leitung. Die erwartete Bandbreite beträgt 20 kHz (s. Text). (B) Harter ($\pi/2$)-Puls. (C) Zusammengesetzter Puls $24_x152_{-x}346_x152_{-x}24_x$ [44].

faktor 1/2). In Abbildung 7.6.A sind die experimentellen Ergebnisse für die optimierte Pulsform zusammengefaßt. Diesen werden in Tafel B und C analoge Ergebnisse unter Verwendung bekannter Anregungspulse gegenübergestellt. Zum einen handelt sich dabei um einen harten ($\pi/2$)-Puls (Tafel B), zum anderen um einen zusammengesetzten Puls (Tafel C). Die numerisch optimierte RF-Form deckt zum einen wiederum die erwartete Bandbreite vollständig ab, darüber hinaus zeigt die Variation der B_1-Felder nur geringen Einfluß auf die Güte der Anregung über den gesamten Offset-Bereich. Dieses Verhalten bestätigt das der Simulation (vergl. Abbildung 7.4). Im Gegensatz zu der scharfen Grenze der durch die Optimierung erzwungenen Bandbreite, ist der Einfluß von Amplituden-Variationen der B_1-Felder erheblich geringer.

7.3 Schlußfolgerung

Die Optimierungs-Bibliothek OCTANE wurde zur numerischen Optimierung eines Anregungspulses mit hoher Bandbreite und Toleranz gegenüber B_1-Inhomogenitäten verwendet. Außer einer für dieses Optimierungsziel speziell angepaßten Kostenfunktion wurden nur Standard-Methoden der Bibliothek verwendet. Die optimierte RF-Form zeigte bereits in den Simulationen sehr gute Eigenschaften, die durch die experimentellen Daten bestätigt werden konnten.

Durch die Ergebnisse konnte gezeigt werden, daß sich OCTANE über die in Kapitel 5 betrachteten Ideal-Fälle hinaus auch zur Optimierung von Pulssequenzen unter dem Einfluß von Störeffekten produktiv einsetzen läßt.

Teil III

Experimente

Kapitel 8

Zeit-optimaler Inphase-Transfer in IS-Spinsystemen

Um die Sensitivität von NMR-Experimenten zu maximieren, ist die Verwendung optimaler Pulssequenz-Elemente für jeden einzelnen Kohärenztransfer-Schritt notwendig [45]. Für den Fall, daß Relaxation vernachlässigbar ist, kann man die theoretisch möglichen oberen Grenzen eines Transfers zwischen einem beliebigen Anfangs- und Endzustand des Dichteoperators bestimmen [46, 47, 48]. Für die Implementation eines Experimentes hingegen ist es nun erstrebenswert, diese theoretisch maximal mögliche Transferamplitude in möglichst kurzer Zeit zu erreichen, um die durch die Relaxations-Eigenschaften des Moleküls zeitgleich zum Kohärenztransfer stattfindende Dekohärenz auf ein Minimum zu beschränken [49]. Im Rahmen von NMR-Untersuchungen großer Moleküle (z.B. der Strukturparameter-Bestimmung von Biomolekülen) liegen die Relaxationsraten im Bereich der Größe typischer Kopplungskonstanten, d.h. der Prozeß der Dekohärenz ist vergleichbar effizient wie der gewünschte Kohärenztransfer unter Ausnutzung der Kopplungs-Eigenschaften der Spins.

Ein häufig gewählter Ansatz, um einen akzeptablen Kompromiß zwischen Relaxation und Kohärenztransfer zu erreichen, ist die Verkürzung der im Experiment verwandten Mischzeiten τ_{mix}. Grundlage dieses Vorgehens ist die experimentelle Erfahrung, daß Aufbaukurven in solchen Fällen tatsächlich ein Maximum für kürzere Mischzeiten aufweisen als die für den vollständigen Transfer unter idealen Bedingungen benötigten. In der Regel ist es allerdings so, daß dieses durch linerares Verkürzen einer Sequenz erreichte Transfer-Maximum keinesfalls die für diese Mischzeit erreichbare obere Transfergrenze

darstellt.

Zur Lösung dieses Problems gilt es die damit verknüpften theoretischen und praktischen Fragen zu beantworten:

- Was ist die kürzest mögliche Mischzeit τ^*_{mix} mit der noch die unitäre Grenze η^* [47, 48] des erwünschten Transfers erreicht werden kann?
- Wie ist der zeitliche Verlauf der maximal möglichen Transferamplitude $\eta_{opt}(\tau_{mix})$ für Mischzeiten $\tau_{mix} < \tau^*_{mix}$ und was sind die zugehörigen unitären Transformationen?
- Durch welche experimentellen Bausteine können diese optimalen unitären Transformationen für $\tau_{mix} < \tau^*_{mix}$ umgesetzt werden?

Bezüglich der ersten beiden Punkte wurden bereits analytische Lösungen für den Fall des zeitoptimalen Kohärenzordnungs-selektiven Transfers in heteronuklearen 2-Spinsystemen (IS Spinsystem) formuliert [50]. In diesem Kapitel wird nach einer kurzen Besprechung der theoretischen Grundlagen die Übersetzung dieser theroretischen Ergebnisse in praktisch einsetzbare Pulssequenzen diskutiert. Die mit diesen Pulssequenzen erzielten experimentellen Ergebnisse werden zum einen mit den theoretischen Voraussagen, zum anderen mit den durch konventionelle Verkürzung der Mischzeiten erreichbaren Transferamplituden für gleiche Gesamt-Mischzeiten verglichen.

8.1 Theorie

8.1.1 Die optimale Lösung

Es wird ein heteronukleares IS Spinsystem mit einer sklaren Kopplungskonstante J betrachtet. Die beiden Spins haben die Offset-Frequenzen ν_I und ν_S. Der entsprechende das System beschreibende Hamilton-Operator hat die Form

$$\mathcal{H}_0 = 2\pi\nu_I I_z + 2\pi\nu_S S_z + 2\pi J I_z S_z. \tag{8.1}$$

Der zu untersuchende Kohärenzordnungs-selektive Transfer von Inphase-Kohärenzen ausgehend vom S-Spin zum I-Spin $S^- \rightarrow I^-$ wird durch folgende Dichteoperatoren beschrieben:

$$[\rho(0) = S_x - \imath S_y] \longrightarrow [\rho(T) = I_x - \imath I_y]\,. \tag{8.2}$$

Die normierte absolute Transferamplitude für eine Mischzeit τ_{mix} für diesen Transfer unter Ausnutzung des Hamilton-Operators (8.1) ist gegeben durch [48]:

$$\begin{aligned} \eta(\tau_{mix}) &= \frac{|\langle I^-|\rho(\tau_{mix})\rangle|}{\|S^-\|\,\|I^-\|} \\ &= \frac{1}{2}\mathrm{Tr}\{I^+\rho(\tau_{mix})\}. \end{aligned} \tag{8.3}$$

Das Ziel ist es nun, eine optimale Pulssequenz zu finden, die die Transferamplitude $\eta(\tau_{mix})$ für jede mögliche Mischzeit τ_{mix} maximiert. Beruhend auf Prinzipien der geometrischen Steuerungstheorie wurde in [51] gezeigt, daß der optimale Hamilton-Operator $\mathcal{H}_{opt}$ in Abhängigkeit der Mischzeit τ_{mix} formuliert werden muß und die allgemeine Form

$$\mathcal{H}_{opt} = \frac{2\pi J}{\tau_{mix}}\left(\tau_x I_x S_x + \tau_y I_y S_y + \tau_z I_z S_z\right) \tag{8.4}$$

hat. Hierbei wird die gesamte Mischzeit in drei Anteile gemäß

$$\tau_{mix} = \tau_x + \tau_y + \tau_z \tag{8.5}$$

zerlegt. Es konnte gezeigt werden, daß gleiche Mischzeit-Anteile wie sie in der konventionellen Sequenz-Verkürzung verwendet werden, nicht zu einer optimalen Transferamplitude führen. Um die Bedingung der Optimalität zu erfüllen, werden die planaren Anteile $\tau_{x,y}$ in Abhängigkeit des longitudinalen τ_z formuliert:

$$\tau_x = \tau_y = \tau_\perp = \frac{1}{J\pi}\arctan\left(\frac{1}{2}\tan\left(J\pi\tau_z\right)\right). \tag{8.6}$$

In Abb. 8.1 ist diese funktionale Abhängigkeit der longitudinalen und transversalen Mischzeit von der Gesamt-Mischzeit aufgeführt. Hierbei wird deutlich, daß die Mischzeit-Anteile $\tau_\perp$ und τ_z zur Erreichung einer für alle Mischzeiten τ_{mix} optimalen Transferamplitude ungleichmäßig verändert werden müssen, was die Ungültigkeit des konventionellen Ansatztes unterstreicht.

Unter Verwendung dieser anteilmäßigen Mischzeiten gemäß Gleichung (8.6) ergibt sich für jede Gesamt-Mischzeit $\tau_{mix} < 1.5J^{-1}$ die zugehörige optimale Transferamplitude als

$$\eta_{opt}(\tau_{mix}) = \sin\left(J\pi\tau_z\right)\sin\left(J\pi\tau_\perp\right). \tag{8.7}$$

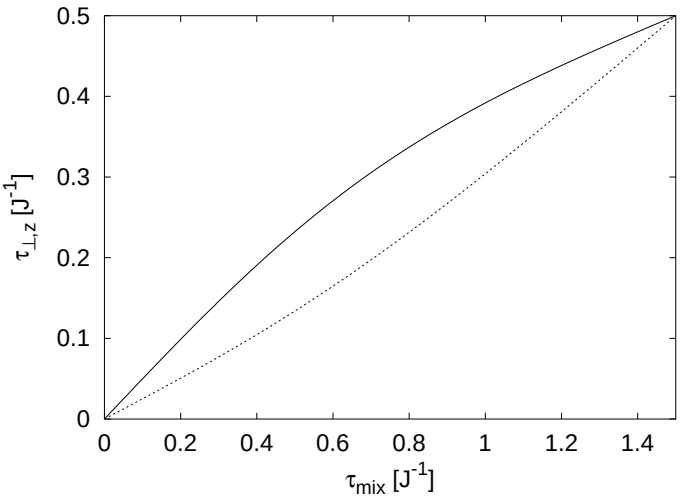

ABBILDUNG 8.1: Grafische Darstellung der optimalen Mischzeiten τ_z (durchgezogene Linie) und $\tau_\perp = \tau_x = \tau_y$ (gepunktete Linie) als Funktion der Gesamt-Mischzeit τ_{mix}.

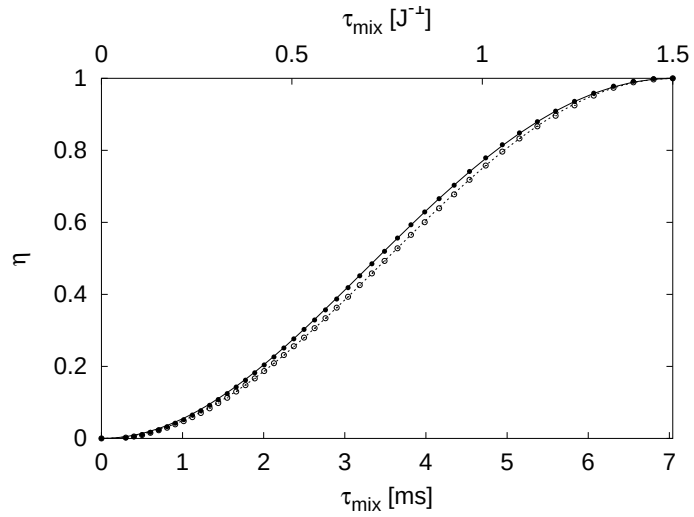

ABBILDUNG 8.2: Vergleich der konventionellen isotropen Transferamplitude η_{iso} (Theorie: gestrichelte Linie, berechnet nach Gleichung (8.10), experimentelle Daten: offene Punkte) mit der des optimalen Kohärenztransfers η_{opt} (Theorie: durchgezogene Linie, berechnet nach Gleichung (8.7); experimentelle Daten: gefüllte Punkte) als Funktion der Gesamt-Mischzeit τ_{mix} für eine skalare Kopplungskonstante $J = 213$ Hz.

Die funktionale Abhängigkeit der Transferamplituden von der Gesamtmischzeit τ_{mix} für sowohl den optimale Kohärenztransfer $\eta_{opt}(\tau_{mix})$ als auch den konventionellen isotropen $\eta_{iso}(\tau_{mix})$ ist in Abbildung 8.2 gezeigt.

Für Mischzeiten $\tau_{mix} \geq 1.5J^{-1}$ ist das Maixmum der Transferamplitude $\eta_{max}(\tau_{mix}) = 1$ invariant gegenüber der Gesamtmischzeit des Kohärenztransfers. Dies entspricht dem Erreichen der unitären Grenze des Kohärenztransfers [48]. Somit bestimmt sich die Mischzeit für den zeitoptimalen Kohärenztransfer zu

$$\tau^*_{mix} = 1.5J^{-1}. \tag{8.8}$$

8.1.2 Vergleich optimaler und konventioneller Lösung

Der optimale Hamilton-Operator $\mathcal{H}_{opt}$ erreicht für die Mischzeit $\tau_{mix} = \tau^*_{mix}$ gemäß Gleichungen (8.4-8.6) und Abbildung 8.1 den Grenzwert des isotropen Hamilton-Operators [52, 53, 54] mit $\tau_x = \tau_y = \tau_z = \tau_{mix}/3$.

$$\mathcal{H}_{iso} = \frac{2\pi J}{3} \left(I_x S_x + I_y S_y + I_z S_z\right) \tag{8.9}$$

Dies entspricht dem bekannten Befund, daß isotropes Mischen die unitäre Grenze für $\tau_{mix} = 1.5J^{-1}$ [54] erreicht. Daher ist es es interessant die durch Gleichung (8.7) gegebene optimale Transferamplitude $\eta_{opt}(\tau_{mix})$ mit der für heteronukleare isotrope Mischexperimente [54]

$$\eta_{iso}(\tau_{mix}) = \sin^2\left(\pi J \tau_{mix}/3\right). \tag{8.10}$$

zu vergleichen.

In Abbildung 8.2 sind die von der Gesamt-Mischzeit abhängigen Verläufe der Transferamplituden für diesen beiden Fälle ($\eta_{opt}(\tau_{mix})$ für das optimale und $\eta_{iso}(\tau_{mix})$ für das konventionelle Experiment) grafisch dargestellt. Um einen besseren Vergleich zwischen den beiden Misch-Experimenten zu ermöglichen, ist in Abbildung 8.3 das Verhältnis der Transferamplituden η_{opt}/η_{iso} gegen die Mischzeit aufgetragen. Aus dieser Abbildung wird ersichtlich, daß die Transferamplitude des optimalen Misch-Experimentes im Grenzfall kurzer Mischzeiten $\tau_{mix} \ll \tau^*_{mix}$ bis zu 12,5% höher ist als die des konventionellen Experimentes.

Vereinfachend kann der Einfluß von Relaxation durch eine zeitabhängige exponentielle Dämpfung mit einer einzigen Relaxationsrate Γ beschrieben

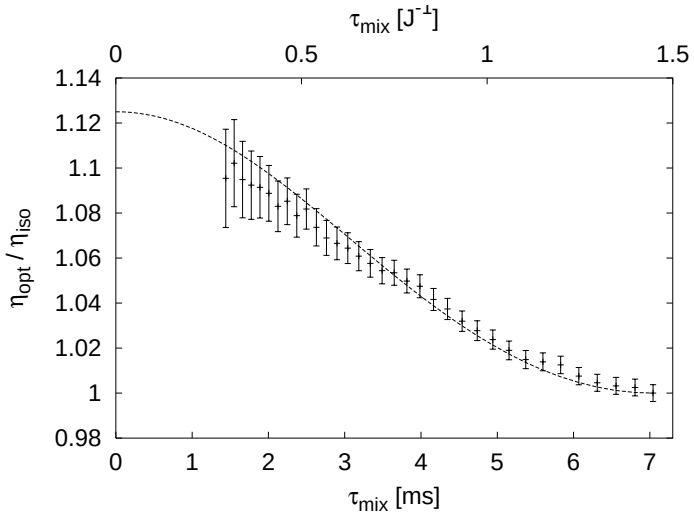

ABBILDUNG 8.3: Grafische Darstellung des Verhältnisses η_{opt}/η_{iso} der von der Gesamt-Mischzeit τ_{mix} abhängigen Transferamplituden des optimalen und des konventionellen isotropen Mischexperiments (Theorie: gestrichelte Linie, berechnet nach Gleichungen (8.5-8.7); experimentelle Werte: Kreuze, mit Fehlerbalken versehen, die durch das im Experiment auftretende Hintergrundrauschen bestimmt sind).

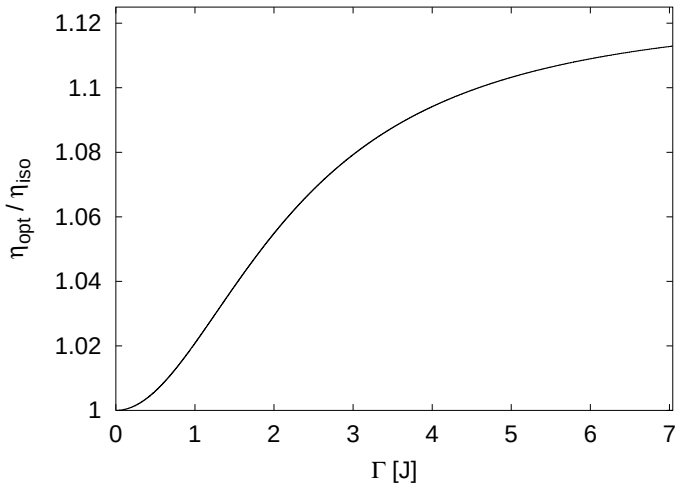

ABBILDUNG 8.4: Gewinn in der maximalen Transferamplitude des Misch-Experimentes in Abhängigkeit der Relaxationsrate Γ (in Einheiten der skalaren Kopplung) gemäß Gleichungen (8.11-8.12) durch Verwendung des optimalen Hamilton-Operators im Vergleich zum konventionellen isotropen Mischen.

werden. So ist es möglich aus den durch Gleichungen (8.7, 8.10) gegebenen ungedämpften Transferamplituden die Maxima der entsprechenden gedämpften Transferamplituden zu bestimmen:

$$\eta_{opt}^{\max}(\tau_{mix}) = \max\left\{\eta_{opt}(\tau_{mix}) \cdot \exp\left(-\Gamma\tau_{mix}\right)\right\}, \tag{8.11}$$

$$\eta_{iso}^{\max}(\tau_{mix}) = \max\left\{\eta_{iso}(\tau_{mix}) \cdot \exp\left(-\Gamma\tau_{mix}\right)\right\}. \tag{8.12}$$

Wiederum der besseren Vergleichbarkeit wegen wird in Abbildung 8.4 das Verhältnis dieser Maxima der gedämpften Transferamplituden $\eta_{opt}^{\max}/\eta_{iso}^{\max}$ als Funktion der Relaxationsrate Γ für einen Bereich $0 \leq \Gamma \leq 7J$ dargestellt. Dem Graphen kann man entnehmen, daß im Falle von Relaxationsraten im Bereich $\Gamma = 2J$ der maximale Gewinn durch Verwendung des optimalen Hamilton-Operators ca. 6% gegenüber dem konventionellen Ansatz beträgt, für Relaxationsraten $\Gamma > 5J$ übersteigt der Gewinn in der maximalen Transferamplitude bereits 10%.

8.2 Experimente

Um die theoretischen Voraussagen zu überprüfen, wurden der durch Gleichung (8.4) definierte optimale $\mathcal{H}_{opt}$ und der durch Gleichung (8.9) gegebene isotrope Hamilton-Operator $\mathcal{H}_{iso}$ unter Verwendung eines Modell IS-Spinsystems implementiert. Das Modell-System wird durch den natürlichen ^{13}CH-Anteil von Chloroform (70% $CHCl_3$ gelöst in in Acteon-d6, T=298 K) mit einer skalaren Kopplung von $J = 213$ Hz repräsentiert. Im Folgenden werden ^{13}C-Kerne als S-Spin und die Protonen als I-Spin bezeichnet.

8.2.1 Diskussion der Pulssequenz

Die zur Messung der Transferfunktionen verwendete Pulssequenz ist in Abbildung 8.5 gezeigt. Die Inphase-Kohärenz des S-Spins wurde durch heteronuklearen NOE-Transfer mit anschließendem $(\pi/2)_y$(S)-Puls präpariert. Der sich vor diesem Puls befindende Gradient zerstört die zu diesem Zeitpunkt vorhandene transversale Magnetisierung, d.h. die durch den Puls erzeugte x-Magnetisierung wird nicht verunreinigt ("Spoil Gradient"). Das heteronukleare Kohärenzordnungs-selektive Transfer-Echo [55, 56] wurde durch einen vor den Transfer-Block plazierten defokussierenden und einen nach dem Transfer refokussierenden Gradienten realisiert. Darüberhinaus wird so das

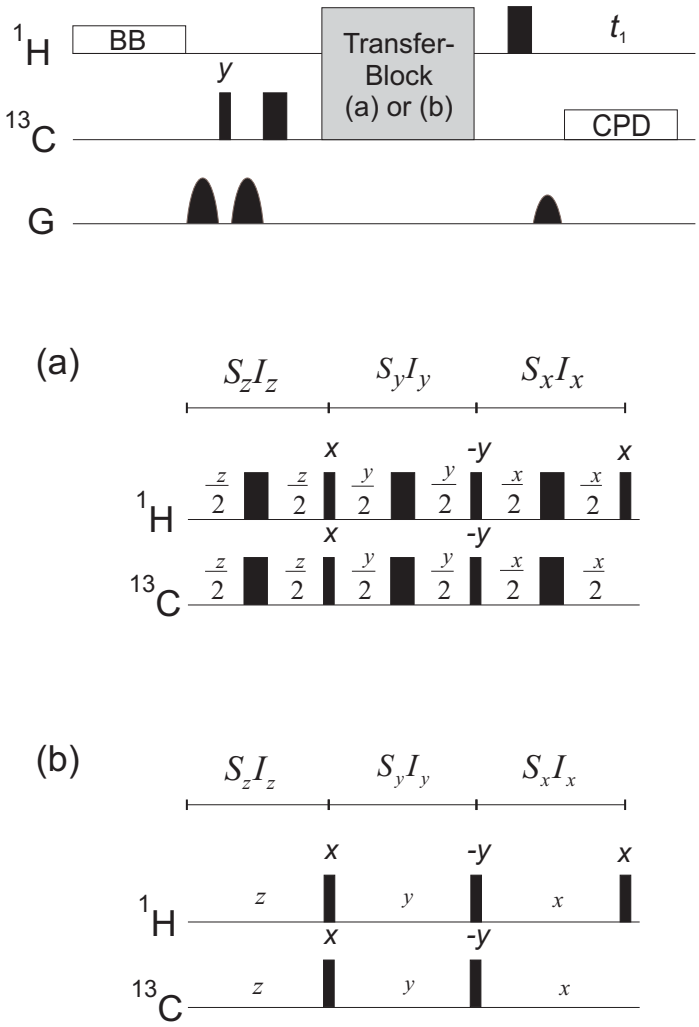

ABBILDUNG 8.5: Pulssequenz zur Implementierung des Kohärenzordnungsselektiven Inphase Transfers: (a) zyx-ICOS-CT Sequenz [57] und (b) vereinfachte Sequenz für Offset-Frequenzen $\nu_I = \nu_S = 0$ Hz. Die Mischzeiten für den optimalen Transfer werden durch Gleichung (8.6) definiert. Die konventionelle isotrope Pulssequenz wird durch die Mischzeiten $\tau_x = \tau_y = \tau_z = \tau_{mix}/3$ erreicht.

Signal der $^{12}CHCl_3$-Protonen unterdrückt, weil hier der letzte Gradient die transversale Magnetisierung der ^{12}C-gebundenen Protonen dephasiert, jedoch die der ^{13}C-gebundenen refokussiert (vor Transfer-Beginn ist keine transversale Protonen-Magnetisierung vorhanden, die durch den Transfer erreichte liegt jedoch dephasiert vor und wird durch diesen Gradienten rephasiert). Die Protonen-Signale werden unter Entkopplung von Kohlenstoff aufgenommen. In dem bisher noch nicht näher diskutierten mittleren Block (vergl. die graue Box in Abbildung 8.5) können nun verschiedene Pulssequenz-Elemente verwendet werden, um ihre Transfer-Eigenschaften bezüglich des Inphase-Transfers zu messen.

Experimentelle Transfer-Blocks, die den optimalen Misch-Hamiltonian $\mathcal{H}_{opt}$ implementieren, können von bekannten Pulssequenz-Elementen [57] dadurch abgeleitet werden, daß die Mischzeiten τ_x, τ_y und τ_z gemäß Gleichung (8.6) angepaßt werden. In den durchgeführten Experimenten wurde die zyx-ICOS-CT Pulssequenz (vergl. Abildung 8.5.a und [57]) als Basis für die optimalen und isotropen Misch-Sequenzen gewählt. Man beachte, daß diese Sequenz selbst für Mischzeiten $\tau_x = \tau_y = \tau_z$ nur dann einen wirklich isotropen Hamilton-Operator implementieren würde, wenn am Ende der Sequenz ein $(\pi/2)_x$(S)-Puls gefolgt von einer $(\pi/2)_z$(S)-Rotation eingeführt werden würde. Jedoch haben diese Pulse für die Messung des Transfers von S^- (oder S^+) $\rightarrow I^-$ keinen Einfluß auf die Transfereffizienz und können daher ausgelassen werden. Desweiteren wurde auch die Anzahl der Pulse auf ein Minimum beschränkt, um weitere experimentelle Imperfektionen auszuschließen (z.B. die B_1-Inhomogenitäten der RF-Spulen). Da die Offsetfrequenzen der beiden Spins im IS-System auf 0 Hz (d.h. $\nu_I = \nu_S = 0$ Hz) gesetzt werden können, entfällt die Notwendigkeit, die durch die Offset-Terme in $\mathcal{H}_0$ auftretenden Effekte durch die sechs π-Pulse zu refokussieren. Da allerdings in dem so modifizierten Experiment eine ungerade Anzahl von π-Pulsen pro Kanal entfernt wurde (vergl. Abbildung 8.5.b, es werden drei π-Pulse pro Kanal ausgelassen), kehrt sich die Kohärenzordnung um, d.h. der stattfindende Transfer entspricht $S^\pm \rightarrow I^\mp$ anstatt dem gewünschten Transfer $S^\pm \rightarrow I^\pm$. In den durchgeführten Experimenten wurde diese Tatsache dadurch kompensiert, daß das Vorzeichen des Refokussierungs-Gradienten vor der Detektion invertiert wurde. Eine andere Möglichkeit wäre es gewesen, die Phase des letzten $(\pi/2)$(I)-Pulses zu invertieren, um den Transfer$S^\pm \rightarrow I^\pm$ zu erreichen.

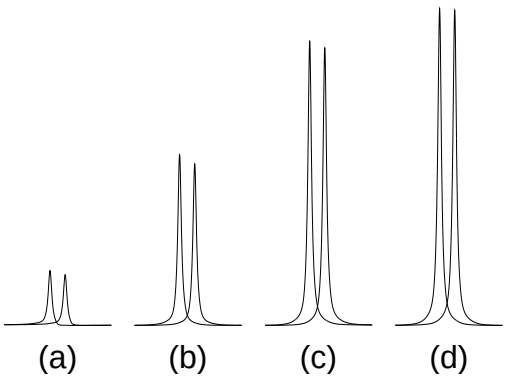

ABBILDUNG 8.6: ^{1}H-NMR-Spektren von $^{13}CHCl_3$ nach optimalem (linkes Signal) und isotropem (rechtes Signal) Kohrenzordnungs-selektiven Inphase Transfer ausgehend von ^{13}C unter Verwendung von Mischzeiten τ_{mix} (a) $1,76$ ms$= 0.267\tau^*_{mix}$, (b) $3,52$ ms$= 0.515\tau^*_{mix}$, (c) $5,28$ ms$= 0.788\tau^*_{mix}$ und (d) $7,04$ ms$= \tau^*_{mix}$.

8.2.2 Diskussion der Ergebnisse

Um die Mischzeit-abhängigen Transferkurven für sowohl den optimalen als auch isotropen Inphase-Transfer zu messen, wurden NMR-Spektren für 47 verschiedene Mischzeiten $\tau_{mix} \leq \tau^*_{mix}$ mit je 42 Scans und einem Relaxations-Delay von 30s aufgenommen. In Abbildung 8.6 werden beispielhaft Spektren für Mischzeiten $\tau_{mix} \approx (1/4)\tau^*_{mix}$, $\tau_{mix} \approx (1/2)\tau^*_{mix}$, $\tau_{mix} \approx (3/4)\tau^*_{mix}$ und $\tau_{mix} = \tau^*_{mix}$ gezeigt. Um die Effizienz des Transfers zu quantifizieren, wurden die entprechenden Signale integriert und auf einen direkt erzeugten Zielzustand normiert (d.h. die Werte liegen im Bereich [0,1] und sind direkt mit den berechneten Transferamplituden η vergleichbar). In Abbildung 8.2 sind diese Intergrale für den Fall des optimalen Transfers als Punkte, für den des isotropen Transfers als Kreise eingetragen. Man sieht eine gute Übereinstimmung zwischen den theoretischen und experimentellen Werten. Um einen besseren Vergleich der relativen Transferamplituden der beiden Pulssequenzen zu ermöglichen, sind diese in Abbildung 8.3 aufgeführt. Die dort eingezeichneten Fehlerbalken berechnen sich aus dem Signal/Rausch-Verhältnis, das durchaus die Intergrale beeinflussen kann. Wiederum kann man eine gute Übereinstimmung zwischen Theorie und Experiment feststellen. Die vorhandene systematische Abweichung für kleine Mischzeiten können einer unvollständigen Unterdrückung der $^{12}CHCl_3$ Protonen-Signale zugeschrieben werden. Es

wurden keine weiteren Versuche unternommen, diese verbleibenden Artefakte zu korrigieren.

8.3 Schlußfolgerung

Es konnte gezeigt werden, daß sich die Prinzipien der geometrischen Steuerungstheorie nicht nur zur Ableitung der bestmöglichen Kohärenztransfer-Amplituden für Spinsysteme aus zwei gekoppelten Spins anwenden lassen [50], sondern sich diese Erkenntnise auch experimentell verifizieren lassen. Verglichen mit konventionell eingesetzten Experimenten (isotropes Mischen) wurden Verbesserungen der Transferamplituden des Inphase-Transfers $S^- \rightarrow I^-$ in IS-Spinsystemen von bis zu 12,5% vorausgesagt und experimentell bestätigt. Mit den durchgeführten Experimenten wurde darüberhinaus gezeigt, daß bereits existierende Pulssequenz-Elemente durch einfache Anpassungen der experimentellen Parameter in zeitoptimale Pulssequenzen überführt werden können.

Selbst geringe Verbesserungen der Transferamplituden eines Transfer-Blocks können in der Praxis von großer Bedeutung sein, insbesondere wenn mehrere solcher Transferschritte in einem Experiment sequentiell durchgeführt werden sollen. So kann die insgesamt benötigte Meßzeit eines Experimentes mit drei aufeinander folgenden Kohärenzordnungs-selektiven Inphase-Transfer Schritten ohne Verlust an Gesamt-Sensitivität um mehr als 25% (50%) verringert werden, wenn jeder dieser Transfer-Schritte eine um 5% (12,5%) verbesserte Transfer-Amplitude aufweist.

Diese Untersuchung zeigt darüber hinaus, daß die übliche Strategie der linearen zeitlichen Stauchung zur Minimierung der Einflüsse von Relaxation zu nicht optimalen Transfer-Eigenschaften führt. Generell ist es nicht ausreichend, einfach die Misch-Perioden eines Transferschrittes zu verkürzen, um einen Kompromiß zwischen Kohährenztransfer und durch Relaxation bedingten Verlusten zu finden. Unter Umständen muß die Pulssequenz modifiziert werden, um die maximal mögliche Transferamplitude zu erreichen.

In den in diesem Kapitel diskutierten Sequenzen wurde der Einfluß von Relaxation der Einfachheit halber als eine exponentielle Dämpfung mit einer einzigen Relaxationsrate Γ dargestellt. Durch diese Betrachtung wird die Tatsache vernachlässigt, daß es Terme im Dichteoperator eines Spinsystems

gibt, die langsamer als andere relaxieren. Wie in Kapitel 6 gezeigt, kann man durch Einschluß konkreter Relaxationsmechanismen auf quantenmechanischer Basis bereits während der Entwicklung von Pulssequenzen weitere Sensitivitätsgewinne erreichen.

Kapitel 9

Zeitoptimale Implementierung eines SWAP(1,3)-Gatters

Wie bereits in Kapitel 8 gezeigt, kann man durch Anwendung der geometrischen Optimale-Kontroll-Theorie die Transfereigenschaften von Pulssequenzen erheblich verbessern. In diesem Kapitel wird die Implementierung einer zeitoptimalen geodätischen Pulssequenz für ein linear gekoppeltes 3-Spinsystem vorgestellt (vergl. auch die in Kapitel 5 vorgestellten numerischen Simulationen bezüglich dieser Kopplungtopologie).

Die Motivation, eine zeitoptimale Pulssequenz zu verwenden, wurde bereits in Kapitel 8 erwähnt. Im Allgemeinen versucht man die Sensitivität eines Experimentes dadurch zu maximieren, daß man die diesem Ziel entgegengesetzten Einflüsse minimiert. Da einer der wichtigsten Effekte, die die Sensitivität eines NMR-Experimentes herabsetzen, das Relaxationsverhalten eines molekularen Systems ist, ist es erstrebenswert, den gewünschten Kohärenztransfer in möglichst kurzer Zeit durchzuführen (es gehen weniger Informationen durch Dekohärenz verloren). Durch quantenmechanische Betrachtungen unter Vernachlässigung von Relaxation kann man nun die unitäre Grenze η_{uni} eines Kohärenztransfers für ein bestimmtes System finden [58, 59]. Bezüglich der Entwicklung zeitoptimaler NMR-Experimente ist die Kenntnis der maximalen Transferamplitude nur eine Teilinformation, es fehlt weiterhin die Information, welches die kürzest mögliche Zeit ist, in der man diesen maximalen Kohärenztransfer erreichen kann. Durch die inhärente Quantendynamik eines Spinsystems, welche durch die Kopplungseigenschaften der Kernspins definiert wird, muß es eine Zeit geben, deren Unterschreitung die Transferamplitude η in einen zeitabhängigen Bereich mit $\eta < \eta_{\text{uni}}$

überführt (vergl. Kapitel 5-6). Eine Pulssequenz, die die unitäre Grenze des Kohärenztransfers für die minimal notwendige Zeit erreicht, bezeichnet man als zeitoptimal. Speziell in der Quanteninformatik ist die Verfügbarkeit zeitoptimaler Operationen von großem Interesse, weil der Informations-Verlust jeder Operation durch z.B. Dekohärenz die generelle Brauchbarkeit eines Quantenalgorithmus einschränkt.

Ein in der Quanteninformatik häufig auftretendes Problem ist die Erzeugung trilinearer Propagatoren [64, 65, 66]. Diesbezügliche zeitoptimale Lösungen wurden für eine linear gekoppelte Spinkette für Spin 1/2-Teilchen theoretisch abgeleitetet [62]. Konventionelle Ansätze zur Erzeugung der benötigten Hamilton-Operatoren beruhen auf der Entkopplung bestimmter Spinbeziehungen während der Pulssequenz. Dies erweist sich als zeitlich uneffizient. Man kann nun diese konventionellen Pulssequenzen verbessern, indem man die Notwendigkeit der Entkopplung aufhebt, wodurch die Sequenz erheblich verkürzt werden kann. Verwendet man nun die geometrische Optimale-Kontroll-Theorie zur Lösung des Problems, so kann man Sequenzen entwickeln, deren Zeitoptimalität gezeigt werden kann.

In diesem Kapitel wird die praktische Implementierung dieser Sequenzen vorgestellt. Hierbei mußten einige Änderungen eingeführt werden, die einen größeren Bereich von Offset-Frequenzen der Kerne erlaubt. Diese breitbandigen Pulssequenzen werden hierbei sowohl durch numerische Simulationen als auch durch experimentelle Daten charaktersiert.

9.1 Theorie

9.1.1 Grundlagen

Als zu Grunde liegendes Spinsystem wird eine Kette von drei heteronukleraren Spins mit einer Kopplungstopologie $J_{1,2} = J_{2,3} = J$ und $J_{1,3} = 0$ angenommen. Die einzelnen Kerne besitzen die Resonanz-Frequenzen ν_1, ν_2 und ν_3. Im mehrfach rotierenden Koordinatensystem [63] ergibt sich der freie Evolutions-Hamiltonian $\mathcal{H}_0$ zu

$$\mathcal{H}_0 = \mathcal{H}_{\text{coup}} + \mathcal{H}_{\text{off}}, \tag{9.1}$$

wobei der Anteil der skalaren Koppung $\mathcal{H}_{\text{coup}}$ durch

$$\mathcal{H}_{\text{coup}} = 2\pi J I_{1z} I_{2z} + 2\pi J I_{2z} I_{3z} \tag{9.2}$$

und der Offset-Hamiltonian $\mathcal{H}_{\text{off}}$ durch

$$\mathcal{H}_{\text{off}} = 2\pi\nu_1 I_{1z} + 2\pi\nu_2 I_{2z} + 2\pi\nu_3 I_{3z} \tag{9.3}$$

gegeben ist. Der Hamilton-Operator (9.1) gilt auch für den Fall, daß der erste und dritte Kernspin homonuklear sind, während nur der zweite Kern heteronuklear ist.

Für viele Anwendungen in der NMR-Spektroskopie und Quanteninformatik [64, 65, 66] ist es erstrebenswert, trilineare effektive Hamilton-Operatoren der Form

$$\mathcal{H}_{\alpha\beta\gamma} = 2\pi J^{\text{eff}}_{123}\ I_{1\alpha} I_{2\beta} I_{3\gamma} \tag{9.4}$$

zu erzeugen. Hierbei sei $\{\alpha, \beta, \gamma\} \in \{x, y, z\}$ und J^{eff}_{123} eine effektive trilineare Kopplungskonstante. Die Generierung eines solchen effektiven Hamilton-Operators ermöglicht die Implementierung trilinearer Propagatoren der Form

$$\mathcal{U}_{\alpha\beta\gamma}(\kappa) = \exp\{-\imath\kappa\ 2\pi\ I_{1\alpha} I_{2\beta} I_{3\gamma}\} \tag{9.5}$$

in einer Zeit

$$\tau = \frac{\kappa}{J^{\text{eff}}_{123}}. \tag{9.6}$$

Beispielsweise benötigt die Implementierung eines Λ_2-Gatters [67] die Erzeugung eines trilinearen Propagators $\mathcal{U}_{zzz}(1)$. Indirekte SWAP-Gatter können nun (*vide infra*) durch die trilinearen Propagatoren $\mathcal{U}_{xzx}(1)$, $\mathcal{U}_{yzy}(1)$ und $\mathcal{U}_{zzz}(1)$ erzeugt werden [62].

9.1.2 Diskussion der Pulssequenzen

Das Ziel einer SWAP-Operation ist es im Allgemeinen den Inhalt zweier Qubits auszutauschen ohne dabei die Inhalte weiterer Qubits zu beeinflussen. In der NMR-Quanteninformatik werden die Werte eines Qubits durch die möglichen Zustände des magnetischen Kernmoments dargestellt, was die Zielsetzung eines SWAP-Gatters als Austausch der Spinzustände zweier Kerne formulieren läßt. Hierbei werden die Möglichkeiten der Implementierung einer solchen Operation dadurch erschwert, wie sich diese Spinzustände durch ein Spinsystem bewegen lassen. Es kommen nur physikalischen Pfade in Frage wie z.B. die Ausnutzung der Fermi-Kontakte der Kerne mit den Elektronen

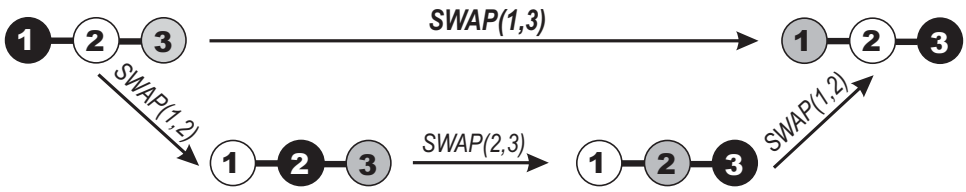

ABBILDUNG 9.1: Erzeugung einer indirekten SWAP(1,3)-Operation durch sequentielle Anwendung direkter SWAP-Operationen gemäß Gleichung (9.7).

(manifestiert in skalaren Kopplungen). Im Falle einer SWAP(1,2)-Operation ist die Umsetzung recht unproblematisch, weil bei der Verwendung zweier direkt gekoppelter Spins der Bedarf an intermediärer Spins als temporäre Register entfällt, deren ursprünglicher Spinzustand wieder hergestellt werden muß. Diese sog. direkten SWAP-Gatter sind Gegenstand aktueller Forschung [68, 69, 70].

Da die in der NMR-Quanten Informatik verwendeten Spinsysteme immer mehr Spins (Qubits) enthalten, ist es nun von Interesse SWAP-Operationen zu entwickeln, die es ermöglichen auch die Zustände entfernter Spins auszutauschen. Eine Möglichkeit dieses Ziel zu erreichen ist es, direkte SWAP-Gatter sequentiell so anzuwenden, daß nach der Operation nur die Zustände der gewünschten Spins vertauscht sind. Im Falle eines 3-Spinsystems könnte man eine SWAP(1,3)-Operation z.B. durch drei sequentielle direkte SWAP-Gatter erreichen (vergl. auch Abbildung 9.1):

$$\mathrm{SWAP}(1,3) = \mathrm{SWAP}(1,2) \times \mathrm{SWAP}(2,3) \times \mathrm{SWAP}(1,2) \tag{9.7}$$

Hierbei ist die Dauer der gesamten Operation durch $3 \cdot 3/2J = 4.5/J$ gegeben. Erhöht sich nun die Zahl der beteiligten Kernspins, so wird eine solch sequentielle Implementation immer mehr Zeit benötigen und daher sehr viel anfälliger für Informationsverluste durch Relaxation.

Um diesen Effekt zu minimieren, ist es erstrebenswert, die Erzeugung des trilinearen Propagators (9.5) zeitlich zu verkürzen und im Grenzfall zeitoptimal zu gestalten. In Abbildung 9.2 ist die konventionelle Implementierung dieses Propagators gezeigt. Man erkennt deutlich zwei Perioden Δ_1 und Δ_2, während denen selektiv je eine Kopplung aufgehoben wird (J_{23} während Δ_1 bzw. J_{12} während Δ_2). Die Gesamtdauer dieser Pulssequenz beträgt allge-

mein

$$\tau_A(\kappa) = 2 \cdot \Delta_1 + \Delta_2 = \frac{2+\kappa}{2J}, \tag{9.8}$$

bzw. für den Fall des für ein SWAP-Gatter benötigten trilinearen Propagators mit $\kappa = 1$

$$\tau_A(1) = \frac{3}{2J}. \tag{9.9}$$

Eine erste Verbesserung läßt sich dadurch erzielen, daß man die Notwendigkeit der partiellen Entkopplung des Spinsystems aufhebt. Die entsprechende Pulssequenz ist in Abbildung 9.3 gezeigt. Ein weiterer Vorteil dieser Sequenz liegt in der Tatsache, daß man nun die die Kerne $I_{1,3}$ bedienenden RF-Kanäle nicht mehr benötigt. D.h. in dem Falle, daß diese Kerne homonuklerar sind, entfällt der Einsatz selektiver Pulse, die noch in der in Abbildung 9.2 gezeigten Pulssequenz nötig sind. Der gesamte Zeitbedarf der Erzeugung eines trinlinearen Propagators mittels dieser Pulssequenz beträgt allgemein

$$\tau_B(\kappa) = 2 \cdot \Delta_1 + \Delta_2 = \frac{1+\kappa}{2J}, \tag{9.10}$$

bzw. für den Fall des für ein SWAP-Gatter benötigten trilinearen Propagators mit $\kappa = 1$

$$\tau_B(1) = \frac{1}{J}. \tag{9.11}$$

was einer Verringerung der Gesamtdauer auf 66,6% gegenüber der konventionellen Pulssequenz bedeutet.

In [62] konnte jedoch gezeigt werden, daß die in Abbildung 9.4 gezeigte Pulssequenz zeitoptimal ist. Hierbei wird die Magnetisierung des Kerns auf einer geodätischen Bahn kontrolliert bewegt, welcher der kürzest möglichen entspricht. Hierbei errechnet sich der Zeitbedarf der Sequenz im allgemeinen Falle zu

$$\tau_C(\kappa) = \Delta_1 = \frac{\sqrt{\kappa \cdot (4-\kappa)}}{2J}, \tag{9.12}$$

bzw. für den Fall des für ein SWAP-Gatter benötigten trilinearen Propagators mit $\kappa = 1$

$$\tau_C(1) = \frac{\sqrt{3}}{2J}. \tag{9.13}$$

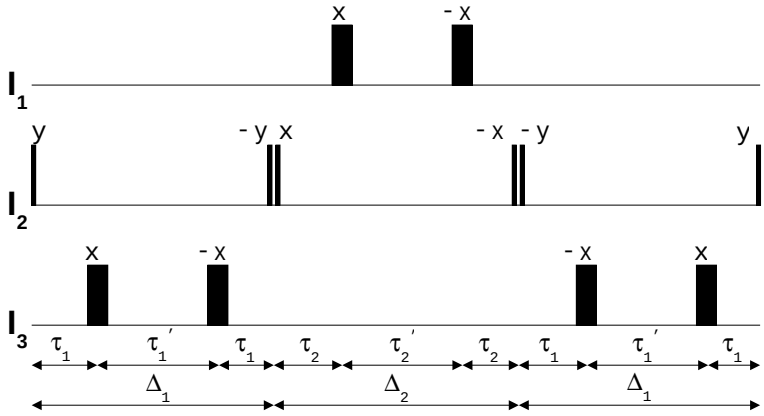

ABBILDUNG 9.2: Die konventionelle Pulssequenz, schmalbandige Implementierung des Propagators $\mathcal{U}_{zzz}$. Die angegebenen Delays sind definiert als $\tau_1 = 1/8J$, $\tau_1' = 1/4J$, $\Delta_1 = 1/2J$, $\tau_2 = \kappa/8J$, $\tau_2' = \kappa/4J$, $\Delta_2 = \kappa/2J$.

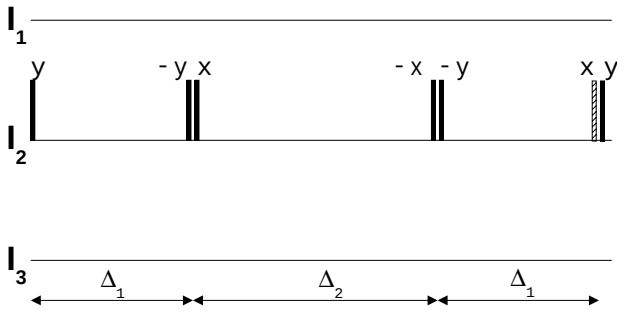

ABBILDUNG 9.3: Die verbesserte Pulssequenz, schmalbandige Implementierung des Propagators $\mathcal{U}_{zzz}$. Die angegebenen Delays sind definiert als $\Delta_1 = 1/4J$, $\Delta_2 = \kappa/2J$. Der Winkel des schraffiert dargestellten Pulses ist $\pi\kappa/2$.

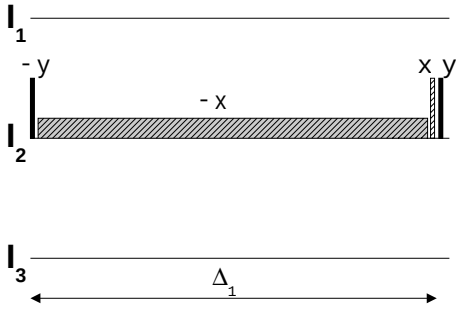

ABBILDUNG 9.4: Die zeitoptimale Pulssequenz, schmalbandige Implementierung des Propagators $\mathcal{U}_{zzz}$. Das angegebene Delay ist $\Delta_1 = \sqrt{\kappa(4-\kappa)}/2J$). Die Winkel der grauen und weißen schraffierten Pulse sind $\pi\,(2-\kappa)$ bzw. $\pi\,(2-\kappa/2)$. Die RF-Amplitude des grauen schraffierten Pulses ist $(2-\kappa)J/\sqrt{\kappa(4-\kappa)}$.

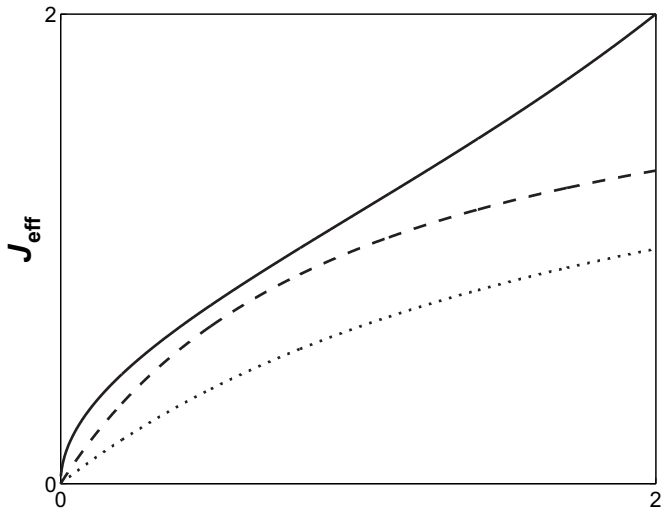

ABBILDUNG 9.5: In Abhängigkeit von κ erzeugen die Pulssequenzen bestimmte effektive Kopplung J^{eff}_{123}. Diese ergeben sich unter Verwendung von Gleichung (9.6) und den Gleichungen (9.8, 9.10, 9.12). Gezeigt sind die effektive Kopplungskonstanten für angenommene Einheitskopplungen $J_{12} = J_{23} = 1$ Hz, die von der konventionellen Pulssequenz (gepunktet), der verbesserten Pulssequenz (gestrichelt) und der zeitoptimalen Pulssequenz (durchgezogen) erzeugt werden.

Dies entspricht einer Reduktion der Dauer auf 57,7% der konventionellen bzw. auf 86,6% der verbesserten Pulssequenz.

Nachdem nun der Zeitbedarf der einzelnen Sequenzen bekannt ist, kann man die in Gleichung (9.4) angeführte effektive Kopplungskonstante für einen Bereich von κ unter Verwendung der Gleichungen (9.8, 9.10, 9.12) berechnen. In Abbildung 9.5 sind die entsprechenden Kurven gezeigt. Deutlich erkennt man die von der konventionellen über die verbesserte hin zur zeitoptimalen Sequenz ansteigende Skalierung der Kopplungskonstante.

Mit diesen Bausteinen kann man nun den für ein indirektes SWAP(1,3)-Gatter benötigten Gesamt-Propagator

$$\mathcal{U}_{\text{tot}} = \mathcal{U}_{xzx}\, \mathcal{U}_{yzy}\, \mathcal{U}_{zzz}\ \exp(\imath\frac{\pi}{2} I_{2z}). \tag{9.14}$$

erzeugen. Hierbei fällt auf, daß die Sub-Propagatoren stets vom Typus $\mathcal{U}_{\alpha z\alpha}$ sind, d.h. nur die Phasen der $I_{1,3}$-Kerne müssen durch Rotationen der entsprechenden Referenz-Koordinatensysteme angepaßt werden. In Abbildung 9.6 ist das prinzipielle Vorgehen gezeigt.

Durch diese Art der Verkettung der in den Abildungen 9.2-9.4 gezeig-

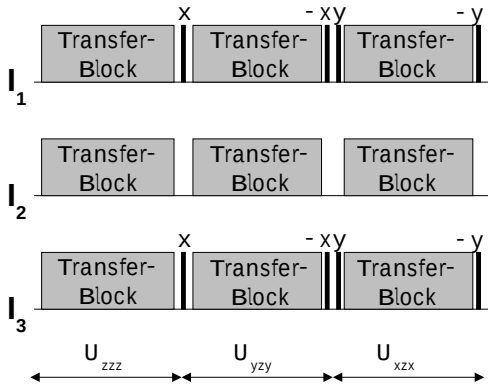

ABBILDUNG 9.6: Allgemeine Vorgehensweise, um individuelle Pulssequenz-Elemente, die trilineare Propagatoren $\mathcal{U}_{zzz}$ erzeugen, durch Rotationen der entsprechenden $I_{1,3}$-Referenzsysteme in eine Sequenz trilinearer Propagatoren $\mathcal{U}_{\alpha z\alpha}$ mit $\alpha = \{z, y, x\}$ zu überführen. Diese Abfolge dreier trilinearer Propagatoren ist für die Erzeugung eines indirekten SWAP(1,3)-Gatters gemäß Gleichung (9.14) nötig.

ten Bausteine entstehen notwendigerweise zusammengesetzte Pulse an den Übergängen von einem Baustein zum nächsten. Man kann nun diese entstehenden Komposit-Pulse durch einen Puls neuer Phase und eine z-Rotation ersetzen. Diese z-Rotationen um den Winkel ϕ können nun durch die Pulssequenz geschoben werden, indem man die Phasen der "überholten" Pulse um $-\phi$ korigiert. Schließlich kann man diese nun am Ende der Sequenz stehenden z-Rotationen zu einer einzigen zusammenfassen und entweder die Receiver-Phase so anpassen, daß diese Rotation kompensiert wird, oder in einen zusammengesetzten Puls zurückverwandeln. Wendet man diese Vorgehensweise auf die drei bisher besprochenen Bausteine an, so erreicht man eine wesentlich geringere Anzahl an notwendigen Pulsen und kann so die experimentellen Fehler reduzieren.

Nun hat man eine Pulssequenz zur Verfügung, die den für die SWAP(1,3)-Operation benötigten Propagator (9.14) erzeugt. Da jedoch bisher während der freien Entwicklungsphasen auf eine Refokussierung der chemischen Verschiebungen verzichtet wurde, besitzen die Experimente ein sehr schlechtes Verhalten für Kerne, deren Resonanz-Frequenzen von der Trägerfrequenz abweichen. Um nun breitbandige Varianten der Pulssequenzen zu erhalten, kann man in den Fällen, in denen tatsächlich freie Evolutionsperioden vorhanden sind, π-Pulse zur Refokussierung einführen.

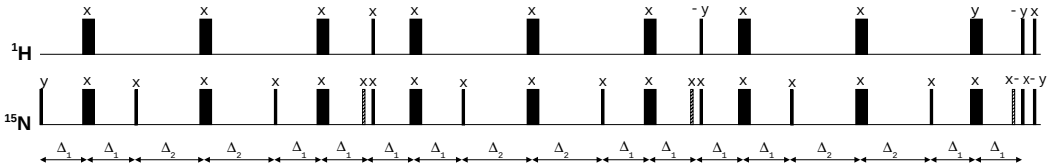

ABBILDUNG 9.7: Breitbandige Variante zur Erzeugung einer SWAP(1,3)-Operation gemäß Gleichung (9.14) auf Basis des Bausteines zur Erzeugung trilinearer Propagatoren aus Abbildung 9.3 unter Verwendung eines I_2S-Spinsystems. Die angegebenen Delays sind wie folgt definiert: $\Delta_1 = 1/8J$ und $\Delta_1 = \kappa/4J$. Der schraffierte Puls ist von κ abhängig: Für $\kappa < 1$ beträgt der Pulswinkel $(1-\kappa)\pi/2$ und für $\kappa > 1$ $(\kappa - 1)\pi/2$. Im idealen Falle des SWAP(1,3)-Gatters ist $\kappa = 1$, was einem Winkel von 0 entspricht, d.h. dieser Puls kann ausgelassen werden

Im Falle des konventionellen Pulssequenz-Bausteins (Abbildung 9.2) führt die Anwendung der eben besprochenen Vorgehensweisen zu einer recht langen Pulssequenz; daher wird hier auf eine bildliche Darstellung dieser Sequenz verzichtet. Wendet man diese Vorgehensweise auf den verbesserten konventionellen Pulssequenz-Baustein (Abbildung 9.3) an, so erhält man die in Abbildung 9.7 gezeigte Pulssequenz, die den Propagator (9.14) vollständig implementiert.

Betrachtet man den zeitoptimalen Baustein (Abbildung 9.4), so erkennt man, daß in diesem Falle keine Perioden freier Evolution vorhanden sind, d.h. diese Sequenz kann nicht ohne weiteres den üblichen Schemata zur Refokussierung der chemischen Verschiebung unterworfen werden. Darüberhinaus ist die Einstrahlleistung dieses geodätischen Pulses gering, was auch die Anregungsbandbreite der Sequenz einschränkt. Daher ist es erstrebenswert diese zeitoptimale Lösung so zu gestalten, daß sowohl die chemische Verschiebung refokussiert als auch die Einstrahlleistung erhöht werden kann. Anstatt die Kopplungsentwicklung und Rotation innerhalb eines Pulses zu kombinieren, kann man dies ebenfalls durch eine Serie kurzer harter Pulse und freier Entwicklungsperioden erreichen. In Abbildung 9.8 ist das prinzipielle Vorgehen beispielhaft gezeigt. Die Gesamtrotation um den Winkel α wird durch n einzelne Rotationen um den Winkel α' ersetzt, so daß gilt

$$\alpha' = \frac{\alpha}{N \cdot n}, \tag{9.15}$$

wobei N der Anzahl Pulse pro Basiszyklus und n der Anzahl Basiszyklen entspricht. Für $\Delta_1/(N \cdot n) \ll 1/J$ ist die Entwicklung des Spinsystems durch die

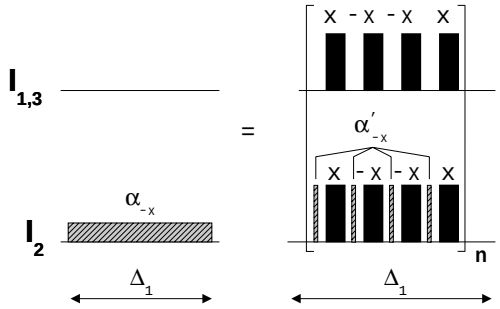

ABBILDUNG 9.8: Expansion des geodätischen Pulses durch eine Folge harter Pulse. Der Grundzyklus der Expansion bestimmt sich durch den Phasenzyklus der in die entstehenden freien Entwicklungsperioden zur Refokussierung der chemischen Verschiebung eingefügten π-Pulse. In dem gezeigten Falle wird ein MLEV-4-Zyklus verwendet, andere sind aber ebenfalls möglich. Soll der geodätische Puls eine Gesamtrotation um den Winkel α implementieren, so errechnet sich der Winkel der harten Pulse zu $\alpha' = \alpha/(N \cdot n)$, wobei N derAnzahl Pulse des Basiszyklus entspricht und n der Anzahl Wiederholung dieses Basiszyklus.

in Abbildung 9.8 gezeigten Sequenzen identisch, falls die Spins *on-resonance* sind. Selbst für Offset-Frequenzen $\Delta\nu > J$ stellt die Expansion eine sehr gute Näherung für den geodätischen Puls dar, falls $\Delta_1/(N \cdot n) \ll 1/\Delta\nu$. Sind diese Bedingungen erfüllt, so kann man die ursprüngliche Entwicklungsperiode durch $N \cdot n$ Perioden der Länge

$$\Delta_1' = \frac{\Delta_1}{N \cdot n} \tag{9.16}$$

ersetzen, d.h. man nähert den ursprünglich für den geodätischen Puls angenommenen Propagator

$$\mathcal{U}_\alpha = \exp(\imath I_{2x}\alpha - \imath\Delta_1\mathcal{H}_0) \tag{9.17}$$

durch eine Produktsumme von Propagatoren der paarweisen freien Evolution und Rotation gemäß

$$\mathcal{U}_{\alpha'} = \prod_{1}^{n+N} \exp(\imath I_{2x}\alpha') \cdot \exp(-\imath\Delta_1'\mathcal{H}_0). \tag{9.18}$$

Durch dieses Vorgehen erreicht man die Verwendung harter Pulse mit hoher Anregungsbandbreite und die Entstehung freier Evolutionsperioden, die nun wiederum dazu genutzt werden können, die Entwicklung der chemischen Verschiebung durch Einführung von π-Pulsen zu refokussieren. Hierbei können

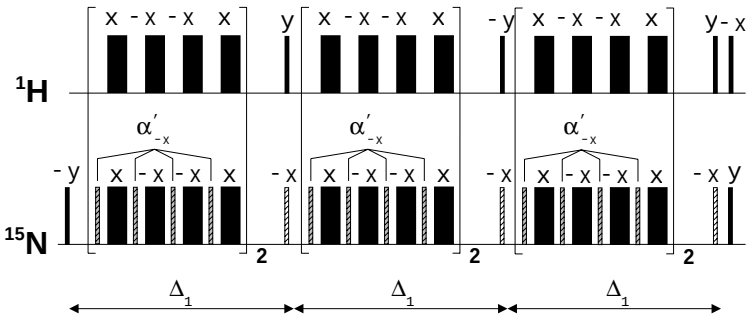

ABBILDUNG 9.9: Breitbandige Variante zur Erzeugung einer SWAP(1,3)-Operation gemäß Gleichung (9.14) auf Basis des Bausteines zur Erzeugung trilinearer Propagatoren aus Abbildung 9.4 unter Verwendung eines I_2S-Spinsystems. Hierbei wurde der geodätischen Puls durch zwei des in Abbildung 9.8 gezeigten Zyklus ersetzt. Der Pulswinkel ergibt sich zu $\alpha' = \alpha/8 = (2-\kappa)\pi/8$, die freien Entwicklungsperioden entsprechen $\Delta_1' = \Delta_1/8$.

nun verschiedene Phasenzyklen verwendet werden, die experimentelle Imperfektionen reduzieren, z.B. MLEV-N-Zyklen mit $N = 2^{i+1}$.

In Abbildung 9.9 ist eine mögliche Pulssequenz gezeigt, die die Bedingungen für diese Näherung erfüllt. Hierbei werden zwei Basiszyklen (d.h. $N = 2$) mit $n = 4$ verwendet, wobei die Phasen der zur Refokussierung verwendeten π-Pulse einem MLEV-4-Zyklus entsprechen. Somit ergibt sich der Pulswinkel zu

$$\alpha' = \frac{\alpha}{2 \cdot 4} = \frac{(2\pi - \theta/2)}{8} = \frac{(2-\kappa)\,\pi}{8}. \tag{9.19}$$

Die freien Entwicklungsperioden zwischen diesen Pulsen betragen entsprechend $\Delta_1' = \Delta_1/8$.

9.2 Experimente

Um das von der Theorie vorausgesetzte linear gekoppelte Spinsystem experimentell zu realisieren, wurde die rotationsgehinderte Amidgruppe von [^{15}N]-Acetamid (vergl. Abbildung 9.10) mit einem Anreicherungsgrad von 95% ^{15}N verwendet. In Tabelle 9.1 sind die wichtigsten NMR-Daten dieses Systems zusammengefaßt. Wie man erkennen kann, sind die $^1J_{N,H}$-Kopplungen sowohl hinreichend ähnlich als auch groß genug im Vergleich zu den restlichen Kopplungen, um eine lineare Kopplungstopologie darzustellen.

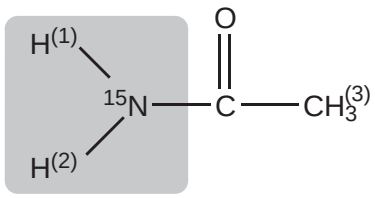

Abbildung 9.10: Als Modellsystem wird in den Experimenten die rotationsgehinderte Amidgruppe von [^{15}N-95%]-Acetamid verwendet. Die innerhalb dieses Kapitels verwendete Nummerierung der Protonen ist in Klammern angegeben.

Spin	N	H^1	H^2	H^3
ν [Hz]	6705,08	4071,31	4429,15	-
$J_{N,}$ [Hz]	-	88,79	87,26	1,2
$J_{H^1,}$ [Hz]	88.79	-	2.9	0,7
$J_{H^2,}$ [Hz]	87,26	2,9	-	0,0
$J_{H^3,}$ [Hz]	1,2	0,7	0,0	-

Tabelle 9.1: Relevante NMR-Daten von [^{15}N]-Acetamid. Die $^1J_{N,H}$-Kopplungen sind ähnlich und groß genug gegenüber den restlichen Kopplungen, um die von der Theorie geforderte lineare Kopplungstopologie zu erfüllen.

direkt	Mischzeit (th) [ms]	Mischzeit (exp) [ms]
konventionell	517,23	399,41
DIPSI	172,41	152,52
indirekt	Mischzeit (th) [ms]	Mischzeit (exp) [ms]
konventionell	51,12	50,88
verbessert	34,08	33,06
geodätisch	29,52	27,99

Tabelle 9.2: Benötigte Mischzeiten einer SWAP(1,3)-Operation durch verschiedene Pulssequenzen. Die Werte beziehen sich auf den jeweilig zu erreichenden maximalen Transfer. Die experimentellen Mischzeiten (exp) liegen unterhalb der theoretischen (th), weil sich das Maximum der Transferkurven durch Relaxationseinflüsse zu kürzeren Mischzeiten verschiebt. Diese Verschiebung ist umso größer, je länger die Mischzeit der entsprechenden Sequenz ist.

Da das experimentelle System noch eine direkte $^1J_{H,H}$-Kopplung aufweist, besteht die Möglichkeit diese zur Implementation eines direkten SWAP(1,3)-Operation auszunutzen. Hierzu wurde sowohl homonukleares isotropes Mischen (DIPSI) [71] als auch die in [68] vorgeschlagene Sequenz (konventionelles direktes SWAP(1,2)-Gatter) verwendet. Da die direkte Kopplung jedoch sehr klein ist, ergibt sich für beide Sequenzen eine sehr lange Mischzeit. Im Falle des Acetamids sind die theoretisch idealen Mischzeiten in Bereichen, in denen Relaxation bereits starke Einflüsse zeigt. Dies kann man daran erkennen, daß die in Tabelle 9.2 angegebenen Zeiten für das Erreichen des Maximums der experimentellen Aufbaukaurven erheblich kürzer sind, als die theoretisch (ohne Berücksichtigung von Relaxationseinflüssen) für den maximalen Transfer benötigten.

Aus Gründen der besseren Vergleichbarkeit werden die Pulssequenzen durch den Transfer $[\sigma(0) = I_x(H^1)] \longrightarrow [\sigma(t_{mix}) = I_x(H^2)]$ charakterisiert. Die Anwendung eines SWAP-Gatters vertauscht nun detektierbare Magnetisierung des Spins H^1 mit undetektierbarer des Spins H^2, wodurch die Effizienz durch Intergration des Antwortsignals quantifiziert werden kann. Normiert man dieses Intergral auf das der direkten Anregung, so liegt die Transferamplitude in $\eta \in [0, 1]$, wodurch die Ergebnisse direkt mit den entsprechenden Simulationen für das dieser Transferamplitude entsprechende Skalarprodukt

$$\eta = \langle \sigma(\tau_p) | I_{1x} \rangle \tag{9.20}$$

vergleichbar sind.

9.2.1 Direkte und schmalbandige indirekte SWAP-Gatter

In Abbildung 9.11 sind die Simulationsergebnisse der schmalbandigen Pulssequenzen gezeigt. Die Simulationen wurden unter der Annahme des Spinsystems der Amidgruppe für ein statisches Magnetfeld von 14,09 T ($\nu^H_{\text{Lamor}} =$ 600 MHz) durchgeführt. In der Abbildung ist die Abhängigkeit der Sequenzen vom Parameter κ gezeigt. Alle Sequenzen sollten die SWAP-Operation für $\kappa = 1$ implementieren, was durch die Simulationen bestätigt wird. Die Schwankungen der Transferamplituden für $\kappa \neq 1$ unterstreichen recht deut-

lich, daß die schmalbandigen Sequenzen ausschließlich für ein ideales 3-Spinsystem hergeleitet wurden.

In Abbildung 9.12 werden zusammenfassend die numerischen (Abbildung 9.12.A) und experimentellen Daten (Abbildung 9.12.B) der bisher besprochenen Sequenzen (direkte und schmalbandige indirekte SWAP-Gatter) verglichen. Man erkennt eine gute Übereinstimung der experimentellen mit den numerischen Daten. Einzig der nicht in der Simulation berücksichtigte Effekt der Relaxation zeigt für die direkten SWAP-Operationen einen starken Einfluß im Experiment, was allerdings durch die sehr langen Dauern der Operationen verständlich ist. Dieser Einfluß läßt ebenfalls die im Experiment zu erreichende maximale Transferamplitude stark sinken. Die repräsentativ für die indirekten SWAP-Gatter gezeigte Transferamplitude der schmalbandingen verbesserten Sequenz zeigt diesen Effekt nicht und erfüllt den durch numerische Simulation vorhergesagten Kurvenverlauf sehr gut.

Um die Schmalbandigkeit zu illustrieren, wurde die schmalbandige geodätische Pulssequenz in Abhängigkeit der Offset-Frequenzen vermessen. In Abbildung 9.13 sind die experimentellen Transferamplituden der schmalbandigen und der breitbandigen Variante gegenübergestellt. Es werden Isolinien $\eta\left(\nu_{\text{off}}\right)$ für Variationen der Protonen-Frequenz bei konstanter Stickstoff-Frequenz (schmalbandige Variante: Tafel 9.13.A; breitbandige Variante: Tafel 9.13.B) und der Stickstoff-Frequenz bei konstanter Protonen-Frequenz (schmalbandige Variante: Tafel 9.13.C; breitbandige Variante: Tafel 9.13.D) gezeigt. Hierbei ist die jeweils konstante Frequenz so gewählt, daß die entsprechende Kernfrequenz on-resonance ist. Man kann der Abbildung entnehmen, daß die schmalbandige Variante für den idealen Fall die SWAP-Operation korrekt implementiert, jedoch sehr empfindlich auf geringe Änderungen der Offset-Frequenzen reagiert. Die breitbandige Variante erweist sich als wesentlich robuster gegenüber diesen Einflüssen.

9.2.2 Breitbandige indirekte SWAP-Gatter

In Abbildung 9.14 sind die Transferamplituden der breitbandigen Varianten der indirekten SWAP-Gatter in Abhängigkeit von κ dargestellt. Durch Variation von κ erhält man nach Gleichungen (9.8-9.12) eine Zeitabhängigkeit, welche als Abszisse dargestellt ist.

Die in Abbildung 9.14.A gezeigten Ergebnisse beziehen sich auf eine Simu-

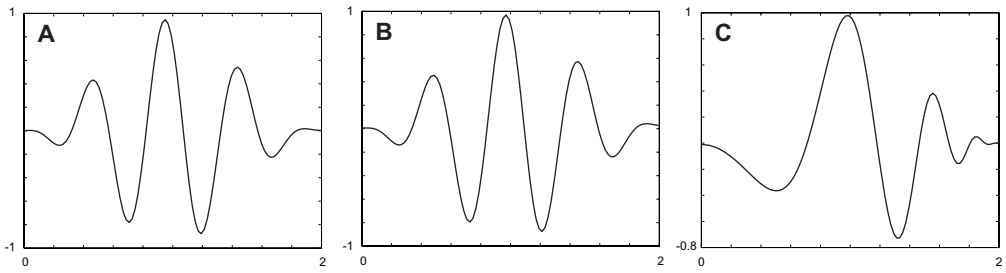

ABBILDUNG 9.11: Numerische Simulation der Transferamplitude η der schmalbandigen Sequenzen in Abhängigkeit des Parameters κ. (A) konventionelle indirekte Sequenz. (B) verbesserte indirkete Sequenz. (C) zeitoptimale indirekte Sequenz.

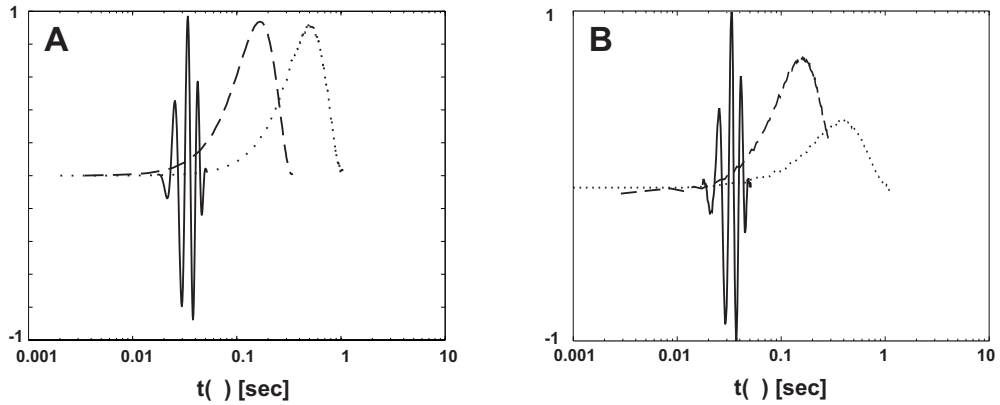

ABBILDUNG 9.12: Transferamplituden $\eta(\tau_p)$ der schmalbandigen verbesserten indirekten Sequenz (durchgezogen), DIPSI (gestrichelt) und der konventionellen direkten Sequenz (gepunktet).(A) Numerische Simulation. (B) Experimentelle Daten.

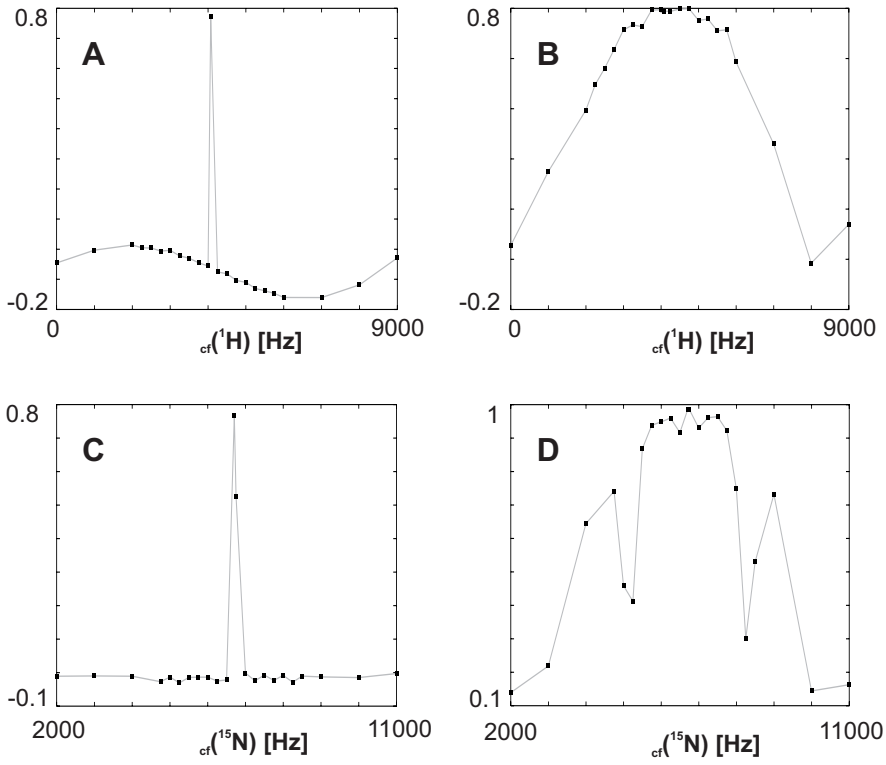

ABBILDUNG 9.13: Vergleich der Offset-Abhängigkeit der experimentellen Transferamplitude der geodätischen Sequenz. Die linken Teilabbildungen entsprechen Isolinien der Offset-Abhängigkeit der schmalbandigen Sequenz für den ^{1}H-Kanal (A) und den ^{15}N-Kanal (C), während die rechten Teilabbildungen Isolinien der Offset-Abhängigkeit der breitbandigen Sequenz für den ^{1}H-Kanal (B) und den ^{15}N-Kanal (D) entsprechen.

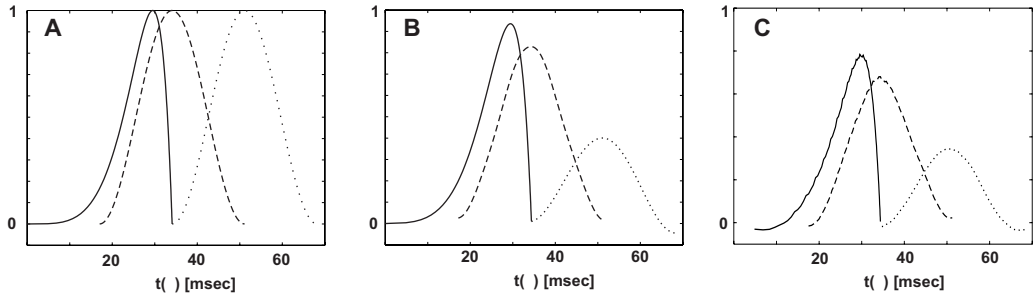

ABBILDUNG 9.14: $t(\kappa)$-Abhängigkeit der Transferamplitude η für die breitbandigen Varianten der konventionellen Sequenz (gepunktet), der verbeseerten Sequenz (gestrichelt) und der zeitoptimalen Sequenz (durchgezogen). (A) Simulation unter der Annahme idealer Bedingungen (3-Spinsystem H^1-$^{[15]}N$-H^2 mit linearer Kopplungstopolgie, alle Spins sind *on-resonance*. Die Pulse haben keine zeitliche Dauer und besitzten keine B_1-Inhomogenität). (B) Simulation unter Annahme realer Bedingungen (4-Spinsystem $\{H^1$-$^{[15]}N$-$H^2\}$- H^3 mit realer Kopplungstopologie, die Offset-Frequenzen der Spins wurden berücksichtigt (vergl. Tabelle 9.1). Die Pulse wurden mit ihren respektiven Dauern (7μs für den ^{1}H-Kanal bzw. 45,5μs für den ^{15}N-Kanal, bestimmt für eine $\pi/2$-Rotation) und einer Fehlerverteilung zur Modellierung der B_1-Inhomogenität (Gauss-Verteilung mit 10% Halbwertsbreite) versehen). (C) Experimentelle Daten.

lation der Pulssequenzen unter idealen Bedingungen. Da hierbei alle von der Theorie geforderten Bedingungen erfüllt sind und weiterhin mögliche Störeffekte außer Acht gelassen werden, verhalten sich alle Sequenzen gleich gut. Sie implementieren für $\kappa = 1$ die SWAP-Operation vollständig, für $\kappa \neq 1$ sinkt die Transferamplitude. Dies zeigt, daß die in die Pulssequenzen eingeführten Modifikationen den geforderten Propagator nicht verändern.

In Abbildung 9.14.B sind wiederum Ergebnisse numerischer Simulationen gezeigt. Hierbei wurden allerdings realistische Bedingungen angenommen. Das Spinsystem wurde um das störende Methyl-Proton $H^{(3)}$ erweitert und die Kopplungstopologie sowie die Kernfrequenzen dem realen Spinsystem angeglichen (vergl. Tabelle 9.1). Neben dem realistischen Spinsystem wurden auch realistische RF-Pulse angenommen. Zum einen wurden die im Experiment ermittelten Einstrahldauern berücksichtigt ($7\mu s$ für einen $\pi/2$-Puls des Protonen-Kanals; $45,5\mu s$ für einen $\pi/2$-Puls des Stickstoff-Kanals). Zum anderen wurde die Inhomogenität der erzeugten RF-Felder durch eine synthetische Fehlerverteilung simuliert (Fehlerverteilung nach Gauß mit 10% Halbwertsbreite). Dieser Einschluß realistischer Effekte zeigt bereits in der Si-

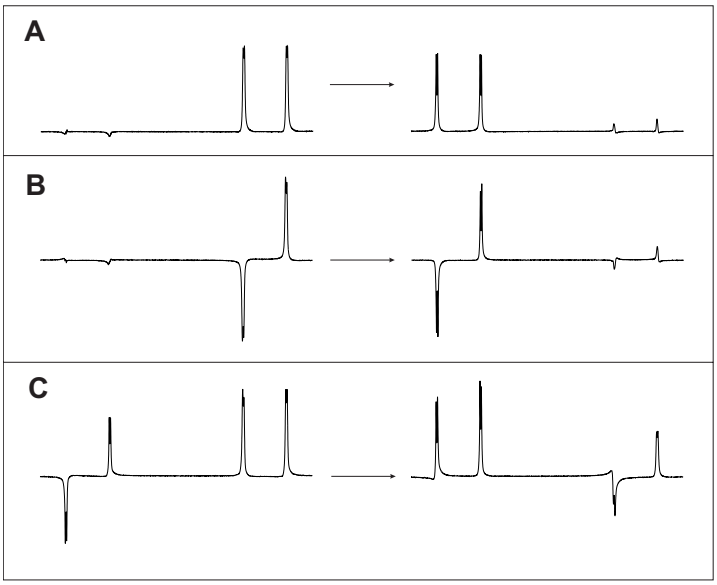

ABBILDUNG 9.15: Beispiele von SWAP(1,3)-Operationen. (A) $I_{1x} \rightarrow I_{2x}$, (B) $2\ I_{1x}\ S_z \rightarrow 2\ I_{2x}\ S_z$, and (C) $I_{1x} + 2\ I_{2x}\ S_z \rightarrow I_{2x} + 2\ I_{1x}\ S_z$

mulation einen merklichen Einfluß auf die erreichten Transferamplituden. Die maximal erreichte Effizienz sinkt für alle Sequenzen, wobei die Einflüsse auf die Sequenzen unterschiedlich sind. Die beste Robustheit zeigt die breitbandige geodätische Sequenz, während die beiden konventionellen Pulssequenzen stärker von den realistischen Simulationsbedingungen beeinflußt werden. Die SWAP-Operationen werden für $\kappa = 1$ implementiert, die Einflüsse der relativen Offset-Frequenzen der einzelnen Spins sind jedoch nicht mehr vorhanden, was durch das Fehlen der in Abbildung 9.11 noch vorhandenen Modulationen zum Ausdruck kommt.

In Abbildung 9.14.C sind die exerimentell ermittelten Transferamplituden aufgetragen. Die Übereinstimmung mit den numerischen Transferamplituden ist gut, man kann lediglich eine Dämpfung der Kurven festellen. Dies ist durch Relaxationseinflüsse zu erklären, die in den Simulationen nicht berücksichtigt wurden.

In Abbildung 9.15 sind drei Beispiele für SWAP(1,3)-Operationen gezeigt. Hierbei steigt die Komplexität der verwendeten Spinterme mit jedem Beispiel. Im ersten Beispiel (vergl. Abbildung 9.15.A) wird die SWAP-Operation auf einen Zustand des Systems angewendet, in dem nur ein Spin angeregt wurde. In diesem Fall wurde x-Magnetisierung auf dem Proton H^1 präpariert,

d.h. $\sigma(0) = I_{1x}$. Nach Anwendung des SWAP-Gatters haben die Protonen ihren Zustand wie erwartet ausgetauscht, d.h. der Zustand des Systems entspricht $\sigma(\tau_p) = I_{2x}$. Im zweiten Beispiel (vergl. Abbildung 9.15.B) entspricht der Präparationszustand einem Antiphase-Term des Protons H^1 bezüglich des Hetero-Kerns, d.h. $\sigma(0) = 2\ I_{1x}S_z$. Wie erwartet entsteht nach Anwendung der SWAP-Operation der entsprechende Antiphase-Term $\sigma(\tau_p) = 2\ I_{2x}S_z$, d.h. die Protonen tauschen ihre Zustände aus, ohne den Stickstoff-Kern zu beeinflussen. Das dritte Beispiel (vergl. Abbildung 9.15.C) entspricht einer Kombination aus den beiden vorherigen Beispielen. Der präparierte Zustand besteht aus einem transversalen Term des Protons H^1 und einem Antiphase-Term des Protons H^2, d.h. $\sigma(0) = I_{1x} + 2\ I_{2x}S_z$. Nach erfolgter SWAP-Operation zeigt das Spektrum die erwarteten Terme $\sigma(\tau_p) = I_{2x} + 2\ I_{1x}S_z$. Da hierbei allerdings beide Protonen beteiligt sind und die Sequenzen im realen Fall Transferamplituden $\eta < 1$ aufweisen (s.o.), überlagern sich die Endzustände mit den verbleibenden Fragmenten der Anregungszustände. Im Spektrum ist dies daran zu sehen, daß das jeweils linke Teilsignal der beiden Dubletts kleiner erscheint, weil dort im Präparationszustand jeweils ein Teilsignal mit einem negativen relativen Vorzeichen vorhanden war.

In Abbildung 9.16 sind die Transferamplituden der verbesserten (Simulation: Abbildung 9.16.A; Experiment: Abbildung 9.16.B) und zeitoptimalen Sequenz (Simulation: Abbildung 9.16.C; Experiment: Abbildung 9.16.D) in Abhängigkeit der Offset-Frequenzen gezeigt. Die Simulationen wurden unter der Annahme realer Bedingungen (s.o.) für eine (11×11)-Matrix von Offset-Frequenzen durchgeführt und ließen eine Bandbreite von $\pm$5kHz erwarten. Unter Bandbreite ist in diesem Fall der Bereich zu verstehen, in dem die Transferamplitude $\eta \geq 0.5$ ist. Die experimentellen Transferamplituden wurden für die gleiche Matrix von Offset-Frequenzen bestimmt. Man sieht eine qualitativ gute Übereinstimmung zwischen den numerischen und experimentellen Daten.

9.3 Schlußfolgerung

Durch die Anwendung der geometrischen Optimalen-Kontroll-Theorie konnte eine zeitoptimale Lösung zur Erzeugung einer SWAP(1,3)-Operation angegeben werden [62]. Diese geodätische Pulssequenz konnte direkt in einem

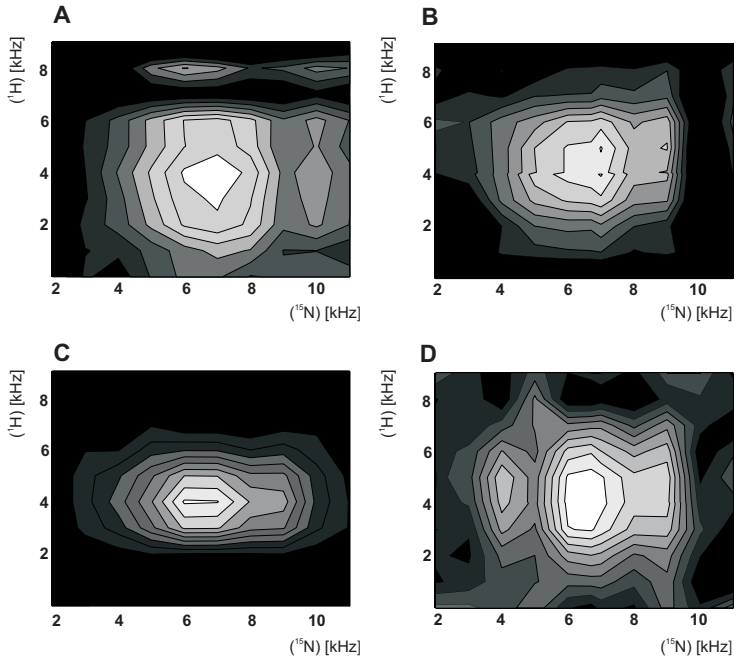

Abbildung 9.16: Offset-Abhängigkeit der breitbandigen Implementierungen. Die Konturlinien entsprechen Transferamplituden $\eta = 0,0$ (schwarz); 0,1; 0,2; ...; 1,0 (weiß). (A) verbeserte Pulssequenz, Simulation. (B) verbesserte Pulssequenz, experimentelle Daten. (C) zeitoptimale Sequenz, Simulation. (D) zeitoptimale Sequenz, experimentelle Daten.

Spinsystem implementiert werden. Durch einen Vergleich mit konventionellen Implementierungen konnte experimentell nachvollzogen werden, daß die zeitoptimale Lösung 57,7% der Dauer eines konventionellen indirekten SWAP-Gatters bzw. 5,7% der einer konventionellen direkten SWAP-Operation benötigt.

Neben den experimentellen Untersuchungen der theoretisch abgeleiteten Pulssequenzen konnte darüberhinaus gezeigt werden, daß sich diese Sequenzen durch Anwendung von NMR-spektroskopischen Überlegungen soweit modifizieren lassen, daß sich sowohl die Anregungsbandbreite als auch die Robustheit gegenüber variierenden Offset-Frequenzen erheblich verbessern lassen. Diese modifizierte Sequenz wurde sowohl durch numerische Simulationen als auch experimentell charaktersiert. Durch die Modifikationen an der theoretisch abgeleiteten Pulssequenz konnte ein zeitoptimales indirektes SWAP(1,3)-Gatter realisiert werden, welches allgemein anwendbar ist.

Kapitel 10

Effizienter Kohärenztransfer in 5-Spinketten

Der Transfer von Kohärenz entlang einer Kette von Spins ist ein in der (biomolekularen) NMR Spektroskopie häufig auftretender Schritt. Daher ist es von Interesse, diese Problematik im Rahmen der Steuerungstheorie zu untersuchen. Bereits in Kapitel 5 sind numerische Betrachtung bezüglich des Transfers von Inphase-Kohärenz entlang Spinketten I_n für $n \in [2, 5]$ vorgestellt worden. Diese Ergebnisse konnten in [72] verallgemeinert und durch eine analytische Lösung beschrieben werden.

In diesem Kapitel werden die ersten experimentellen Ergebnisse vorgestellt, bei denen Pulssequenzen verwendet wurden, die dieser analytischen Lösung entsprechen.

10.1 Theorie

Vorausgesetzt wird ein Spinsystem, das dem 1-dimensionalen Ising-Modell [73] entspricht. D.h. es wird ein System betrachtet, welches eine lineare Kette von Spin-1/2-Teilchen enthält. Darüberhinaus sind die einzelnen Spins nur mit ihren direkten Nachbarn gekoppelt. In der NMR-Spektroskopie erfüllen Spinsysteme diese Voraussetzung, die aus N Spin-1/2-Teilchen mit einem Kopplungs-Operator

$$\mathcal{H}_c = 2\pi \sum_{k=2}^{N} J_{k-1,k}\ I_{(k-1),z}\ I_{k,z} \tag{10.1}$$

aufgebaut sind. Im Folgenden wird angenommen, daß das Spinsystem über eine einheitliche Kopplungskonstante J verfügt. Weiterhin sollen die Resonanz-

frequenzen der einzelnen Kernspins so große Differenzen aufweisen, daß sie als selektiv bepulsbar anzusehen sind.

Betrachtet man nun einen solche Kette von Kernspins, stellt man fest, daß der Transfer entlang dieser im Wesentlichen von zwei Faktoren beeinflußt wird. Zum einen muß Magnetisierung innerhalb eines einzelnen Atomkerns durch Rotationen

$$\mathcal{U}_{\mathrm{RF}} = \exp\left(-\imath\tau_p\mathcal{H}_{\mathrm{RF}}\right) \tag{10.2}$$

transformiert werden, zum anderen kann Kohärenz nur unter Ausnutzung der Kopplung von einem zum nächsten Kernspin transferiert werden:

$$\mathcal{U}_c = \exp\left(-\imath\tau_m\mathcal{H}_{\mathrm{J}}\right) \tag{10.3}$$

Hierbei gilt in der Regel

$$\nu_{\mathrm{RF}} \gg J. \tag{10.4}$$

Daraus folgt, daß der Zeitbedarf der Transformation durch RF-Pulse gegenüber der Kopplungs-Entwicklung vernachlässigbar ist, d.h. die zum Kohärenztransfer nötige Mischzeit wird durch die Dauer der freien Entwicklungsperioden bestimmt. Diese ist zudem von den zu übertragenden Spintermen abhängig. Soll ein ein 1-Spinzustand, z.B. $\mathcal{A} = I_1^-$, über eine Kopplung übertragen werden, so ergibt sich die minimale Transferzeit zu

$$\min\left\{\tau_c\left(\mathcal{A}_k \rightarrow \mathcal{A}_{k+1}\right)\right\} = 3/2J. \tag{10.5}$$

Sollen hingegen 2-Spinterme wie $\mathcal{B} = 2I_{1x}I_{2z}$ über ein Kopplung transferiert werden, so ergibt sich das Minimum zu

$$\min\left\{\tau_c\left(\mathcal{B}_{k,k+1} \rightarrow \mathcal{B}_{k+1,k+2}\right)\right\} = 1/2J. \tag{10.6}$$

Um nun beliebige Kohärenzen entlang einer Kette mit maximaler Geschwindigkeit zu transferieren, wurde in [72] daher vorgeschlagen, einen sog. Soliton-Operator Λ_k^- zu erzeugen, der mit einer Rate von $2J$ durch die Spinkette bewegt werden kann. Damit der Vorteil der schnellen Propagation nicht durch eine langwierige Erzeugungsphase zunichte gemacht wird, soll dieser Operator zudem möglichst lokalisiert sein. Dies führt zu der allgemeinen Formulierung der zeitoptimalen Pulsequenz gemäß

$$\underbrace{I_1^- \rightarrow \Lambda_1^-}_{\text{Erzeugung}} \underbrace{\rightarrow \Lambda_2^- \rightarrow \cdots \rightarrow \Lambda_{k-1}^- \rightarrow}_{\text{Propagation}} \underbrace{\Lambda_k^- \rightarrow I_k^-}_{\text{Abbau}}.$$

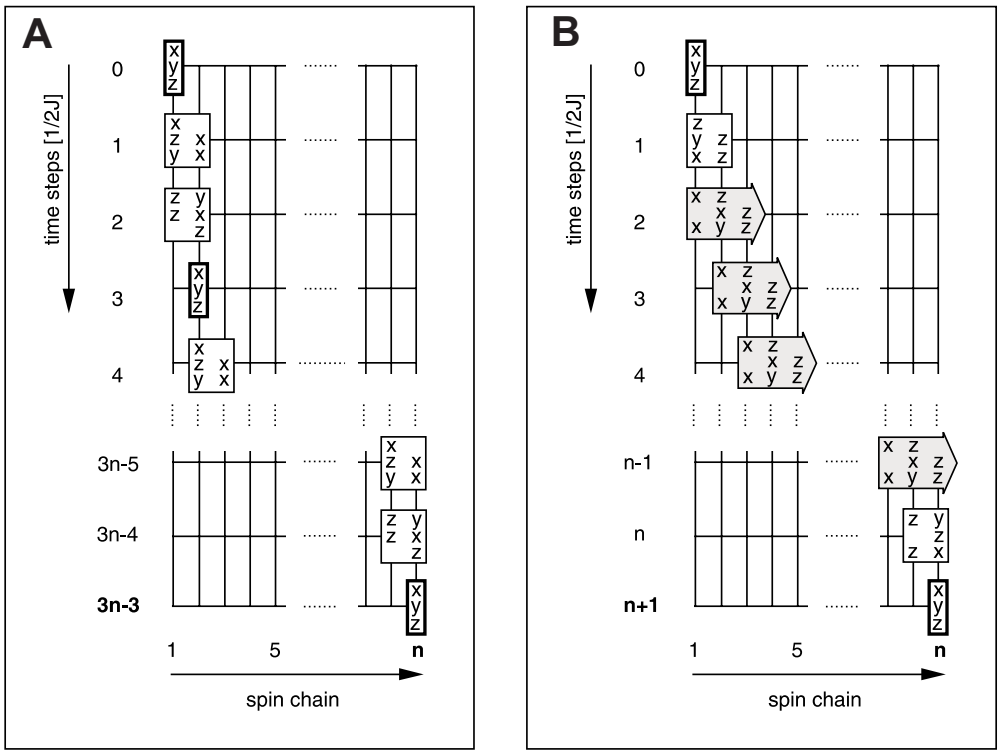

ABBILDUNG 10.1: Schematische Darstellung von Kohärenztransfer entlang einer 1-dimensionalen Ising-Spin-Kette mit N Kernspins. (A) Transfer durch sequentielle SWAP-Operationen. Diese entsprechen selektivem isotropem Mischen, was in jeweils drei Perioden $\mathcal{H}_{xx}, \mathcal{H}_{yy}$ und $\mathcal{H}_{zz}$ der Dauer $1/2J$ zerlegt werden kann. Damit ergibt sich die Dauer eines Schrittes $I_k^- \rightarrow I_{k+1}^-$ zu $3/2J$. (B) Transfer durch effektive Soliton-Operatoren. Die ersten und letzten beiden Mischperioden der Dauer $1/2J$ entsprechen dem Auf- bzw. Abbau des Soliton-Operators (dargestellt durch einen grauen Pfeil) gemäß Gleichungen (10.9, 10.10). Der Soliton-Operator selbst wird innerhalb $1/J$ gemäß Gleichung (10.8) um eine Position in Transferrichtung propagiert.

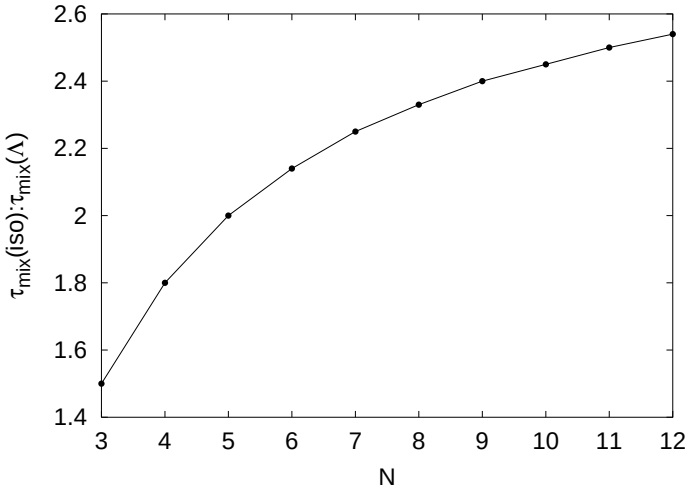

ABBILDUNG 10.2: Mischzeitverhältnis von sequentiellem isotropem Mischen und Soliton-Sequenz in Abhängigkeit von der Länge der Spinkette. Bereits für kleine Spinketten (drei bis fünf Kerne) beträgt der Gewinn der Soliton-Sequenz 50% bis 100%.

Der effektive Soliton-Operator soll die Form

$$\Lambda_k = \begin{pmatrix} \Lambda_{k,x} \\ \Lambda_{k,y} \\ \Lambda_{k,z} \end{pmatrix} = \begin{pmatrix} 2I_{k-2,x}I_{k-1,z} \\ 2I_{k-1,x}I_{k,z} \\ 4I_{k-2,x}I_{k-1,y}I_{k,z} \end{pmatrix} \tag{10.7}$$

haben, damit er durch

$$\mathcal{U}_\Lambda = \exp(-\imath\mathcal{H}_c\Delta t)\exp(-\imath(\pi/2)F_y), \tag{10.8}$$

in $\Delta t = 1/2J$ propagiert werden kann. Der Aufbau des Soliton-Operators erfolgt durch Anwendung von

$$\begin{aligned} \mathcal{U}_1 &= \exp(-\imath\mathcal{H}_c\Delta t)\exp(-\imath(\pi/2)(I_{1x}+I_{2y}))\exp(-\imath\mathcal{H}_c\Delta t) \\ &\quad \times \exp(-\imath(\pi/2)I_{1y})\exp(-\imath(\pi/2)I_{1x}). \end{aligned} \tag{10.9}$$

Hierbei ist $\Delta t = 1/2J$, d.h. die zum Aufbau des Soliton-Operators benötigte Zeit ergibt sich zu $1/J$. Der Abbau des Operators erfolgt durch Anwendung von

$$\begin{aligned} \mathcal{U}_k &= \exp(-\imath(\pi/2)I_{k,x})\exp(-\imath\mathcal{H}_c\Delta t)\exp(-\imath(\pi/2)(I_{k,x}-I_{k-1,y})) \\ &\quad \times \exp(-\imath\mathcal{H}_c\Delta t)\exp(-\imath(\pi/2)F_y), \end{aligned} \tag{10.10}$$

wobei wiederum gilt $\Delta t = 1/2J$. D.h. der Abbau benötigt dieselbe Dauer $1/J$ wie der Aufbau.

Somit ergibt sich die Gesamt-Dauer einer solchen Pulssequenz für ein N-Spinsystem zu

$$\tau_{\text{tot, }\Lambda} = \frac{1}{J} + \frac{N-3}{2J} + \frac{1}{J} = \frac{1+N}{2J}. \tag{10.11}$$

Im Vergleich dazu würde eine schrittweise Propagation durch Anwendung selektiver isotroper Mischperioden

$$I_1^- \rightarrow I_2^- \rightarrow I_3^- \rightarrow \cdots \rightarrow I_{k-1}^- \rightarrow I_k^-$$

gemäß Abbildung 10.1.A

$$\tau_{\text{tot, iso}} = (N-1) \times \frac{3}{2J} \tag{10.12}$$

benötigen.

Der maximale Gewinn des Soliton-Ansatzes ergibt sich durch

$$\lim_{N\rightarrow\infty} \left(\frac{\tau_{\text{tot, iso}}}{\tau_{\text{tot, }\Lambda}} \right) = \lim_{N\rightarrow\infty} \left(3 \cdot \frac{N-1}{N+1} \right) = 3. \tag{10.13}$$

In Abbildung 10.2 ist der Verlauf des Mischzeitverhältnisses zwischen Soliton-Sequenz und selektivem isotropem Mischen für Spinketten mit $N = [3, 12]$ Kernspins gezeigt. Man erkennt, daß bereits für kleine Spinsysteme aus drei bis fünf Kernen der zeitliche Gewinn der Soliton-Sequenz zwischen 50% und 100% beträgt.

10.2 Experimente

Als Modellsystem für eine linear gekoppelte Spinkette wurde die aliphatische Region der Aminosäure Lysin verwendet (vergl. Abbildung 10.3). Um die auf Kohlenstoff basierenden NMR-Experimente durchführen zu können, wurde eine vollständig ^{13}C-markierte Probe mit 99% Isotopenreinheit verwendet. Wie man Tabelle 10.1 entnehmen kann, werden die von der Theorie gemachten Annahmen gut erfüllt. Die Kopplungstopologie sind ähnlich genug, um die in Gleichung 10.1 angenommene einheitliche Kopplung darzustellen.

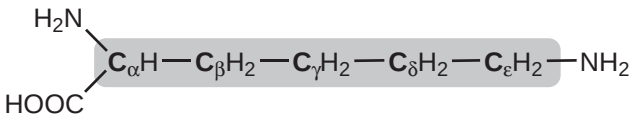

ABBILDUNG 10.3: Als Modellsystem wird die aus den fünf aliphatischen Kohlenstoffatomen gebildete Spinkette von [U-$^{13}C_6$-99%]-Lysin verwendet.

	C_α	C_β	C_γ	C_δ	C_ϵ	C(O)
ν [Hz]	8036,4	4481,6	3288,0	4024,0	5948,8	24555,1
J_{C_α}, [Hz]	-	33,8	-			55,5
J_{C_β}, [Hz]	33,8	-	32,4	-	-	-
J_{C_γ}, [Hz]	-	32,4	-	33,8	-	-
J_{C_δ}, [Hz]	-	-	33,8	-	33,8	-
J_{C_ϵ}, [Hz]	-	-	-	33,8	-	-
$J_{C(O)}$, [Hz]	55,5	-	-	-	-	-

TABELLE 10.1: Relevante NMR-Daten (DMX600) von [U$^{13}C_6$-99%]-Lysin. Die ^1J-Kopplungen sind ähnlich genug, um das von der Theorie geforderte einheitlich gekoppelte Spinsystem darzustellen.

10.2.1 Diskussion der Pulssequenzen

Um nun Spinzustände entlang dieser Spinkette zu transferieren, sind neben der Soliton-Sequenz noch weitere Experimente denkbar. Das zu entwickelnde NMR-Experiment soll neben diesen austauschbaren Transfer-Blöcken eine möglichst flexible Präparation des zu transferierende Anfangszustandes bieten. Im speziellen Fall soll die Möglichkeit bestehen, die Zustände I_x, I_y, I_z und I^- des Kernspin C_ε zu erzeugen. In Abbildung 10.4 ist zunächst dieses generelle Rahmenwerk gezeigt. Das Experiment beginnt mit einer den heteronuklearen Kern-Overhauser-Effekt (NOE) ausnutzenden Polarisationstransfer-Periode, worauf ein C_ε-selektiver und ein unselektiver $(\pi/2)_x$-Puls folgt, so daß der selektierte Kern entlang der longitudinalen Achse polarisiert ist, während die Magnetisierung der übrigen Spins transversal ist. Diese wird mit einem Gradienten dephasiert. Es folgen optionale (grau bzw. grau schraffiert dargestellte) Elemente, die je nach gewünschtem Anfangszustand benötigt werden.

- Anfangszustand I_x: Der sich vor dem Transfer-Block befindende Puls muß $\alpha_\Phi = (\pi/2)_y$ gewählt werden. Alle weiteren optionalen Elemente

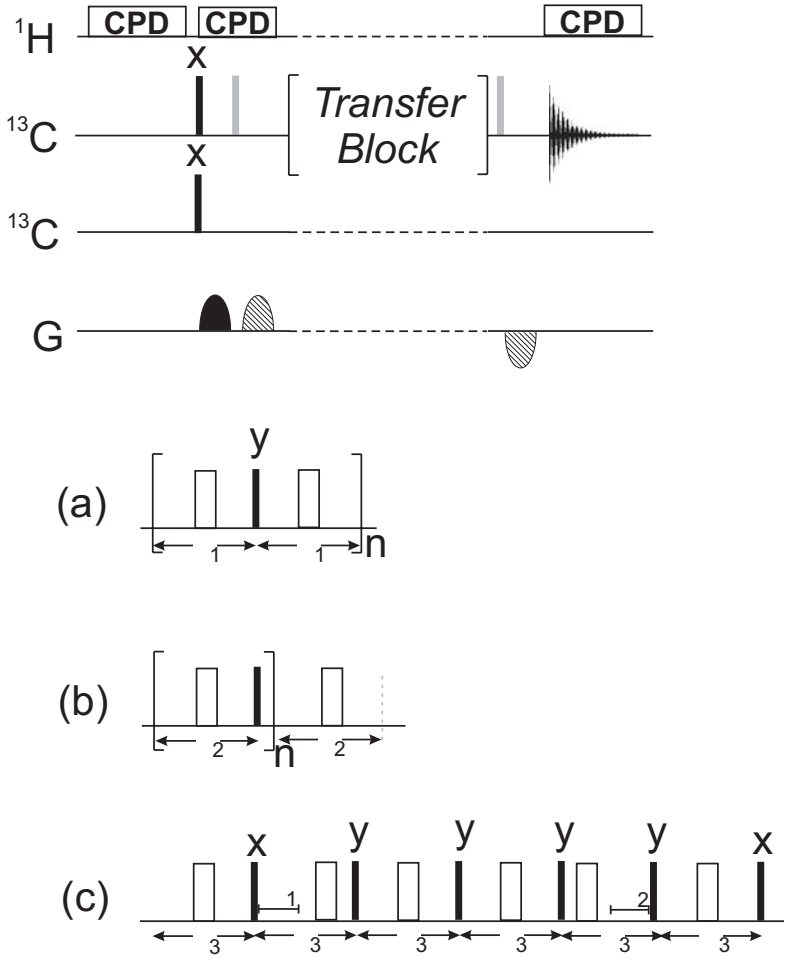

Abbildung 10.4: Pulssequenzen zum Kohärenztransfer entlang einer Spinkette. Die allgemeine Sequenz dient zur Präpation des zu übertragenden Zustandes. Je nach gefordertem Anfangszustand sind die grau (schraffiert) eingezeichneten optionalen Elemente zu verwenden (s. Text). In dieses Gerüst kann eine beliebige Sequenz als Transfer-Block verwendet werden, z.B. (a) planares Mischen [74], (b) sequentielles INEPT, (c) SOLITON.

entfallen.

- Anfangszustand I_y: Der sich vor dem Transfer-Block befindende Puls muß $\alpha_\Phi = (\pi/2)_{-x}$ gewählt werden. Alle weiteren optionalen Elemente entfallen.

- Anfangszustand I_z: Der selektierte Spin ist bereits entlang der longitudinalen Achse polarisiert, d.h. vor dem Transfer-Block werden keine weiteren Elemente benötigt. Lediglich um den transferierten Zustand detektieren zu können, muß der Puls $\beta_\Phi = (\pi/2)_y$ gewählt werden.

- Anfangszustand I^-: Um für den Transfer $I_1^- \rightarrow I_n^-$ Inphase-Magnetisierung zu erzeugen, muss der selektierte Kernspin zunächst in die transversale Ebene überführt werden, z.B. durch $\alpha_\Phi = (\pi/2)_y$. Nun wird die Magnetisierung durch einen Gradienten (grau schraffiert) dephasiert. Nach erfolgtem Transfer wird die Magnetisierung durch einem dem Dephasierungsgradienten invers entsprechenden Gradienten rephasiert. Da nach der Rephasierung bereits ein detektierbarer Zustand vorliegt, entfallen alle weiteren Elemente.

Das bisher als Transfer-Block bezeichnete Element der Sequenz kann nun durch verschiedene Elemente ersetzt werden, die den präparierten Zustand des selektierten Spins durch die Spinkette transferieren. Im Rahmen der hier vorgestellten Experimente wurden isotropes Mischen (DIPSI), planares Mischen (Abbildung 10.4.a [74]), INEPT (Abbildung 10.4.b) und die bereits besprochene Soliton-Sequenz (Abbildung 10.4.c) verwendet.

Ein wichtiger Unterschied zwischen den verwendeten Transfer-Blöcken liegt in der Anzahl der transferierten Komponenten. So sind INEPT und planares Mischen ausschließlich dazu in der Lage, eine transversale Komponente zu übertragen. Der in Abbildung 10.4.a gezeigte Basiszyklus des planaren Mischens ist ausschließlich dazu in der Lage, Terme des Typs I_y zu transferieren. Der in Abbildung 10.4.b gezeigt INEPT-Schritt ist in Abhängigkeit der Phase Φ auf den Transfer von Termen $I_{x\vee y}(\Phi = y \vee x)$ beschränkt. Verwendet man hingegen isotropes Mischen oder die Soliton-Sequenz, so können gleichzeitig alle transversalen Komponenten transferiert werden.

Planares Mischen

Im Fall des planaren Mischens gilt zu beachten, daß durch die Sequenz der planare Hamilton-Operator $\mathcal{H}_{xx+yy}$ durch sequentielle Implementierung der jeweiligen bilinearen Anteile $\mathcal{H}_{xx}$ bzw. $\mathcal{H}_{yy}$ generiert wird. In Mehrspin-Systemen kommutieren diese Terme nur für $\tau \ll 1/J$ [75]. Somit bestimmen sich die Parameter τ und n durch empirische Maximierung des Antwortsignals. Der optimale Transfer wurde bei einer Zykluszeit von $2\tau = 7,6$ ms und $n = 7$ gefunden. Dies ergibt eine Mischeit von 53,20 ms, was gut mit den in [74] bestimmten Daten korrespondiert ($2\tau = 8$ ms, $\tau_{\mathrm{mix}} \simeq 65$ ms).

Sequentielles INEPT

Die experimentellen Parameter der sequentiellen INEPT-Sequenz werden durch das verwendete Spinsystem bestimmt. Um einen Spinzustand über eine Kopplung zu transferieren, muß die Periode $\tau = 1/2J$ gewählt werden. Da man in jedem Schritt genau eine Kopplung überbrückt, beträgt $n = N - 1$. Die letzte τ-Periode dient dazu, den Antiphase-Term zu refokussieren. Damit dies geschieht, darf nur die Kopplung aus der Transferrichtung aktiv sein, andere Kopplungen müssen gegebenenfalls entkoppelt werden. Da in den an Lysin durchgeführten Experimenten C_α der Zielkern ist, mußte die Kopplung $J_{C_\alpha,C(O)}$ entkoppelt werden. Dazu wurde der in der letzten τ-Phase vorhandene π-Puls so kalibriert, daß er bei der C(O)-Resonanz seinen Null-Durchgang besitzt.

SOLITON

Wie bereits in Abschnitt 10.1 erläutert, beträgt die Dauer einer Periode $\tau = 1/2J$. Während des Auf- bzw. Abbaus des Soliton-Operators sind nach Gleichungen (10.9) und (10.10) noch zusätzliche selektive Pulse erforderlich. Der Aufbau beginnt mit zwei selektiven Pulsen, die im Experiment jedoch ausgelassen werden können, weil sie nur die Elemente des Start-Vektors umsortieren, die Werte jedoch gleich bleiben. Die nächsten zwei selektiven Pulse lassen sich durch einen unselektiven $(\pi/2)_x$-Puls und eine den Offset-Frequenzen der Kerne angepaßte Verschiebung des Refokussierungs-Pulses ersetzen. Hierbei wurde die Verschiebung Δ_1 so gewählt, daß C_δ aufgrund seiner Offset-Frequenz eine $(\pi/2)_z$-Rotation bezüglich C_ε erfuhr, d.h. C_ε erfährt

keine z-Rotation. Während des Abbaus des Soliton-Operators wurden die gleichen Prinzipien auf die Implementierung der selektiven Pulse angewendet. Hierbei ist die Verschiebung Δ_2 so bestimmt, daß C_α bezogen auf C_β eine $(\pi/2)_z$-Rotation und C_β keine effektive z-Rotation erfährt. Nun können die urspünglich selektiven Pulse wiederum durch einen unselektiven $(\pi/2)_y$-Pulse ersetzt werden. Der letzte ursprünglich selektive Puls kann prinzipiell unselektiv sein, weil hier der Zielkern bereits die vollständige Polarisation erhalten hat. Zusätzlich muss während des Soliton-Abbaus die Kopplung $J_{C_\alpha,C(O)}$ entkoppelt werden, weil C_α den Zielkern darstellt. Dies wird dadurch erreicht, daß die letzten beiden Refokussierungspulse so kalibriert sind, daß sie bei der C(O)-Resonanz einen Nulldurchgang besitzten.

10.2.2 Experimentelle Daten

Die besprochenen Pulssequenzen wurden für den Transfer verschiedener Spinzustände verwendet. In Abbildung 10.5 sind die experimentellen NMR-Signale des Kerns Kerns C_α für diese Kohärenztransfers gezeigt. Hierbei beschränkt sich die Abbildung auf jene Pulssequenzen, welche den erwünschten Transfer ausschließlich unter Ausnutzung der schwachen Kopplung ermöglichen, die Daten für isotropes Mischen sind in Tabelle 10.2 aufgenommen.

Betrachtet man zunächst den Transfer kartesischer 1-Spinoperatoren (Abbildung 10.5.(a): $I_x(C_\varepsilon) \rightarrow I_x(C_\alpha)$; Abbildung 10.5.(b): $I_y(C_\varepsilon) \rightarrow I_y(C_\alpha)$; Abbildung 10.5.(c): $I_z(C_\varepsilon) \rightarrow I_z(C_\alpha)$), so erkennt man, daß alle Pulssequenzen wie erwartet mindestens eine transversale Komponente entlang der Spinkette transferieren. Im speziellen ermöglicht das sequentielle INEPT mit $\Phi = y$ den Transfer von x-Magnetisierung, während planares Mischen den erwarteten Transfer von y-Magnetisierung zeigt. Wie bereits in Abbildung 10.1.B dargestellt, ist die Propagation effektiver Soliton-Operatoren darauf ausgelegt, alle kartesischen 1-Spinzuständen transferieren zu können. Wie Abbildung 10.5 zu entnehmen, läßt sich dies experimentell bestätigen. Zu beachten ist hierbei, daß durch die Sequenz in Abhängigkeit der zu übertragenden Komponente zwei unterschiedliche Typen von Operatoren erzeugt werden, im speziellen werden die x- und z-Komponenten durch die Erzeugung bilinearer Operatoren durch die Kette propagiert, während die y-Komponente einen durch trilineare Operatoren gebildeten Transferweg beschreibt. In Abhängigkeit dieser Transferwege während sowohl Auf- und Abbau als auch Propagation des

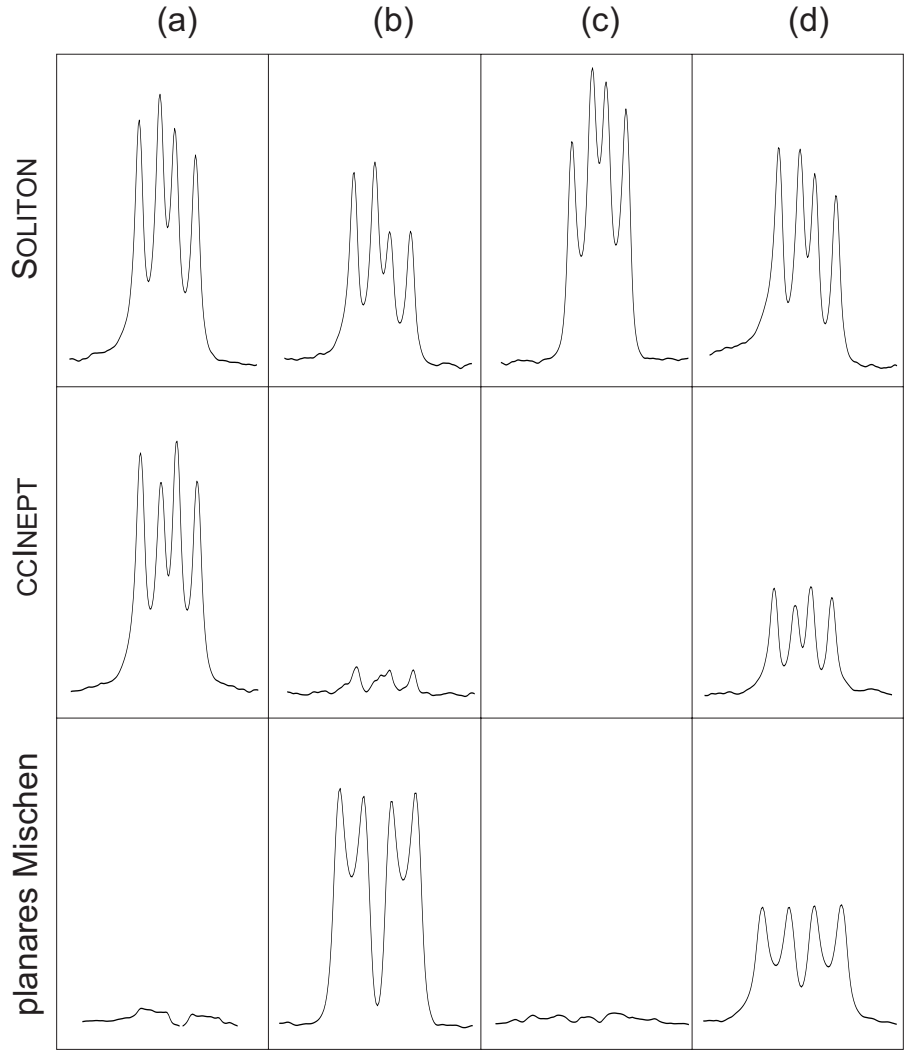

ABBILDUNG 10.5: NMR-Signale des Kernspins C_α nach erfolgtem Transfer entlang der Spinkette für verschiedene Pulssequenzen (SOLITON: $\tau = 14,3$ ms, $\Delta_1 = 129,8$ μs; sequentielles INEPT: $\tau = 14,3$ ms, $n = 4$, planares Mischen: $\tau = 3,8$ ms, $n = 7$). Gezeigt sind erfolgte Transfers von (a) $I_x(C_\varepsilon)$, (b) $I_y(C_\varepsilon)$, (c) $I_z(C_\varepsilon)$ und (d) $I^-(C_\varepsilon)$.

Soliton-Operators ist der Einfluß von Relaxation unterschiedlich. Durch diese Tatsache erklärt sich zum einen die unterschiedliche Signalform als auch die Variation in der Intensität des Signals bei dem Transfer von x- und y-Magnetisierung. Der Transfer der z-Komponente erfolgt über den bilinearen Weg, so daß die Signalform ähnlich zu dem Transfer der x-Komponente ist, jedoch ist die Intensität des Signals ein wenig größer. Dies läßt sich durch die Tatsache erklären, daß die Relaxationsrate der longitudinalen Komponenten geringer ist als die der transversalen. Da im Mittel während des Transfers der z-Komponente mehr longitudinale Zustände exisitieren ist der Dämpfungseffekt durch Relaxation geringer.

Präpariert man -1-Quantenkohärenz als zu transferierenden Zustand (Abbildung 10.5.(d): $I^-(C_\varepsilon) \rightarrow I^-(C_\alpha)$), so ist zu erwarten, daß sequentielles INEPT und planares Mischen nur die Hälfte des ursprünglichen Zustandes übertragen, während die Soliton-Propagation zu vollständigem Transfer des Ausgangszustandes führen. D.h. das Antwortsignal der Soliton-Sequenz wird die doppelte Intensität verglichen mit den restlichen Sequenzen aufweisen. Hierbei gilt zu beachten, daß die gezeigten NMR-Signale wegen der verwendeten Gradienten-Selektion nicht direkt mit den in Abbildung 10.5.(a)-(c) gezeigten vegleichbar sind. Vergleicht man die Antwort-Signale der verwendeten Pulssequenzen jedoch untereinander, so wird ersichtlich, daß der Soliton-Transfer diese Annahme erfüllt. Da -1-Quantenkohärenz $I^- = I_x - \imath I_y$ aus beiden transversalen Komponenten besteht, erscheint das Signal als Überlagerung der Formen des Transfers der Einzelkomponenten. Wie bereits festgestellt, unterscheiden sich die Transferwege dieser Komponenten durch die auftretenden Operatoren. Darüberhinaus wurde deutlich, daß der bilineare Transferweg der longitudinalen Komponente bezüglich der Relaxationseigenschaften günstiger ist, als der entsprechende bilinear erfolgende Transfer der x-Komponente. Hieraus läßt sich eine Verbesserung des Transfers von -1-Quantenkohärenz durch das Transferschema

$$I_1^- = I_{1,x} - \imath I_{1,y} \overset{(\pi/2)_x}{\longrightarrow} I_{1,x} - \imath I_{1,z} \overset{\text{Soliton}}{\longrightarrow} I_{n,x} - \imath I_{n,z} \overset{-(\pi/2)_x}{\longrightarrow} I_{n,x} - \imath I_{n,y} = I_n^-$$

erreichen, weil auf diesem Weg die günstigeren Transferpfade der x- und z-Komponente verwendet werden.

In Tabelle 10.2 sind die quantitativen Transferdaten der vier experimentell verwendeten Transfer-Elemente zusammengefaßt. Hierbei ist die Trans-

Sequenz	Transfer	τ_{mix} [msec]	$\eta(\tau_{\mathrm{mix}})$
planares Mischen	$I_{\varepsilon,y} \rightarrow I_{\alpha,y}$	53,20	0,2422
	$I_{\varepsilon,z} \rightarrow I_{\alpha,z}$		-
	$I_{\varepsilon}^{-} \rightarrow I_{\alpha}^{-}$		0,1405
sequentielles INEPT	$I_{\varepsilon,x} \rightarrow I_{\alpha,x}$	71,43	0,2890
	$I_{\varepsilon,z} \rightarrow I_{\alpha,z}$		-
	$I_{\varepsilon}^{-} \rightarrow I_{\alpha}^{-}$		0,1282
SOLITON	$I_{\varepsilon,x} \rightarrow I_{\alpha,x}$	85,71	0,2835
	$I_{\varepsilon,z} \rightarrow I_{\alpha,z}$		0,3007
	$I_{\varepsilon}^{-} \rightarrow I_{\alpha}^{-}$		0,2357
isotropes Mischen	$I_{\varepsilon,x} \rightarrow I_{\alpha,x}$	30,58	0,2669
	$I_{\varepsilon}^{-} \rightarrow I_{\alpha}^{-}$		0,2167

TABELLE 10.2: Experimentelle Mischzeiten und Transferamplituden für den Transfer eines transversalen, eines longitudinalen und eines -1-Quantenkohärenz entsprechenden Spinzustandes.

feramplitude als auf einen direkten Anregungszustand normiertes Transfer-Integral des C_α-NMR-Signals definiert. Dadurch beschränkt sich der Wertebereich der Transferamplitude auf $\eta \in [0, 1]$. Die absoluten Werte bezüglich des Transfers von -1-Quantenkohärenzen liegen wegen der hierbei verwendeten Gradientenselektion unter den entsprechenden transversalen Transferamplituden. Vergleicht man jedoch die Werte einer Transferart für die verschiedenen Sequenzen, so bestätigen die Werte die zuvor qualitativ gemachten Aussagen. Die Einflüsse von Relaxation wurden bereits diskutiert.

Betrachtet man die Mischzeiten, so wird deutlich, daß die Dauer des Transfers (bei gleicher Art der Kopplung) mit der Anzahl der übertragbaren Terme steigt. Wie bereits erwähnt, wird durch isotropes Mischen ein isotroper Hamiltonoperator erzeugt, wodurch die Mischzeit im Vergleich zu den weiteren Pulssequnzen erheblich geringer ist.

10.3 Schlußfolgerung

Es konnte gezeigt werden, daß durch die Propagation effektiver Soliton-Operatoren Kohärenztransfer in schwach gekoppelten Ising-Spinketten experimentell umgesetzt werden kann. Die in Kapitel 5.2 für dieselbe Zielsetzung durchgeführten Optimierungen zeigen, daß der sich Zeitbedarf der Soliton-

sequenz mit $\tau_{\mathrm{Sol}} = (N+1)/2J$ in der Näher der minimalen Zeit für den vollständigen Transfer von -1-Quantenkohärenz befindet, d.h. die Sequenz stellt eine effiziente Methode für solche Kohärenztransfers dar, die jedoch noch Spielraum für Verbesserungen bietet.

Teil IV

Software

Kapitel 11

Optimierungsbibliothek
OCTANE

Die Optimierungs-Bibliothek OCTANE (Optimum Control Theory Applied to NMR Experiments) wurde entwickelt, um den Prozeß der Pulsprogramm-Entwicklung zu automatisieren. Hierbei sollte das in Teil I vorgestellten theoretische Rahmenwerk eingesetzt werden. Die verwendeten Algorithmen wurden zunächst als Prototypen in Matlab [76] implementiert und getestet. Da die Verwendung dieser Umgebung jedoch eine interpretierte Code-Ausführung bedeutet, was das Laufzeit-Verhalten der Routinen negativ beeinflußt, wurde die endgültige Version der Optimierungs-Bibliothek in der Programmier-Sprache C++ [77]-[82] realisiert[1]. Dies bot über das verbesserte Laufzeitverhalten hinaus die Möglichkeit, den Code nach einem objektorientierten Ansatz[2] zu strukturieren, was die Verwendung, Pflege und Erweiterung der Bibliothek stark vereinfacht. Ein grundlegendes Objekt wird duch die Templat-Klasse [77]-[80] `Matrix` dargstellt. Diese Klasse stellt die benötigten Matrix-Oprationen typunabhängig zur Verfügung. Diese Klasse kann unabhängig von der Optimierungs-Bibliothek verwendet werden, ihre Synthax wird im Referenz-Teil IV in Kapitel 12 detailliert erläutert.

[1] Es wurde zunächst auch versucht, den Matlab-Compiler von *The MathWorks* zu verwenden, der Matlab-Code autmatisch in C++-Code übersetzen soll und dann mit einem nativen Compiler unter zu Hilfenahme der Matlab C/C++ Math Library in lauffähigen Binärcode übersetzt werden soll. Dies scheiterte jedoch an der Umsetzung der von *The MathWorks* bereitgestellten Tools; die Probleme konnten trotz intensiver Beratung mit Support-Mitarbeitern und Entwicklern von *The MathWorks* nicht zufriedenstellend gelöst werden.

[2] Die objektorientierte Programmierung (OOP) faßt Daten und die zur Bearbeitung benötigten Funktionen (Methoden) in einem Variablentyp (Klasse) zusammen. Variablen dieses Typs bezeichnet man als Objekt.

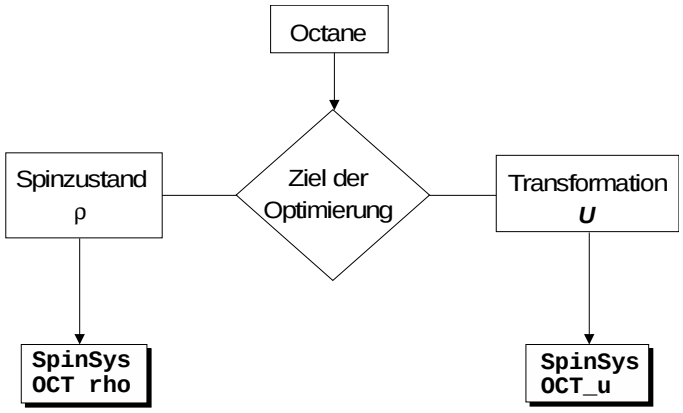

ABBILDUNG 11.1: Die Problemstellung bestimmt die Wahl der zu verwendenden Klassen. In dem Falle, daß das Ziel der Optimierung durch Spinzustände beschreibbar ist, kommt die Klasse `OCT_rho` zur Anwendung. Soll jedoch eine Pulssequenz berechnet werden, die eine Transformation implementiert, so ist die Klasse `OCT_u` zu verwenden. Beide Klassen sind von der Basisklasse `OCT_base` abgeleitet (vergl. Abbildung 11.2). Beiden Problemen gemeinsam ist die Darstellung der Pulssequenz in Form von Spinmatrizen, weshalb die Klasse `SpinSys` in allen Fällen Anwendung findet.

OCTANE kann derzeit (ab Version 0.6) zwei grundsätzliche Optimierungs-Probleme behandeln (vergl. Abbildung 11.1). In der NMR-Spektroskopie ist in der Regel die Überführung eines Spinsystems (dargestellt durch seine Dichtematrix) von einem definierten Anfangs-Zustand in einen End-Zustand Ziel eines Experimentes. Eine hierauf abzielende Optimierung muß nun Routinen beinhalten, die diese Spin-Zustände als Grundlage für die Berechnung einer Pulssequenz verwenden. Die Klasse `OCT_rho` stellt die für diesen Fall benötigten Methoden zur Verfügung. In der Quanten-Informationsverarbeitung ist es jedoch häufig erstrebenswert, eine gewisse Transformation (repräsentiert durch den zugehörigen Propagator) durch eine Pulssequenz zu implementieren. Die entsprechende Klasse `OCT_u` enthält die dazu benötigten Methoden. Für eine bessere Übersicht sind in Abbildung 11.2 die Abhängigkeiten und Vererbungsebenen der Bibliothek in einem UML-Digramm (Unified Modelling Language) [83] skizziert.

Im Laufe dieses Kapitels werden die verschiedenen Klassen erklärt, die von der Bibliothek bereit gestellt werden. Hierbei wird zunächt die Konstruktion der entsprechenden Objekte besprochen, worauf eine Erläuterung

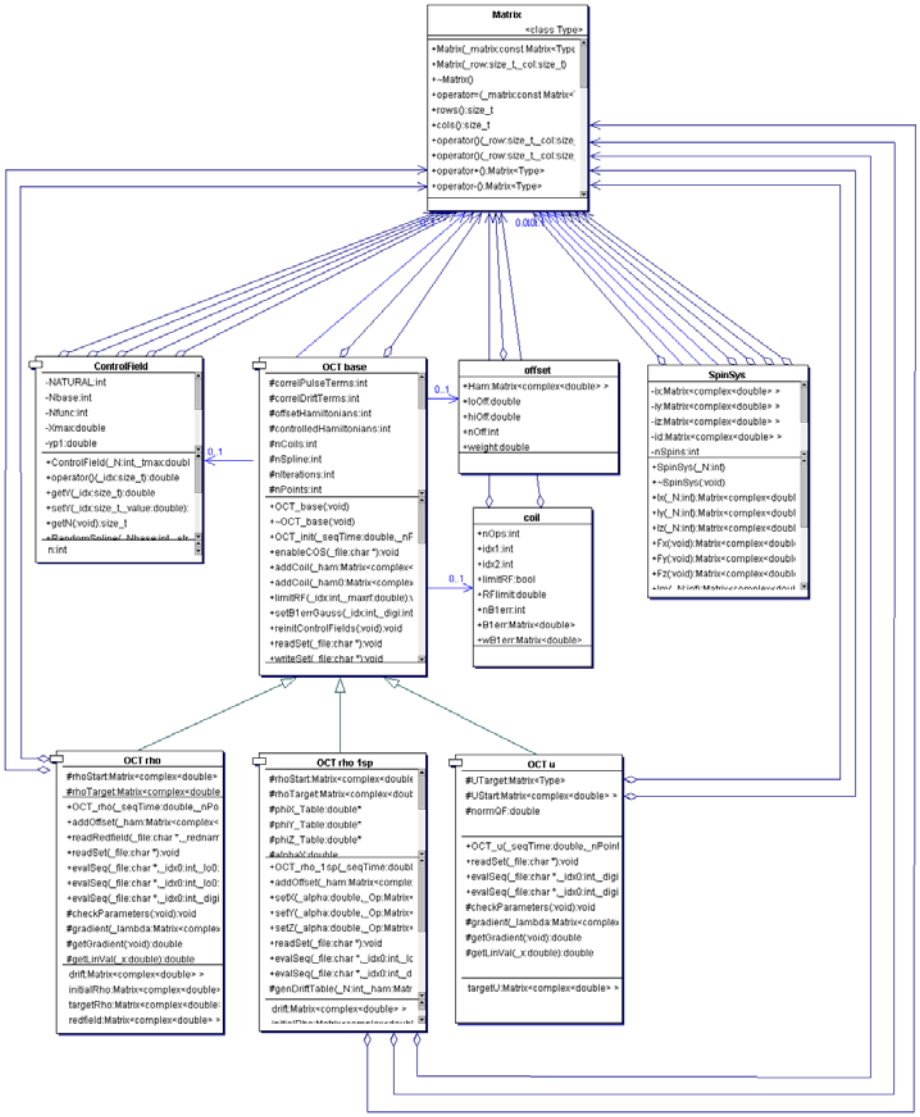

ABBILDUNG 11.2: UML-Digramm von OCTANE, welches Abhängigkeiten aller Objekte beschreibt. Aufgeführt sind die öffentlichen Methoden der Bibliothek.

aller öffentlichen Methoden[3] der Klasse folgt. Jedes Unterkapitel schließt mit einem einfachen Beispiel-Programm ab, was die Verwendung der Bibliothek zur Entwicklung eines spezifischen Optimierungs-Programmes verdeutlichen soll.

11.1 Definition eines Spinsystems

Grundlage der Darstellung eines NMR-Experimentes im Dichtematrix-Formalismus sind stets Spin-Matrizen. Eine Bibliothek, deren Ziel es ist, optimale Pulssequenzen zu berechnen, muß neben den benötigten Optimierungs-Routinen auch ein Rahmenwerk bereitstellen, mit dem sich die benötigten Spin-Operatoren darstellen lassen. Innerhalb von OCTANE wird dafür die Klasse `SpinSys` verwendet. Im folgenden wird die Verwendung dieser Klasse erläutert.

11.1.1 Objekt-Konstruktor

Zunächst muß der Anwender ein Objekt der Klasse `SpinSys` erzeugen. Zur Deklaration von Objekten dieser Klasse stellt OCTANE den Konstruktor `SpinSys( int )` bereit. Die Bedeutung des Parameters ist wie folgt.

- `int N`:
  Das Spinsystem soll über eine bestimmte Anzahl an Spins verfügen. Diese Zahl wird während der Objekt-Initialisierung dem Konstruktor als Parameter übergeben.

  Während der Objekt-Initalisierung wird eine vereinfachte kartesische Spinbasis $I_\nu(1), ..., I_\nu(N)$ mit $\nu \in [x, y, z]$ für die gegebene Größe des

[3]Durch den objektorientierten Ansatz besteht die Notwendigkeit hierarchischer Zugriffsrechte auf die Methoden einer Klasse. Diese werden über definierte Zugriffs-Spezifikationssymbole (*access specifiers*) dargestellt. Die Schnittstelle zur Verwendung der Klasse (*class interface*) wird durch öffentliche Methoden (*public members*) dargestellt. Zugriff auf diese Mehtoden ist auf der Ebene eines deklarierten Objektes möglich. Darüberhinaus gibt es private Methoden (*private members*), die die Daten der Klasse manipulieren. Diese stellen die Implementierung der Klasse dar (*class implementation*), weshalb der Zugriff intern auf das Objekt selbst beschränkt ist. Auch im Falle einer abgeleiteten Klassen sind diese auf die Basisklasse beschränkt. Eine spezielle Art privater Methoden wird durch die geschützten Methoden (*protected members*) dargestellt. Diese verhalten sich prinzipiell wie private Methoden mit einem auf die gesamte Vererbungshierarchie erweiterten Zugriffsbereich.

Spinsystems im Hilbert-Raum berechnet. Hierzu werden rekursiv so lange Kronecker-Produkte zwischen der Einheitsmatrix und der entsprechenden Pauli-Matrix gemäß Gleichung (2.28) berechnet, bis die Größe der Spinmatrizen $(2^N, 2^N)$ für ein N-Spinsystem erreicht hat.

Zusätzlich zur Berechnung der Spinbasis wird eine $N \times N$ Matrix reserviert, die zur Verwaltung der Kopplungs-Topologie dient. Diese Matrix wird als Null-Matrix initialisiert, d.h. jeder Spin wird als isoliert betrachtet. Will der Anwender Kopplungen in das System einführen, so kann er die entsprechende Interface-Funktion `setJ` (s. dort) verwenden.

11.1.2 Objekt-Methoden

Nachdem ein Objekt der Klasse `SpinSys` angelegt wurde, können die von dieser Klasse bereit gestellten Methoden verwendet werden. Diese Funktionen dienen dazu, Parameter des Spinsystems an das Objekt zu übergeben bzw. Parameter und Operatoren (Basis- und Spin-Hamilton-Operatoren) für das gegebene Spinsystem abzufragen.

- `void setJ( int, int, double )`:
  Bereits während der Objekt-Konstruktion wird eine Kopplungs-Matrix angelegt. Diese ist zunächst als Null-Matrix initialisiert, d.h. alle J-Kopplungen sind auf 0 Hz gesetzt. Durch Verwendung der Methode `setJ` kann man nun Werte in diese Matrix schreiben. Die Funktion erwartet drei Parameter. Die ersten beiden Parameter vom Typ `int` beschreiben die Indices der Spins in nicht-informatischer Zählweise[4], die mit einer J-Kopplung verbunden werden sollen. Der dritte Parameter vom Typ `double` ist die Größe der Kopplung in Hertz.

  Wird eine Kopplung zwischen zwei Spins unter Verwendung dieser Routine gespeichert, so wird intern auch das diagonal-symmetrische Matrix-Element mit diesem Wert versehen. Es reicht also aus, die Kopplung eines Spin-Paares (M, N) ein einziges Mal zu definieren; das entsprechende Element (N, M) wird automatisch korrekt verwaltet.

[4]Unter nicht-informatischer Zählweise versteht man eine Indizierung gemäß $i \in [1, N]$ (postive natrliche Zahlen). Als informatische Zählweise bezeichnet man eine Indizierung gemäß $i \in [0, I]$ (positive ganze Zahlen).

- `double J( int, int )`:
  Möchte man die gesetzte Kopplung zwischen einem Spin-Paar (M, N) abfragen, so kann man diese Routine benutzen. Sie liefert den Wert der Kopplungs-Matrix der ensprechenden als Parameter vom Typ `int` übergebenen Spin-Indices als Wert vom Typ `double` zurück. Wie unter der Beschreibung der Routine `setJ` angemerkt, werden diagonalsymmetrische Elemente der Kopplungs-Matrix automatisch mit gleichen Werten versehen. Daher kann man z.B. mit `J(2,1)` den Wert der Kopplung des Spin-Paares $(1, 2)$ abfragen, der vorher mit `setJ(1,2, 1.0)` gesetzt wurde.

- `Matrix<complex<double> > Ix( int )`:
  Diese Routine dient dazu, Operatoren I_x der vereinfachten Spin-Basis abzufragen. In der Regel wird diese Routine in Verbindung mit ihren Analoga `Iy` und `Iz` verwendet, um alle im Rahmen einer Optimierung benötigten Operatoren aufzubauen. Diese Routine verlangt einen Parameter, der dem Index des Spins in nicht-informatischer Zählweise entspricht, dessen Basis-Matrix I_x man abfragen möchte.

- `Matrix<complex<double> > Iy( int )`:
  Diese Routine ist analog zu der unter `Ix` beschriebenen. Sie liefert Basis-Matrizen I_y als Rückgabe-Wert.

- `Matrix<complex<double> > Iz( int )`:
  Diese Routine ist analog zu der unter `Ix` beschriebenen. Sie liefert Basis-Matrizen I_z als Rückgabe-Wert.

- `Matrix<complex<double> > Im( int )`:
  Diese Methode stellt den Absteige-Operator

$$I^- = I_x - \imath I_y \tag{11.1}$$

  als Rückgabewert bereit. Der Parameter entspricht dem Index des Spins, dessen Absteige-Operators berechnet werden soll.

- `Matrix<complex<double> > Ip( int )`:
  Diese Methode stellt den Aufsteige-Operator

$$I^+ = I_x + \imath I_y \tag{11.2}$$

als Rückgabewert bereit. Der Parameter entspricht dem Index des Spins, dessen Aufsteige-Operators berechnet werden soll.

- `Matrix<complex<double> > Fx( void )`:
  Diese Methode stellt den Summen-Operator

$$F_x = \sum_1^N I_{x,n} \tag{11.3}$$

  als Rückgabewert bereit. Die Summe wird über alle N Kernspins des Spinsystems berechnet und ist daher parameterfrei.

- `Matrix<complex<double> > Fy( void )`:
  Diese Routine ist analog zu der unter `Fx` beschriebenen. Sie liefert Basis-Matrizen F_y als Rückgabe-Wert.

- `Matrix<complex<double> > Fz( void )`:
  Diese Routine ist analog zu der unter `Fx` beschriebenen. Sie liefert den Summenoperator F_z als Rückgabe-Wert.

- `Matrix<complex<double> > Hweak( void )`:
  Der Hamilton-Operator der schwachen Kopplung

$$H_{weak} = 2\pi \sum_{i=2}^{N} \sum_{k=1}^{i-1} J_{i,k} I_z(i) I_z(k) \tag{11.4}$$

  wird von dieser Routine als Rückgabewert bereitgestellt. Er wird für das aktuell durch die Kopplungs-Matrix definierte Spinsystem berechnet.

- `Matrix<complex<double> > Hiso( void )`:
  Der isotrope skalare Kopplungs-Hamiltonian

$$H_{iso} = 2\pi \sum_{i=2}^{N} \sum_{k=1}^{i-1} J_{i,k} (I_x(i) I_x(k) + I_y(i) I_y(k) + I_z(i) I_z(k)) \tag{11.5}$$

  wird von dieser Routine als Rückgabewert bereitgestellt. Er wird für das aktuell durch die Kopplungs-Matrix definierte Spinsystem berechnet.

11.1.3 Beispiel-Programm

Innerhalb dieses Kapitels soll ein einfaches Beispiel-Programm entwickelt werden. Ziel dieses Programmes ist es, ein einfaches Optimierungs-Problem zu behandeln. Bevor jedoch das Lösungsverfahren für dieses Problem in ein Programm umgesetzt werden kann, muß man das Spinsystem initialisieren, für das im Anschluß eine Pulssequenz optimiert werden soll. Im Beispiel-Programm soll nun ein 3-Spinsystem mit einer linearen Kopplungs-Topologie beschrieben werden. Da für das Optimierungs-Problem zunächst nur ideale Bedingungen angenommen werden, sollen alle Spin-Kopplungen der Einheitskopplung entsprechen. Unter Verwendung von OCTANE wird dies wie folgt in Programm-Code umgesetzt.

```
1 #include "spinsys.h"
:
: int main( void )
: {
2    SpinSys mySys(3);
:
3    mySys.setJ(1,2, 1.0);
4    mySys.setJ(2,3, 1.0);
: }
```

Unter Position 1 wird zunächst die benötigte Spinsystem-Bilbliothek eingebunden. Unter Position 2 wird durch Aufruf des Spinsystem-Konstruktors ein Objekt der Klasse `SpinSys` unter dem Namen `mySys` angelegt, welches für die Verwaltung von 3 Spins ausgelegt ist. Die Größe des Spinsystems wird dem Konstrukor als Parameter übergeben. Unter den Positionen 3 und 4 wird nun die Kopplungstopologie festgelegt. In diesem Falle soll die Kopplungs-Topologie linear sein, d.h. es müssen nur Kopplung zwischen den Spinpaaren $(1,2)$ und $(2,3)$ definiert werden. Im Beispiel werden Einheits-Kopplungen $J_{12} = J_{23} = 1\text{Hz}$ an das Objekt übergeben. Damit ist das Spinsystem für eine Optimierung hinreichend beschrieben.

11.2 Basis-Klasse OCT_base

Vor den in den Unterkapiteln 11.3-11.4 zu besprechenden abgeleiteten Klassen zur vollständigen Implementierung einer Optimierungen, sollen an dieser Stelle die öffentlichen Methoden der Basis-Klasse vorgestellt werden, weil diese einen integraler Bestandteil der aus ihr abgeleiteten Klassen darstellen. Die Basisklasse selbst kann nicht eigenständig verwendet werden, viele ihrer (geschützten) Methoden sind nur virtuelle Funktionen und werden daher erst durch Überladung in den Unterklassen mit den entsprechenden algorithmischen Strukturen versehen. Darüberhinaus implementieren die Unterklassen zusätzlich neue Methoden, die eine Optimierung erst ermöglichen. Da jedoch grundlegende Methoden von den Unterklassen ererbt werden, soll deren Syntax an dieser Stelle erklärt werden. Da diese Klasse nicht direkt vom Anwender zu verwenden ist, wird auf das Beispiel am Ende dieses Unterkapitels verzichtet. Es sei auf die Beispiele der abgeleiteten Klassen verwiesen, die die hier besprochenen Methoden enthalten.

11.2.1 Objekt-Konstruktor

Die Basis-Klasse verfügt über einen Parameter-freien Standard-Konstruktor, der allerdings nur auf Ebene des Compilers eine Rolle spielt. Das Objekt selbst wird intern über die Konstruktoren der respektiven Unterklassen initialisiert (s. dort und `OCT_init`).

11.2.2 Objekt-Methoden

- `void OCT_init( double, int, int, double )`:
  Diese Methode wird von den abgeleiteten Klassen verwendet, die dynamische Initialisierung der Basisklasse vorzunehmen. Sie erwartet vier Parameter, deren Bedeutung wie folgt ist.

  `double seqTime`:
  Die im späteren Programm-Ablauf noch zu definerenden Kontroll-Felder (s. `addCoil`) sollen für eine bestimmte Zeit τ_p auf das Spinsystem eingestrahlt werden. Dieser Parameter definiert die Zeit der Einstrahlung in Sekunden.

`int nPointsPerSec`:
Dieser Parameter bestimmt die Anzahl der für die digitale Darstellung der Puls-Shape zu verwendenden Datenpunkte. In der Regel sind für kurze Zeiten τ_p geringere Digitaliserungen notwendig, daher definert dieser Parameter die Anzahl Datenpunkte pro Sekunde. Je höher dieser Wert gewählt wird, desto häufiger läßt man eine Veränderung der Radiofrequenz-Felder zu.

Bei der Wahl der Digitalisierung der Puls-Shape sollte man beachten, daß der Radiofrequenz-Verstärker im Experiment mit dieser zeitlichen Digitalisierung die Amplituden der einzustrahlenden Radiofrequenz verändern soll. Die vorhandene Spektrometer-Ausstattung definiert somit das sinnvolle Maximum der Digitalisierung. Darüber hinaus werden während der Optimierung für jeden Digitalisierungs-Punkt Gradienten berechnet, d.h. die Anzahl numerischer Operationen pro Iteration wird durch die Wahl dieses Parameters direkt beeinflußt; je feiner die Digitalisierung gewählt wird, desto mehr Operationen müssen durchgeführt werden.

Neben der vom Benutzer zu ermittelnden maximalen Digitalisierung sollte eine minimale absolute Digitalisierung von 4 Punkten nicht unterschritten werden. Zum einen muß das Kontrollfeld über genügend Freiheitsgrade verfügen, um optimale Ergebnisse erzielen zu können, zum anderen werden 10% der Gesamt-Digitalisierung bei einer digitalen Auflösung über 50 Punkten bzw. 50% für kleinere Digitalisierungen während der Initialisierung mit der maximalen Amplitude `uStrength` zufällig erzeug, während die restlichen Datenpunkte durch einen kubsichen Spline-Fit über diese Werte berechnet werden, um eine glatte Kurvenform der Puls-Shape zu gewährleisten. Die Berechnung eines kubischen Spline-Fits ist bei einer zu geringen Digitalisierung nicht mehr möglich. Daher entsprechen in diesem Fall alle digitalen RF-Werte der erzeugten Pulsform Zufallswerten.

`int nIterations`:
Unterschreitet die Differenz des funktionalen Optimierungsziels der letzten beiden Optimierungs-Schritte einen Schwellenwert, so wird der Algorithmus als konvergiert angenommen und die Optimierung beendet.

Dafür sind in der Regel N Iterationen zu durchlaufen. Durch diesen Parameter definiert man eine Obergrenze an zu durchlaufenden Iterationen, d.h. der Algorithmus wird gezwungen, sich nach diesen `nIterations` $< N$ Iterationen zu beenden, ohne das normale Abbruchkriterium erfüllt zu haben.

`double uStrength`:
Ausgangspunkt der Optimierung ist eine zufällig erzeugte Pulsform. Neben der bereits definierten Länge τ_p und Digitalisierung der Pulsform benötigt das Objekt zur Erzeugung einer solchen Sequenz noch die erwünschte Radiofrequenz-Feldstärke der zufällig erzeugten Start-Pulsform. Die maximale Amplitude wird durch diesen Parameter gesetzt.

- `int addCoil(Matrix<complex<double> >)`,
  `int addCoil(Matrix<complex<double> >,Matrix<complex<double> >)`:
  Neben dem durch den Drift-Hamiltoninan (s. dazu Abschnitt 11.6) definierten statischen Anteil enthält ein Hamilton-Operator, der die Puls-Sequenz zu jeder Zeit beschreibt, einen dynamischen Anteil, der den eingestrahlten Radiofrequenz-Feldern entspricht:

$$\mathcal{H}_{seq} = \mathcal{H}_{stat} + \mathcal{H}_{RF}, \tag{11.6}$$

wobei der die Puls-Sequenz beschreibende Anteil $\mathcal{H}_{RF}$ in der Regel als Summe über alle verfügbaren Kontrollen, die auf das Spin-System ausgeübt werden können, dargestellt wird:

$$\mathcal{H}_{RF} = \sum_j \sum_k u_{j,k}(t) I_k(j). \tag{11.7}$$

Hierbei läuft der Index j über die selektiv bepulsbaren Kerne, d.h. in der Regel die verfügbaren RF-Kanäle; der Index k läuft über die während der Optimierung erwünschten Phasen der RF-Felder, welche in der Regel durch eine orthogonale Phasen-Basis dargestellt werden. Der Term $u_{j,k}(t)$ repräsentiert die zeitliche Modulation der RF-Felder; die optimale Form dieser Terme ist das Ziel der Optimierung und wird somit automatisch intern verwaltet.

OCTANE-Syntax	$\mathcal{H}_{RF} =$	
`addCoil(_2pi*mySys.Ix(1),_2pi*mySys.Iy(1));`		$u_1(t)2\pi I_x$
	$+$	$u_2(t)2\pi I_y$
`addCoil(_2pi*mySys.Ix(2));`	$+$	$u_3(t)2\pi S_x$

TABELLE 11.1: Übersezung von erwünschten Pulsen (s. Text) in OCTANE-Syntax am Beispiel eines hetero-nukleraren 2-Spin-Systems. Der Optimierung werden zwei RF-Kanäle hinzugefügt, wobei einer der Kanäle durch seine orthogonale Phasen-Basis I_x und I_y dargestellt wird, der andere durch eine feste Phasen-Basis S_x.

Durch die Methode `addCoil` können dem Optimierungs-Problem nun sequentiell Summanden $I_k(j)$ hinzugefügt werden. Damit man eine Optimierung jedoch starten kann, muß mindestens ein solcher Term definiert werden. Die darüberhinaus gehende Anzahl an Kontroll-Termen ist duch `sizeof(int)` limitiert, auf normalen 32-bit Systemen sind somit maximal 65536 Kontroll-Terme möglich. Die Reihenfolge der Definition im Quellcode ist nicht wichtig, weil Summen immer kommutieren. Darüberhinaus behandelt OCTANE die dynamischen, modulierten Hamiltonian-Operatoren stets als Einheit aus Modulations-Feld und Operator.

Für Routinen, die den Verwaltungs-Index der Kontroll-Terme benötigen, wird dieser von der Methode als Rückgabewert bereitgestellt. Diese Indices werden in der Regel von optionalen Methoden verlangt, die Eigenschaften eines physikalischen RF-Kanales modellieren (vergl. die Unterpunkte `limitRF` und `setB1errGauss` in Abschnitt 11.6). Daher werden Kontroll-Terme in Form logischer RF-Kanäle in OCTANE definiert, die entweder aus einem Operator zur Darstellung einer konstanten RF-Phase oder zwei Operatoren zur Beschreibung einer möglichen RF-Phase von 2π bestehen können.

In Tabelle 11.1 wird der Mechanismus des sequentiellen Definierens von Kontroll-Termen verdeutlicht. In dem dort angenommenen Falle eines hetero-nukleraren 2-Spin-Systems IS sollen beide Kerne über getrennte RF-Kanäle ansprechbar sein. Zur Veranschaulichung der Verwendung beider überladenen Varianten der Methode soll nur der erste Kanal

über eine RF-Phase von 2π verfügen, der zweite soll eine feste Phase x besitzen. Die zwei Methoden-Aufrufe resultieren somit in drei Kontroll-Feldern gemäß Gleichung (11.7).

11.3 Optimierung von Kohärenz-Transfers

In diesem Kapitel wird die Klasse `OCT_rho` vorgstellt. Sie wird zur Optimierung von Pulssequenzen verwendet, die ein Spinsystem in einer vorgegebenen Zeit möglichst optimal von einem definierten Anfangszustand in einen Endzustand überführen sollen. Hierzu müssen vom Anwender einige Parameter vorgegeben werden, welche intern in Datenstrukturen umgesetzt und verwaltet werden. Im folgenden wird das prinzipielle Design eines Optimierungs-Programmes beschrieben, das über den minimalen Funktionumfang verfügt.

Unter einem minimal definierten Optimierungs-Problem versteht man die Aufgabe eine Puls-Sequenz unter Ausschluß aller Stör-Effekte berechnen zu lassen. Im Falle der Darstellung des zu optimierenden Zustandes durch eine Spin-Dichtematrix zählen zu den Störefekten neben den Inhomogenitäten der eingestrahlten Radiofrequenz-Felder auch die Störungen durch die Eigenschaften des Spinsystems, d.h. Relaxation und die Abstände der Kern-Resonanzfrequenzen bezüglich der Trägerfrequenz ("RF-Offsets"). Dieses Problem wird deshalb als minimal bezeichnet, weil dies ausschließlich durch Parameter beschrieben wird, deren Definition OCTANE zwingend erwartet. Ein auf diese Weise minimal definiertes Optimierungs-Problem wird in der Regel dazu benutzt, um für den Kohärenz-Tranfer in einem Spinsystem die durch die Quantenmechanik bestimmten Grenzen ("Unitary Bounds" [58]) zu ermitteln bzw. die wichtige Frage zu beantworten, was die minimale Zeit ist, in der diese Grenze erreicht werden kann. Dieses Problem der Zeit-Optimalität wird in Kapitel 5 für verschiedene Fälle diskutiert.

Dieses minimale Optimierungs-Problem kann schließlich noch durch optionale Funktionen verfeinert werden. Eine Beschreibung dieser optionalen Routinen ist in Kapitel 11.6 zu finden.

Die Optimierung von Pulssequenzen zur Erreichung eines vorgebenen Kohärenz-Transfers ist innerhalb des Objektes `OCT_rho` gekapselt, d.h. der Anwender muß sich vor jeder Optimierung ein Objekt dieses Typs erzeugen. Hat man mittles des Konstruktors das Objekt initialisiert, so kann man das

Optimierungs-Problem unter Verwendung der Objekt-Methoden definieren.

11.3.1 Objekt-Konstruktor

Der Konstruktor `OCT_rho( double, int, int, double )` verlangt vier Parameter. Da diese Klasse aus der Basis-Klasse `OCT_base` abgeleitet ist, wird der Konstruktor neben der eigentlichen Objekt-Konstruktion auch zur Initialisierung der Basis-Klasse verwendet. Die Bedeutung der Parameter ist identisch mit denen in Abschnitt 11.2 unter Punkt `OCT_init` besprochenen.

11.3.2 Objekt-Methoden

Hat man das grundlegende Objekt angelegt, so muß man das Optimierungs-Problem beschreiben. Verwendet man diese Klasse, so ist das Ziel der Optimierung eine Pulssequenz, die ein Spinsystem aus einem definierten Anfangszustand in einen definierten Endzustand überführt. Neben den obligatorischen Methoden der Basisklasse müssen hierzu zwei Methoden der Unterklasse verwendet werden:

- `void setInitialRho( Matrix<complex<double> > )`:
  Die zu optimierende Puls-Sequenz soll ein Spin-System aus einem Start-Zustand in einen Ziel-Zustand überfüren. Im Dichtematrix-Formalismus wird der Zustand eines Spin-Systems durch die entsprechende Dichtematrix des Spinsystems zu einer gegebenen Zeit beschrieben. In der Regel kennt man den Zustand des Spin-Systems vor Beginn des NMR-Experimentes, das optimiert werden soll. Dies kann entweder dem thermischen Dichteoperator entsprechen, der durch die Boltzmann-Verteilung bestimmt ist, wenn keine Präparations-Phase dem Transfer-Block vorangestellt wird. Oftmals hat man schon einige Transformationen der Dichtematrix vorgenommen, um das Spin-System in einen definierten vom thermischen Gleichgewicht entfernten Zustand zu überführen.

  Diesen Zustand vor Beginn des zu optimierenden NMR-Experimentes muß man der Optimierungs-Routine mitteilen. Dazu benutzt man die Methode `setInitialRho`. Die Start-Dichtematrix wird dieser Funktion als Parameter übergeben.

- `void setTargetRho( Matrix<complex<double> > )`:
  Wie unter dem vorigen Punkt erläutert muß man den Start- und Zielzustand des Spin-Systems kennen. Die Anwendung von Radiofrequenz-Pulsen beschrieben durch die vorhandenen Kontroll-Felder überführt nun das Spin-System vom definierten Start-Zustand aus in einen Endzustand. Durch Projektion dieses Endzustandes auf den gewünschten Ziel-Zustand kann man ein Maß dafür errechnen, wie gut die aktuelle Pulssequenz für diesen Kohärenz-Transfer geeignet ist.

  Den Zustand, den man aus einem gegebenen Start-Zustand erreichen möchte, definiert man mittels der Methode `setTargetRho`. Die Routine erwartet die diesen Zustand beschreibende Dichtematrix als Parameter.

11.3.3 Beispiel-Programm

In diesem Abschnitt soll das in Abschnitt 11.1.3 begonne Beispiel-Programm nun um das eigentliche Optimierungs-Problem ergänzt werden. Das Spinsystem soll vom Typ I_2S sein, somit reicht zur Beschreibung des Drift-Hamiltonians der Hamilton-Operator der schwachen Kopplung aus. Zudem sind zwei Kern-Sorten vorhanden, was die Möglichkeit bietet, diese über getrennte RF-Kanäle zu addressieren. Der durch die zu berechnende Pulssequenz zu erreichende Kohährenz-Transfer soll ein Inphase-Transfer $I_1^- \rightarrow I_2^-$ sein, was einer Start-Dichtematrix $\sigma(0) = I_{x,1} - \imath I_{y,1}$ und einem Ziel-Dichteoperator von $\sigma(T) = I_{x,2} - \imath I_{y,2}$ entspricht. Die Zeit, innerhalb welcher der Transfer erreicht werden soll, wird $T = 1/J$ gewählt. Im Besipiel-Programm werden diese Forderungen wie folgt mittels OCTANE-Syntax formuliert:

```
: #include "spinsys.h"
1 #include "oct_rho.h"
:
: int main( void )
: {
:    SpinSys mySys(3);
:    mySys.setJ(1,2, 1.0);
:    mySys.setJ(2,3, 1.0);
:
```

```
2    OCT_rho myOCT(1, 200, 1e6, 1);
:
3    myOCT.setInitialRho( (mySys.Ix(1) - _i*mySys.Iy(1))
:      / (mySys.Ix(1) - _i*mySys.Iy(1)).normFrobenius() );
4    myOCT.setTargetRho( (mySys.Ix(3) - _i*mySys.Iy(3))
:      / (mySys.Ix(1) - _i*mySys.Iy(3)).normFrobenius() );
:
5    myOCT.setDrift( mySys.Hweak() );
:
6    myOCT.addCoil( _2pi* ( mySys.Ix(1) + mySys.Ix(3) ),
:                   _2pi* ( mySys.Iy(1) + mySys.Iy(3) ) );
7    myOCT.addCoil( _2pi* mySys.Ix(2),
:                   _2pi* mySys.Iy(2) );
:
8    myOCT.optimizeCG();
: }
```

In Position 1 wird zunächt der kontroll-theoretische Teil `OCT_rho` der Bibliothek eingebunden, der die Behandlung von Kohärenz-Transfers ermöglicht. Nun kann in Position 2 ein Objekt `myOCT` der Klasse `OCT_rho` initialisiert werden. Hierbei werden die grundlegenden Optimierungs-Parameter an den Konstruktor übergeben: (1) Die Pulssequenz soll eine Länge von 1s haben, (2) die Kontroll-Felder sollen in 200 Punkte pro Sekunde digitalisert werden, (3) die Optimierung soll sich spätestens nach 1.000.000 Iterationen beenden und (4) die initiale Feldstärke soll 1Hz betragen. Die geringe Start-Feldstärke begründet sich dadurch, daß keine Offset-Effekte in die Optimierung eingeschlossen werden sollen, daher ist eine geringe Feldstärke zur Kontrolle der Spindynamik ausreichend. Nachdem nun dieses Objekt initalisiert ist, werden die für die Optimierung relevanten Operatoren hinzugefügt. In Position 3 wird der Start-Dichteoperator I_1^- definiert. Hierbei kommen die Interface-Funktionen des Spinsystem-Objektes zum Einsatz, die die Basis-Operatoren des Spinsystems zurückgeben. Damit das Skalarprodukt zwischen dem End-Operator innerhalb eines Intervals von $[0, 1]$ liegt, wird dieser Operator noch mit seiner Frobenius-Norm normiert. In Position 4 wird auf gleiche Weise der Ziel-Dichteoperator I_2^- normiert an das Optimierungs-Objekt übergeben. In Position 5 wird der optionale Drift-Hamiltonian definiert; würde im gege-

ben Fall allerdings kein Drift-Term definiert, so wäre der gewollte Kohärenz-Transfer durch keine Pulssequenz zu erreichen. Da es sich im betrachteten Fall um ein heteronuklerares Spinsystem handelt, für dessen Beschreibung der schwache Kopplungs-Hamiltonian verwendet werden soll, wird dieser über die Interface-Funktion `Hweak` des Spinsystem-Objektes abgefragt und an das Optimierungs-Objekt übergeben. In den Positionen 6 und 7 werden die logischen RF-Kanäle initalisiert. Für ein I_2S-System sind zwei RF-Kanäle möglich, welche durch je zwei orthogonale Basis-Phasen dargestellt wird. So stellt die Position 6 den auf die I-Spins wirkenden RF-Kanal dar, während die Position 7 den auf den S-Spin wirkenden RF-Kanal darstellen. Um nun die Optimierung zu starten wird in Position 8 eine Methode verwendet, die iterativ die Pulssequenz für das definierte Optimierungs-Problem berechnet; die Optimierungs-Methoden werden in Abschnitt 11.5 detailliert besprochen.

11.4 Optimierung von Transformationen

In diesem Kapitel wird die Klasse `OCT_u` besprochen werden. Sie wird zur Optimierung von Pulssequenzen verwendet, die in einer vorgegebenen Zeit eine Transformation (gegeben durch den entsprechenden Propagator) möglichst optimal erreichen sollen. Wie die bereits diskutierte Klasse `OCT_rho` ist sie von der Basisklasse `OCT_base` abgeleitet und an das Optimierungsproblem angepaßt.

Probleme, die man mit dieser Klasse bearbeiten kann, entstammen häufig der Quanten-Informationsverarbeitung. Hierbei ist es oftmals so, daß man eine gewisse Transformation durchführen möchte (z.B. SWAP-Gatter), wobei Zeitoptimalität eine wichtige Rolle spielt (vergl. Kapitel 9).

Im Gegensatz zur Optimierung von Pulssequenzen zur Erreichung eines Kohärenztransfers ist die Anzahl der Definitionen für das Problem geringer. In diesem Fall reicht es aus, den Zielpropagator zu kennen, der durch die Pulssequenz erreicht werden soll; der Startpropagator entspricht in jedem Falle der Einheitsmatrix.

11.4.1 Objekt-Konstruktor

Der Konstruktor `OCT_u( double, int, int, double )` verlangt vier Parameter. Da diese Klasse aus der Basis-Klasse `OCT_base` abgeleitet ist, wird der

Konstruktor neben der eigentlichen Objekt-Konstruktion auch zur Initialisierung der Basis-Klasse verwendet. Die Bedeutung der Parameter ist identisch mit denen in Abschnitt 11.2 unter Punkt `OCT_init` besprochenen.

11.4.2 Objekt-Methoden

- `void setTargetU( Matrix<complex<double> > )`:
  Wie bereits in der Einleitung erläuter, reicht es aus, den Zielpropagator zu kennen, um eine Pulssequenz zu berechnen, durch die dieser dargestellt werden soll. Ausgehend von einer beliebigen Pulssequenz kann man nun den Propagator berechnen, der der Pulssequenz entspricht. Durch einen Vergleich des durch die aktuelle Pulssequenz dargestellten Propagators mit dem Zielpropagator kann man nun ein Maß dafür berechnen, wie man die Pulssequenz verändern muß, um eine bessere Übereinstimmung zwischen dem aktuellen und angestrebten Propagator zu erreichen.

  Den zu erreichenden Propagator definiert man mittels der Methode `setTargetU`. Die Routine erwartet die diesen Propagator beschreibende Matrix als Parameter.

11.4.3 Beispiel-Programm

Um die Verwendung dieser Klasse mit einem kurzen Beispiel zu verdeutlichen, soll ein Optimierungsprogramm für einen Trivial-Fall entwickelt werden. Ziel der Optimierung soll es sein, für ein 1-Spinsystem eine Pulssequenz zu berechnen, die einen $\pi/2$-Puls mit der Phase x implementiert.

```
: #include "spinsys.h"
1 #include "oct_u.h"
:
: int main( void )
: {
2    SpinSys mySys(1);
:
3    OCT_u myOCT(1, 200, 1e6, 1);
:
```

```
:    const complex<double> alpha ( 0.5*M_PI, 0);
4    myOCT.setTargetU( expm(-_i*alpha * mySys.Ix(1) ) );
:
5    myOCT.addCoil( _2pi* ( mySys.Ix(1) ),
:
6    myOCT.optimizeCG();
: }
```

In Position 1 wird zunächst der kontroll-theoretische Teil der Bibliothek eingebunden, mit dem die Behandlung dieses Problems möglich ist. In Position 2 wird das 1-Spinsystem deklariert, was hier ausschließlich dazu dient, Spinmatrizen korrekter Größe zur Hand zu haben. In Position 4 wird der einem $(\pi/2)_x$-Puls entsprechende Propagator als Optimierungsziel an das Objekt übergeben. Um eine solche Transformation zu implementieren, benötigt man nur ein Kontrollfeld, welches in Position 5 als logischer RF-Kanal mit einer festen Phase definiert wird. Schließlich wird in Position 6 die Optimierung gestartet.

11.5 Optimierungs-Methoden

Nachdem man das Optimierungs-Problem durch seinen Parameter-Satz beschrieben hat, reicht es aus, die numerische Optimierung durch den Aufruf einer Optimierungs-Methode zu starten. In den Beispiel-Programmen aus den Abschnitten 11.3.3 und 11.4.3 wurde in Position 8 bzw. 6 bereits die Optimierung unter Verwendung der Methode `optimizeCG` aufgenommen ohne jedoch näher darauf einzugehen. OCTANE stellt zwei Gradienten-Verfahren zur Verfügung, die zur iterativen Optimierung der Pulssequenz verwendet werden können.

11.5.1 Die Methoden

Generell erwarten die Methoden den Wert der für die Optimierung erwünschten Konvergenz-Schranke als Parameter; wird der Wert ausgelassen, so wird eine Schranke von $\varepsilon = 10^{-7}$ angenommen. Unter der Konvergenz-Schranke versteht man den Wert, den in der N-ten Iteration die Differenz der Qualitäts-Faktoren $QF(N) - QF(N-1)$ unterschreiten muß, damit die Opti-

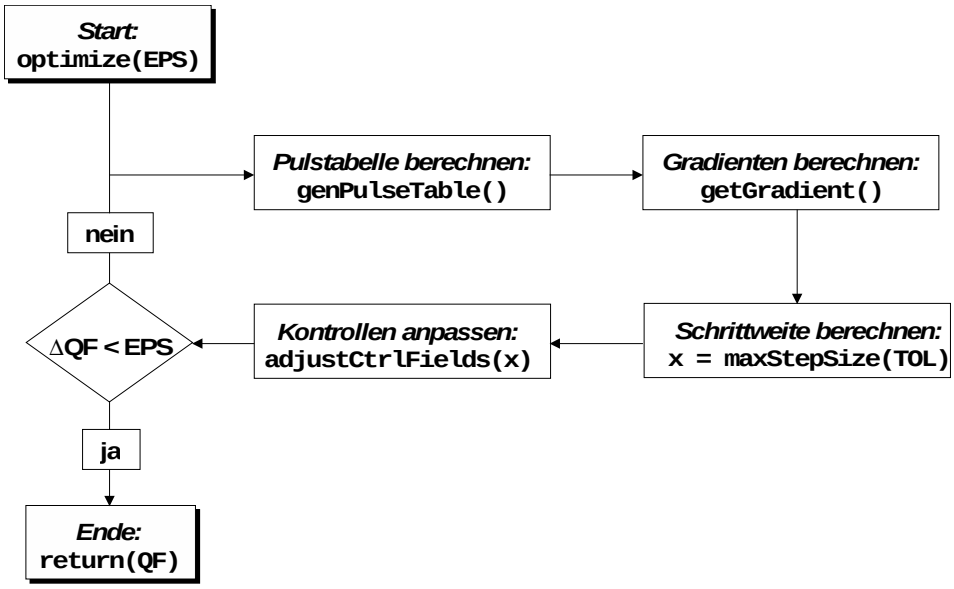

ABBILDUNG 11.3: Flußdiagramm des generellen Verfahrens zur Optimierung. Von der aufrufenden Routine werden zwei Grenzwerte an die Optimierung übergeben. `EPS` ist die absolute Konvergenzgrenze der Optimierung, `TOL` entspricht der zu erreichenden Grenze zur Bestimmung des Linienmaximums.

mierung als konvergiert betrachtet werden kann. Im Falle des minimal definierten Optimierungs-Problems wird in OCTANE das Skalarprodukt aus End-Dichteoperator und Ziel-Dichteoperator als Qualitätsfaktor angenomen. Schließt man jedoch weitere Effekte in das Optimierungs-Problem ein, so ändert sich die Defintion des Qualitätsfaktors (s. dazu Abschnitt 11.6).

Der zweite optionale Parameter der Optimierungs-Routinen entspricht der Toleranz-Grenze zur Bestimmung des Linienmaximums. Nachdem die Gradienten berechnet wurden, wird die maximale Schrittweite in Richtung des Gradienten bestimmt, durch die der Qualitätsfaktor ein Maximum erreicht. Dies entspricht dem Optimierungsproblem $\max\left\{QF(u_n + \kappa\frac{\partial \mathsf{H}}{\partial u_n})\right\}$. Wegen der eindimensionalen Charakteristik bezeichnet man diese Optimierung als Linienmaximierung. Das Linienmaximum gilt als bestimmt, wenn die Grenze $QF(\kappa) - QF(\kappa')$ die definierte Toleranz-Grenze unterschreitet. Je höher man die Anforderungen an die Bestimmung der maximalen Schrittweite stellt, desto höher ist der Rechenaufwand dieser Unterroutine.

In Abbildung 11.3 ist der prinzipielle Ablauf einer Optimierung mittels eines Flußdiagrammes gezeigt. Bevor jedoch die iterative Berechnung be-

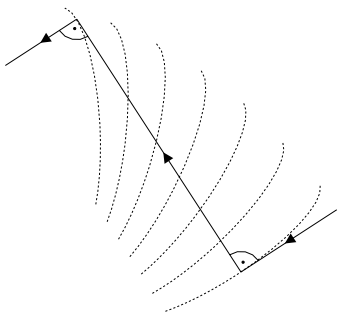

ABBILDUNG 11.4: Illustration der Ineffizienz von Algorithmen, die die stäkrste Steigung zur Optimierung benutzen. Ausgehend von einem Punkt auf einer Hyperfläche entspricht die Richtung der größten Steigung stets der Orthogonalen bezüglich der Kontourlinie. Folgt man diesem Gradienten so lange, bis ein Linienextremum erreicht ist, so endet dieser Schritt als Parallele zu einer Kontourlinie. Die neue Richtung wird wiederum die Orthogonale bezüglich dieser Kontourlinie sein. Je näher dieses Verfahren dem Extremum kommt, desto langsamer wird die Konvergenz, weil dieses nur durch eine infinitesimal kleine Schrittweite erreicht werden kann.

ginnt, werden alle statischen Operatoranteile in Tabellen gespeichert ("Drift Tables"). Da die Kontrollfelder nur innerhalb einer Iteration konstant sind, müssen aus ihnen in jedem Schritt die entsprechenden Tabellen berechnet werden ("Pulse Tables"). Nachdem nun alle Tabellen angepaßt sind, können die Gradienten für jeden Punkt der Kontrollfelder berechnet werden. Im Anschluß wird die Schrittweite bestimmt, welche die größte Verbesserung in Richtung der Gradienten ergibt. Hierbei wird die Schrittweite bis auf die durch den Parameter `TOL` vorgegeben Präzision bestimmt. Nachdem sowohl die Richtung der Veränderung als auch die Größe der Veränderung bekannt sind, können die Kontrollfelder dementsprechend angepäßt werden. Unterschreitet die Differenz der letzten beiden Qualitätsfaktoren die als `EPS` übergebende Konvergenzgrenze, so ist wird der Algorithmus als konvergiert angenommen und die Iteration abgebrochen.

Die zur Verfügung stehenden Methoden unterscheiden sich in der Berechnung der Gradienten, der prinzipielle Ablauf der Optimierung ist jedoch identisch. Im folgenden sollen beide Methoden kurz besprochen werden.

Methode der größten Steigung (*Steepest Descent*)

Dies ist die einfachste Methode eines Gradienten-Verfahrens und zeigt demzufolge auch ein schlechtes Konvergenz-Verhalten (s. dazu Abschnitt 11.5.2). Hierbei werden die nach dem Maximum-Prinzip berechneten Gradienten direkt verwendet, d.h. die eingeführten Veränderungen der Kontrollen folgen direkt der größten Steigung.

Das schlechte Konvergenzverhalten dieser Methode ist in Abbildung 11.4 verdeutlicht. Hier ist eine Vergrößerung einer zweidimensioneln Optimierung gezeigt. Die Suchrichtung entlang der größten Steigung ist stets orthogonal zur Kontourlinie. Berechnet man nun die optimale Schrittweite entlang dieser Richtung, so endet sie dort, wo sie die Parallele zu einer Kontourlinie ist. Bestimmt man nun dort wiederum die größten Steigung, so ist diese wiederum orthogonal zur vorgehenden Suchrichtung. In der Nähe eines Extremums geht hierbei die Schrittweite gegen Null, ohne jedoch genau das Extremum zu treffen, woraus eine unendlich langsame Konvergenzgeschwindigkeit resuliert. Ist die Entfernung vom Extremum jedoch groß, so liefert diese Methode befriedigende Resultate.

Um diese Optimierungs-Methode zu verwenden, sind in OCTANE drei überladene Variante der Methode `optimizeSD` in der Basisklasse `OCT_base` implementiert. In jedem Falle entspricht der Rückgabewerte der Methode dem erreichten Qualitätsfaktor.

- `double optimizeSD( double, double )`:
  Diese Variante verwendet die in der Reihenfolge `EPS, TOL` übergebenen Werte als Schranken für die Optimierung. Hierbei entspricht `EPS` der Konvergenzschranke bezüglich des Qualitätsfaktors und `TOL` der Toleranz bei der Isolierung des Linienmaximums. Beide Werte haben einen direkten Einfluß auf die benötigte Rechenzeit, je niedriger diese Schwellenwerte gewählt werden, desto länger verbringt die Optimierung in den jeweiligen Unterroutinen.

- `double optimizeSD( double )`:
  Übergibt man nur einen Wert an die Methode, so entspricht dieser der Konvergenzschranke des Qualitätsfaktors `EPS`. Der Schwellenwert für die Berechnung der optimalen Schrittweite entspricht hierbei dem Wert `TOL` $= 2 \cdot 10^{-4}$.

- `double optimizeSD( void )`:
  Wird die Optimierungsmehtode parameterfrei aufgerufen, so werden für beide Schranken die Werte `EPS` $= 10^{-7}$ und `TOL` $= 2 \cdot 10^{-4}$ vorgegeben.

Methode der konjugierten Richtung (*Conjugate Gradient*)

Wie bei der Methode der größten Steigung erklärt, ist die Verwendung der direkten Gradienten durch schlechtes Konvergenz-Verhalten in der Nähe des Extremums ausgezeichnet. Daher ist die Berechnung eines Satzes von sog. konjugierten Richtungen aus diesen Gradienten günstiger [85]. Eine notwendige Bedingung zur Berechnung der konjugierten Richtung ist allerdings, daß man sich in einem Punkt auf dem Linienmaximum befindet. Somit wächst der Anspruch an die Genauigkeit der Bestimmung des Linienmaximums.

In der Regel sollte man dieser Methode aufgrund der besseren Konvergenz-Eigenschaften in der Nähe des Extremums den Vorzug geben. Allerdings sollte der Parameter `TOL` nicht größer als der Standardwert gewählt werden.

Die Methode wird in vierfach überladener Form von der Basisklasse bereitgestellt. Der Rückgabewert entspricht dem erreichten Qualitätsfaktor.

- `double optimizeCG( double, double, int )`:
  In diese Variante entsprechen die Parameter den Werten `EPS, TOL` und `CG`. Hierbei entspricht `EPS` der Konvergenzschranke bezüglich des Qualitätsfaktors und `TOL` der Toleranz bei der Isolierung des Linienmaximums. Beide Werte haben einen direkten Einfluß auf die benötigte Rechenzeit, je niedriger diese Schwellenwerte gewählt werden, desto länger verbringt die Optimierung in den jeweiligen Unterroutinen. Der dritte Parameter `CG` ist ein Flag zur Wahl der zu berechnenden konjugierten Richtungen. `CG = 0` bedeutet, daß die konjugierten Richtungen nach dem Kriterium von Pollak-Ribière berechnet werden. Wählt man `CG = 1`, so wird das Kriterium nach Fletcher-Reeves verwendet.

- `double optimizeCG( double, double )`:
  Übergibt man nur zwei Parameter an die Methode, so enstprechen diese den Schwellenwerten in der Reihenfolge `EPS` und `TOL`. Als Kriterium zur Berechnung der konjugierten Richtungen wird das Kriterium von Pollak-Ribière verwendet.

- `double optimizeCG( double )`:
  Übergibt man nur einen Wert an die Methode, so entspricht dieser der Konvergenzschranke des Qualitätsfaktors `EPS`. Der Schwellenwert für die Berechnung der optimalen Schrittweite entspricht hierbei dem Wert $\texttt{TOL} = 2{\cdot}10^{-4}$. Es werden die konjugierten Richtungen nach Pollak-Ribière berechnet.

- `double optimizeCG( void )`:
  Wird die Optimierungsmehtode parameterfrei aufgerufen, so werden für beide Schranken die Werte $\texttt{EPS} = 10^{-7}$ und $\texttt{TOL} = 2 \cdot 10^{-4}$ vorgegeben. Die konjugierten Richtungen werden gemäß dem Kriterium von Pollak-Ribière berechnet.

11.5.2 Vergleich der Methoden

Wie im voran gegangenen Abschnitt erläutert, stellt die gegenwärtige Version von OCTANE zwei Gradienten-Verfahren zur Optimierung bereit. In diesem Abschnitt soll kurz auf die Unterschiede der beiden Verfahren eingegangen werden.

Beide Verfahren benutzen Gradienten zur Bestimmung der optimalen Veränderung der Pulsform. Die Methode `optimizeSD` verwendet die nach dem Maximum-Prinzip berechneten Gradienten direkt, d.h. sie folgt der größten Steigung (engl. “Steepest Descent”). Der große Nachteil dieses Verfahrens besteht in dem Verhalten in der Nähe des zu bestimmenden Extremums. Je näher der Alghoritmus gegen das Extremum konvergiert, desto langsamer ist der Gesamt-Fluß. Dies liegt daran, daß die bestimmten Richtungen orthogonal zu einander liegen werden, die optimale Schrittweite sich dabei jedoch stark reduziert.

Diesen Hauptnachteil des direkten Gradienten-Verfahrens kann man durch die Verwendung von konjugierten Richtungen ausgleichen (engl. “Conjugate Gradients”). Durch die Bestimmung der konjugierten Richtungen ist die Gesamt-Bewegung des Systems in Richtung des Extremums asymptotisch, d.h. die Fluß-Geschwindigkeit wird nicht durch orthogonale Bewegungen verlangsamt. Dies führt zum einen dazu, daß die Flußgeschwindigkeit in Richtung des Extremums erhöht wird, d.h. die Verwendung der konjugierten Richtung wird im Vergleich zum Steepest Descent Verfahren bereits früher gegen

das absolute Extremum konvergieren. Zum anderen erhöht die Verwendung der konjugierten Richtung die Geschwindigkeit der Konvergenz am Extremum.

Dieser Befund soll anhand eines einfachen Optimierungs-Besipiels dargestellt werden. In einem 2-Spin System soll der Term $I_z(1)$ in $I_z(2)$ überführt werden. Die theoretisch minimale Zeit für den vollständigen Transfer $\eta = 1$ unter Verwednung der schwachen Kopplung beträgt $1/J$, welche somit als Zeit τ_p für die zu optimierende Pulsform vorgegeben wurde. Damit beide Verfahren mit einander verglichen werden können, wird eine einheitliche Startform der RF-Kontrollen vorgegeben.

In Abb. 11.5 kommt das vorausgesagte Verhalten klar zum Ausdruck. Hier werden die erreichten Transfer-Amplituden in Abhängigkeit der Konvergenz-Schranke ε gezeigt. Zum einen erkennt man, daß die Methode der konjugierten Richtungen für die schwächste Konvergenz-Grenze bereits einen Wert viel näher am theoretischen Maximum erreicht hat. Darüberhinaus führt die Verstärkung des Abbruchkriteriums zu einer stärker assymptotischen Annäherung an das Transfer-Maximum als bei der Methode der größten Steigung. Durch Verwendung der konjugierten Gradienten kann man also bereits durch ein schwaches Abbruchkriterium relativ gute Ergebnisse erzielen. Die Methode der stärksten Steigung konvergiert zwar ebenfalls gegen das Extremum, jedoch muß das Abbruchkriterium sehr streng gewählt werden; in dem angeführten Beispiel ist hierbei ein Abbruchkriterium $\varepsilon = 10^{-7}$ nötig, um vergleichbare Ergebnisse wie die Methode der konjugierten Richtungen für ein Abbruchkriterium $\varepsilon = 10^{-5}$ zu erreichen.

Dieses Konvergenzverhalten hat einen direkten Einfluß auf die Anzahl der benötigten Rechenoperationen und damit auf die für eine Optimierung benötigte Rechenzeit. In Abb. 11.6 sind die Anzahl der benötigten Iterationen in Abhängigkeit des Abbruch-Kriteriums ε aufgeführt. Betrachtet man zunachst das generelle Verhalten beider Verfahren, so stellt man fest, daß die Anzahl der Operationen (und damit auch die benötigte Rechenzeit) des Verfahrens der größten Steigung in Abhängigkeit der Konvergenz-Schranke stark nicht-linear ansteigt. Das Verfahren der konjugierten Richtung zeigt hingegen einen vergleichweise vernachlässigbaren Anstieg des Aufwandes über den selben Bereich der Konvergenz-Schranke. Betrachtet man nun den bereits oben erwähnten Fall ähnlicher Konvergenz beider Verfahren (Steepest Descent:

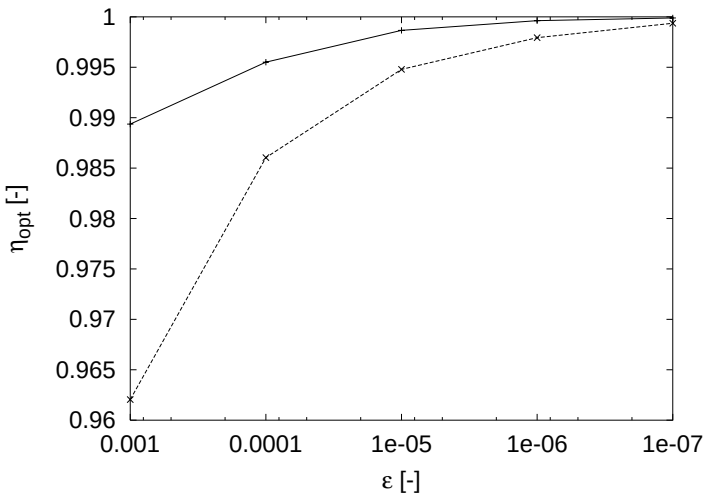

ABBILDUNG 11.5: Vergleich der Konvergenz der beiden Methoden `optimizeSD` (Steepest Descent, gestrichelt) und `optimizeCG` (Conjugate Gradients, durchgezogen). Gezeigt sind die Werte für die Transfer-Amplitude η in Abhämgigkeit der Konvergenz-Schranke ε. Man erkennt, daß die Steepest Descent Methode deutlich langsamer gegen das Optimum konvergiert.

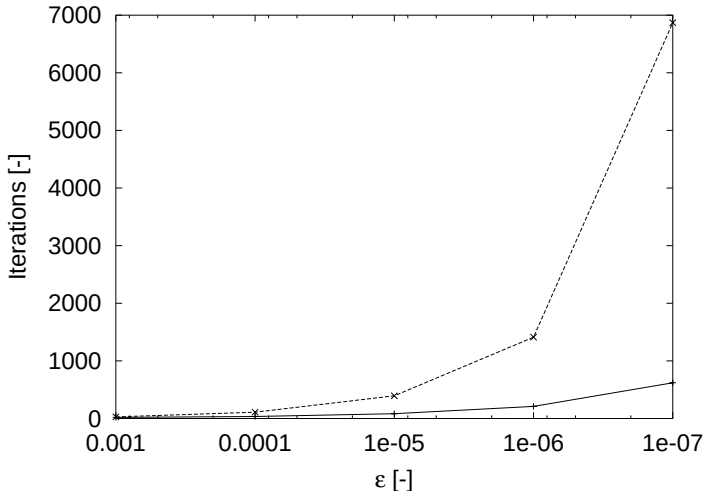

ABBILDUNG 11.6: Vergleich des benötgiten Rechenaufwandes `optimizeSD` (Methode der größten Steigung, gestrichelt) und `optimizeCG` (Methode der konjugierten Richtung, durchgezogen). Dargestellt wird die Anzahl der benötigten Iterationen in Abhängigkeit der Konvergenz-Schranke ε. Es wird deutlich, daß die Methode der größten Steigung aufgrund des schlechteren Konvergenz-Verhaltens auf strengere Abbruch-Bedingungen mit einem starken Anstieg der Rechenoperationen reagiert.

$\eta(\varepsilon = 10^{-7}$, Conjugate Gradients: $\eta(\varepsilon = 10^{-5})$, so muß man feststellen, daß die numerischen Kosten zur Erreichung von Ergebnisse ähnlicher Güte stark unterschiedlich sind. Das Steepest Descent Verfahren benötigt in diesem Falle den 80fachen numerischen Aufwand des Conjugate Gradients Verfahren.

Die Ergebnisse dieses Vergleiches bestätigen in vollem Umfang das aus den mathematischen Prinzipien der Verfahren vorhergesagte Verhalten; das Verfahren der konjugierten Richtung ist dem der größten Steigung eindeutig überlegen und sollte in Optimierungen prinzipiell vorgezogen werden. Darüberhinaus bleibt festzustellen, daß die Verwendung von konjugierten Richtungen die Ansprüche an die verwendete Konvergenz-Schranke erheblich absenkt. Somit können Optimierungen unter Verwendung dieser Methode erheblich schneller zu verwertbaren Pulsformen gelangen. Bedenkt man vor allem, daß der Einschluß weiterer Effekte (breite Anregungsprofile, B_1-Inhomogenitäten, etc.) den numerischen Aufwand durch die Erhöhung der Problem-Komplexität stark (und unabhängig von der verwendeten Gradienten-Methode) ansteigen läßt, so drängt sich die Verwendung geschickter Gradienten-Verfahren geradezu auf.

11.5.3 Kombination von Optimierungs-Methoden

In den vorigen Abschnitten wurde auf die Unterschiede zwischen den Optimierungsmethoden hingewiesen. Aus dem dort Gesagten kann man zwei Fakten festhalten. Zum einen werden beide Verfahren bei großer Entfernung vom Extremum in etwa ähnlich schnell konvergieren, lediglich die Konvergenzgeschwindigkeit in der Nähe des Optimums legt die Verwendung der Methode der konjugierten Richtungen nahe. Andererseits ist man hierbei bei der Wahl des Parameters `TOL` eher festgelegt, bei der Methode der größten Steigung kann man diesen Parameter jedoch frei wählen, weil hier die Erreichung des Linienextremums nicht notwendig ist. Dieser Parameter bestimmt jedoch teilweise die Rechenzeit.

Daher liegt es nahe, Optimierungs-Protokolle zu entwickeln, die beide Verfahren kombinieren. Zu Beginn (man befindet sich sicherlich noch nicht in der Nähe des Optimums) könnte man die Methode der größten Steigung mit einer großzügigen Toleranz `TOL` der Bestimmung des Linienmaximums einsetzen, um sich rasch in die Nähe des Optimums zu begeben, d.h. für eine relativ große Konvergenz-Schranke `EPS`. Befindet man sich an diesem

Punkt, so könnte man nun zur Berechnung der Gradienten die Methode der konjugierten Richtungen verwenden, d.h. man tauscht den Gewinn der schnelleren Linienmaximierung der Methode der größten Steigung gegen eine höhere Konvergenz-Geschwindigkeit der Methode der konjugierten Richtung.

In den Tabellen 11.2 und 11.3 sind Daten zusammengefaßt, die diese Idee illustrieren. In allen Fällen wurde das selbe Problem mit OCTANE optimiert. Die ersten vier Einträge illustrieren die Abhängigkeit vom Parameter der Linienmaximierung `TOL`. Wie man Tabelle 11.2 entnehmen kann, ist der Einfluß auf den insgesamt erreichten Qualitätsfaktor vernachlässigbar. Jedoch der Bedarf an Rechenzeit steigt mit abnehmender Toleranz während der Bestimung des Linienmaximums (s. Tabelle 11.3). Verbindet man nun die Vorteile beider Methoden, so kann man bei Erreichung eines vergleichbaren Qualitätsfaktors einen Rechenzeitgewinn von über 25% erreichen. So kann man der abschließenden Bemerkung aus Abschnitt 11.5.2 nur hinzufügen, daß eine geschickte Kombination der zur Verfügung stehenden Methoden nicht minder erstrebenswert ist.

Diese Kombintationen sind vom jeweiligen Optimierungsproblem abhängig, daher stellt OCTANE diese nicht als fertig implementierte Optimierungsprotokolle zur Verfügung. Es bleibt dem Anwender überlassen, diese in seinem Optimierungsprogramm zu implementieren.

11.6 Optionale Optimierungs-Routinen

Nachdem in den Abschnitten 11.3 und 11.4 beschrieben wurde, wie man ein minimal definiertes Optimierungs-Problem in OCTANE beschreibt, soll in diesem Abschnitt zunächst auf die Routinen der Bibliothek eingegangen werden, welche die Komplexität des Optimierungs-Problem heraufsetzen. Hierbei handelt es um Routinen, die bestimmte experimentelle Effekte berücksichtigen. Darüberhinaus werden noch einige Routinen aufgeführt, die sich mit der Verwaltung eines Optimierungs-Projektes befassen.

11.6.1 Erweiterung des Optimierungs-Problems

Die durch ein minimal definiertes Optimierungs-Problem zu erreichenden Ergebnisse sind prinzipieller Natur, weil dabei bewußt experimentelle Effekte außer acht gelassen werden. Die so erlangten Daten dienen in der Regel dazu,

Methode	EPS	TOL	QF	QF/QF_{max}
optimizeCG	10^{-4}	10^{-1}	0,990704	0,9963
optimizeCG	10^{-4}	10^{-2}	0,994196	0,9998
optimizeCG	10^{-4}	10^{-3}	0,994277	0,9999
optimizeCG	10^{-4}	10^{-4}	0,994069	0,9997
optimizeSD	10^{-4}	10^{-1}	0,990704	0,9963
optimizeCG	10^{-4}	10^{-4}		
optimizeSD	10^{-3}	10^{-1}	0,994392	1,0000
optimizeCG	10^{-4}	10^{-4}		

TABELLE 11.2: Erreichte Qualitätsfaktoren QF in Abhängigkeit von den Optimierungsparametern. Die ersten vier Optimierungen variieren den Parameter TOL bei konstantem EPS bei Verwendung der Methode der konjugierten Richtung. Die letzten beiden Optimierungen kombinieren die Methoden der größten Steigung mit der der konjugierten Richtung.

Methode	EPS	TOL	CPU [sec]	CPU/CPU_{max}	Iterationen
optimizeCG	10^{-4}	10^{-1}	115,270	0,7274	98
optimizeCG	10^{-4}	10^{-2}	148.010	0,9306	113
optimizeCG	10^{-4}	10^{-3}	153.990	0,9682	115
optimizeCG	10^{-4}	10^{-4}	154.360	0,9705	113
optimizeSD	10^{-4}	10^{-1}	116.390	0,7318	99 (98, 1)
optimizeCG	10^{-4}	10^{-4}			
optimizeSD	10^{-3}	10^{-1}	159.050	1,0000	118 (2, 116)
optimizeCG	10^{-4}	10^{-4}			

TABELLE 11.3: Benötigte Rechenzeit CPU in Abhängigkeit von den Optimierungsparametern. Die ersten vier Optimierungen variieren den Parameter TOL bei konstantem EPS bei Verwendung der konjugierten Gradienten Methode. Die letzten beiden Optimierungen kombinieren die Methoden der größten Steigung mit der der konjugierten Gradienten. Hierbei beziehen sich angegeben Daten auf die Kombination der Methoden. Die bei den Iterationen in Klammern angegebenen Zahlen beziehen sich auf die verwendeten Methoden im einzelnen.

die theoretische Grenze von Kohärenz-Transfers zu bestimmen, daher werden die so entwickelten Pulssequenzen in der Praxis nur bedingt einsetztbar sein. In diesem Abschnitt werden Routinen vorgestellt, die solche experimentellen Faktoren in die Optimierung einfließen lassen können.

Jede dieser Routinen kann optional aufgerufen werden, um die Problem-Komplexität dem realen Optimierungsziel anzupassen. Hierbei gilt es zu beachten, daß alle diese Effekte während der Optimierung kreuzkorreliert berücksichtigt werden. Fügt man mehrere Effekte der Optimierung hinzu, so steigt der numerische Aufwand nicht linear, sondern exponentiell. Darüber hinaus verfügen die meisten dieser Effekte über einen Digitalisierungs-Parameter, der für jeden dieser Effekte den Rechenaufwand zusätzlich beeinflußt. Somit ist es für den Anwender von Bedeutung, die Parameter für sein spezielles Problem so zu ermitteln, daß der numerische Aufwand in akzeptablen Grenzen bleibt.

Optionale Methoden der Basisklasse

Die optionalen Methoden der Basisklasse werden von allen abgeleiteten Klassen ererbt und stehen somit dort zur Verfügung. Daher werden diese getrennt von den optionalen Methoden der abgeleiten Klassen besprochen.

- `void setB1errGauss( int, int, double )`:
  Eine wichtige Eigenschaft realer Spulen ist, daß die abgestrahlten RF-Felder fehlerbehaftet sind. In der Regel werden die Flip-Winkel α_{real} über einen bestimmten Bereich $[\alpha_{min}, \alpha_{max}]$ gestreut, d.h. die endgültige Lage des makroskopischen Magentisierungs-Verktors wird nicht durch eine Richtung bestimmt, sondern einen bestimmten von der verwendeten Spule abhängigen Winkelbereich abdecken (vergl. Abbildung 11.7). Diese Eigenschaft kann man durch eine Fehlerverteilung modelieren, in der Regel wird hierzu eine Verteilungsfunktion nach Gauß angenommen.

  Mit der Routine `setB1errGauss` kann man eine solche Verteilungsfunktion für jeden mit `addCoil` definierten logischen RF-Kanal einschließen. Hierbei gilt zu beachten, daß während der Berechnung jedes Gradienten diese Fehlerverteilungen korreliert berücksichtigt werden, d.h. je mehr Fehlerverteilungen in die Optimierung aufgenommen werden, de-

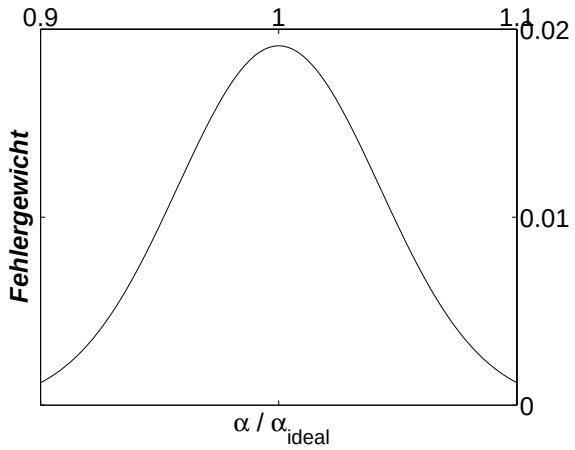

ABBILDUNG 11.7: RF-Fehlerverteilung einer Spule, berechnet mit einer Digitalsierung von 101. Die Abszisse entspricht dem im Verhältnis zum angestrebten Pulswinkels α_{ideal} tatsächlich erreichten. Durch die Ordinate wird der Anteil der mit dem jeweiligen Winkel bepulsten Einheiten dargestellt. Die Summe der Fehlergewichte beträgt 1.

sto stärker steigt die benötigte Rechenzeit an. Darüberhinaus beeinflußt auch die Digitalisierung den Bedarf an Rechenzeit. Je feiner digitalisiert die Fehlerverteilungen werden, desto stärker steigt die benötigte Rechenzeit an, insbesondere wenn mehrere Fehlerverteilungen berücksichtigt werden. Die Definition des Qualitätsfaktors wird durch Verwendung dieser Methode nicht verändert.

Der erste Parameter dieser Funktion vom Typ `int` enspricht dem Index des logischen RF-Kanals, dessen RF-Inhomogenität berücksichtigt werden soll. Dieser Index wird als Rückgabewert von `addCoil` bereitsgetellt. Der zweite Parameter vom Typ `int` definiert die Digitalisierung der Fehlerfunktion und sollte ungerade geweählt werden, damit das Maximum der Verteilungsfunktion während der Rechnung berücksichtigt werden kann. Der dritte Parameter vom Typ `double` entspricht der halben Breite bei halber Höhe der Gauß'schen Verteilungsfunktion und bestimmt die maximale Abweichung vom idealen Pulswinkel.

- `void limitRF(int, double )`:
  Führt man die Berücksichtung von Resonanzfrequenz-Offsets in das Optimierungs-Problem ein, so wird in der Regel die Feldstärke der be-

rechneten RF-Felder in der Größenordung des Offset-Bereiches liegen. Unter Umständen kann es passieren, daß die durch die Optimierung bestimmte Feldstärke der Pulssequenz die vom Verstärker mögliche übersteigt. Daher ist in diesen Fällen wichtig, die über eine Spule einzustrahlende Feldstärke bereits während der Berechnung der Sequenz auf ein gültiges Limit zu begrenzen.

Als ersten Parameter vom Typ `int` erwartet diese Methode den Index des logischen RF-Kanals, dessen Feldstärke auf ein Maximum begrenzt werden soll. Dieser Index wird von der Methode `addCoil` als Rückgabewert bereitgestellt. Der zweite Parameter vom Typ `double` entspricht der maximal erlaubten RF-Amplitude in Hertz.

Optionale Methoden der Klasse `OCT_rho`

Bildet ein Spinsystem die Grundlage der Optimierung, so kann man unter Verwendung der optionalen Methoden dieser Klasse das Verhalten des Spinsystems während der Dauer der Pulssequenz verfeinern. Folgende Methoden werden von der abgeleiteten Klasse dazu bereitgestellt:

- `void setDrift( Matrix<complex<double> > )`:
  Eine wichtige Eigenschaft von Spinsystemen ist es, daß sich Kohärenz aufgrund der Eigenschaften des Systems in ihm verteilen kann. Diese Methode der Basisklasse definiert den während der Optimierung zu verwendenden freien Evolutionsoperator. Im Rahmen von OCTANE wird dieser Operator als statisch betrachtet, weil er nicht durch Kontroll-Felder verändert wird, sondern während der gesamten Puls-Sequenz vollständig unmoduliert aktiv ist. Er entspricht dem in der Kontroll-Theorie als "Drift Hamitonian" bezeichneten Operator, der den intrinsischen Fluß des Systems definiert. In der NMR-Spektrospokpie entspricht dies in der Regel dem Kopplungs-Hamiltonian. Besitzen die Kerne des verwendeten Spinsystems unterschiedliche Resonanz-Frequenzen, so können die entsprechenden Offset-Hamiltonians dem statischen Hamilton-Operator hinzugefügt werden. Innerhalb von OCTANE wird somit ein Operator

$$\mathcal{H}_{stat} = \mathcal{H}_J + \mathcal{H}_{\text{off}} \tag{11.8}$$

als statischer Hamiltonian betrachtet. Z.B. kann der Term $\mathcal{H}_J$ dem Fall der schwachen skalaren Kopplung (z.B. für heteronukleare Experimente) entsprechen. Die Angabe von Offset-Hamiltonians ist möglich, aber nicht zwingend. Werden hier allerdings Offset-Terme definiert, so werden nur diese speziellen Offset-Freqeunzen während der Pulsform-Berechnung berücksichtigt; man erhält auf diese Art keine generell breitbandigen Sequenzen (s. auch Abschnitt 11.6 zur Erweiterung des Optimierungsprogrammes um diese Forderung). Generell ist die Definition des Drift-Hamiltonians jedoch vom jeweiligen Optimierungs-Problem abhängig, z.B. ist im Fall eines 1-Spinsystems kein Drift-Hamiltonian notwendig.

- `void readRedfield( char*, char* )`:
  Ein für reale Spinsysteme wichtiger Effekt ist ihr Relaxations-Verhalten (vergl. Kapitel 3). Mittels WBR-Theory kann man eine Redfield-Matrix berechnen, die die Relaxationsparameter enthält. Hierbei liegt es am Anwender, welche Relaxations-Mechanismen er in der Optimierung einschließen möchte. Die entsprechenden Matrizen lassen sich z.B. mit dem Computer-Programm `wtest` [87] berechnen und als Datei sichern[5]. In Matlab-Dateien wird jede Matrix unter einem symbolischen Namen abgelegt, der vom Anwender z.B. in `wtest` gewählt wurde. Die Routine `readRedfield` erwartet als ersten Parameter den Dateinamen als String und den symbolischen Namen der Redfield-Matrix als zweiten Parameter ebenfalls als String.

  Wird eine Redfield-Matrix eingelesen, werden alle Operatoren in ihre entsprechenden Konstrukte des Liouville-Raumes transformiert. Hierbei gilt zu beachten, daß die Dimension des Liouville-Raumes der quadratischen des Hilbert-Raumes entspricht, entsprechend sind die Matrizen für ein N-Spinsystem nun $(4^N \times 4^N)$, was sich natürlich in einem Anstieg der Rechenzeit manifestiert. Neben der Raum-Transformation werden auch alle Evolutions-Routinen dem Liouville-Raum angepaßt.

[5] OCTANE kann derzeit nur das Little Endian Level 1.0 Format von Matlab lesen. `wtest` kann allerdings nur auf Computern der Firma *sgi* betrieben werden (*Big Endian*). Möchte man also ein mit `wtest` berechnete Matrix in OCTANE verwenden, so muß man die Datei auf einem *Little-Endian* System (z.B. i386-kompatible Systeme) mit Matlab öffnen und unter Verwendung des Befehls `save [MATRIX] [FILENAME] -V4` exportieren. Erst dann vermag OCTANE die Redfield-Matrix korrekt zu importieren

Natürlich kann man pro Spinsystem stets nur eine einzige Redfield-Matrix verwenden. Das Hinzufügen von Relaxations-Effekten hat neben den eben beschriebenen keine weiteren Effekte auf die Optimierung. Prinzipiell wird nur das Drift-Verhalten des Spinsystems modifziert.

- `void setRedfield( Matrix<complex<double> > )`:
  Neben der Möglichkeit, eine Relaxationsmatrix aus einer Datei zu lesen, besteht die Möglichkeit, eine solche Matrix in einer Variablen zur Laufzeit zu erzeugen. Möchte man diese an OCTANE übergeben, so kann man dazu die Methode `setRedfield` verwenden. Sie erwartet die Redfield-Matrix als Parameter. Die anschließende interne Verwendung einer mit dieser Methode an OCTANE übergebenen Redfield-Matrix ist analog zu `readRedfield` (s. dort).

- `int addOffset( Matrix<complex<double>>,double,double,int )`:
  Ein wichtiges Qualitätsmerkmal einer Pulssequenz ist ihre Breitbandigkeit, d.h. wie groß darf die Differenz der Resonanzfrequenz eines Kernes zu der Einstrahlfrequenz des RF-Feldes sein, ohne daß sich das Transfer-Verhalten ändert. Im mininmal definierten Optimierungs-Problem wurde auf den Einschluß solcher Effekte gänzlich verzichtet; die dort berechneten Sequenzen haben in der Regel eine Bandbreite weniger Hertz. Möchte man allerdings eine Pulssequenz berechnen, deren Bandbreite einige Kilohertz beträgt, so muß man diese Forderung mit in die Formulierung des Optimierungs-Problems aufnehmen. Hierzu stellt OCTANE die Routine `addOffsetHamiltonian` zur Verfügung.

  Als Parameter erwartet die Routine (1) den Offset-Hamiltonian $\mathcal{H}_{\text{off}} = 2\pi I_{n,z}$, (2) die niedrigste zu berücksichtigende Offset-Frequenz in Hertz, (3) die größte zu berücksichtigende Offset-Frequenz in Hertz und (4) die Digitalisierung dieses Offset-Bereiches. Hierbei ist eine gute Wahl der Digitalisierung dem Anwender überlassen. Wählt man diesen Wert zu niedrig, so vermag OCTANE zwar den Transfer an diesen Stützpunkten zu optimieren, jedoch kann es passieren, daß die Transfer-Amplituden in den Frequenz-Bereichen zwischen den Digitalisierungs-Punkten des Offset-Bereiches stark abfallen. In der Regel gibt es eine minimale Digitalisierung, ab der auch die Zwischenbereiche in guter Qualität den geforderten Kohärenz-Transfer erfüllen.

Durch sequentielle Verwendung dieser Methode kann man Bereiche für mehrere Offset-Hamiltonians definieren. So macht es Sinn, in einem heteronuklearen Experiment den Offset-Bereich für alle RF-Kanäle möglichst breit zu optimieren. Werden mehrere Offset-Terme in die Optimierung eingeführt, so wird eine entsprechend mehrdimensionale Stützmatrix während der Optimierung verwendet, deren Größe von den gewählten Offset-Parametern abhängt. Allgemein gilt, daß für k Offset-Bereiche O_k eine k-dimensionale Stützmatrix verwendet wird, deren Größe $\dim(O_1) \times \cdots \times \dim(O_k)$ beträgt. Nun werden für jedes Element dieser Stützmatrix Gradienten bestimmt, d.h. der Rechenaufwand steigt mit der Anzahl der Matrixelemente der Stützmatrix.

Durch die Einführung von Offset-Termen ändert sich die Definition des Qualitäts-Faktors. Er ist nun die Summe über die Skalar-Produkte zwischen End- und Ziel-Dichteoperator für alle Elemente der Stützmatrix und liegt somit innerhalb eines Intervalls von $[0, \dim(O_1) \times \cdots \times \dim(O_k)]$. Am Ende einer Optimierung wird der charkteristische Qualitätsfaktor der Sequenz ausgegeben, was dem arithmetischen Mittel des Gesamt-Qualitätsfaktors entspricht.

11.6.2 Optionale Routinen zur Projekt-Verwaltung

In diesem Abschnitt werden Routinen beschrieben, die zur Verwaltung von Optimierungs-Projekten genutzt werden können. Hierbei handelt es sich im speziellen um Reinitialisierungs- und I/O-Routinen, die von der Basisklasse bereitgestellt werden. Darüberhinaus bieten die Unterklassen noch Methoden zur Analyse von Pulssequenzen, die im OCTANE-Format vorliegen.

Optionale Methoden der Basisklasse

- `void reinitControlFields( void )`:
  Möchte man einen neuen Satz zufälliger Start-Kontrollfelder erzeugen, so kann man dies durch Aufruf dieser Routine. Hierbei werden die selben Parameter bei der normalen Initialisierung eines Kontroll-Feldes mit `addControlledHamiltonian` verwendet. Diese Routine wird in der Regel eingesetzt, um nach einer erfolgten Optimierung die optimierten Felder zu löschen und eine neue Optimierung zu starten, falls das

Level	Beschreibung
0	Nur Ausgabe der Optimierungs-Parameter und des erreichten Qualitätsfaktor am Ende der Optimierung
1	Zusätzliche Ausgabe des Qualitätsfaktors während jeder Iteration
2	Vollständige Ausgabe aller Informationen während jeder Iteration

TABELLE 11.4: OCTANE stellt verschiedene Level für die Ausgabe von Informationen während der Laufzeit des Programmes zur Vefügung. Je höher das Level gewählt wird, desto mehr Ausgaben generiert die Bibliothek auf der Standardausgabe.

Optimierungs-Objekt innerhalb des aktiven Gültigkeitsbereich (*Scope*) nicht durch Aufruf des Destruktors vollständig entfernt werden soll.

- `void LogLevel(int)`:
  Die in OCTANE implementierten Methoden sind darauf ausgelegt, den Benutzer über ihren Status zu informieren. Die hierbei möglichen Informationen werden stets auf der Standardausgabe ausgegeben, möchte man also eine Protokoll-Datei der Optimierung speichern, so bietet es sich an, die Standardausgabe in eine Datei umzulenken (s. dazu die üblichen UNIX-Dokumentationen). Da in der Regel während einer Optimierung eine große Menge solcher Informationen anfallen, kann man diese Methode verwenden, um den Umfang der Ausgabe zu steuern. Hierbei erwartet die Methode das erwünschte Ausgabe-Level als Parameter. In Tabelle 11.4 sind die dafür derzeit zur Verfügung stehenden Werte aufgeführt.

  Da es sich hierbei um einen optionalen Methodenaufruf handelt, wird der Ausgaben-Level 0 als Standard angeommen. Dieses entspricht einem rudimentärem Protokoll, d.h. es werden nur Daten während der Initialisierungs-Phase und nach Beendigung der Optimierung ausgegeben. Erst ab Level 1 werden zusätzlich Laufzeit-Informationen der Optimierung ausgegeben. Hierbei beschränkt Level 1 die Ausgabe auf den Qualitätsfaktor während jeder Iteration. Das höchste Level erlaubt allen Methoden alle verfügbaren Informationen auszugeben.

- `void enableCOS( char* )`:
  Ruft man diese Methode auf, so wird der Status der Optimierung in

jeder Iteration in einer Binärdatei gesichert, die dem Matlab Level 1.0 Format [90] entspricht.. Der Parameter, der an diese Methode übergeben wird, entspricht der Wurzel des Dateinamens, der vor der Speicherung gemäß "`[WURZEL]_[#].mat`" erzeugt wird. Hierbei entspricht `[#]` der Nummer der durchlaufenen Iteration. Die Datei "`[WURZEL]_[#-1].mat`" wird nach erfolgter Speicherung gelöscht.

- `void writeSet(char* )`:
  Hat man ein Optimierungs-Problem definiert, so kann man es mit dieser Methode in eine Datei im Binärformat schreiben. Den Namen der Datei wird der Routine als String übergeben. Das Binärformat entspricht einer Matlab Level 1.0 Datei [90], so daß man die Daten dort weiterverarbeiten kann. In der Regel wird diese Routine verwendet, um eine erfolgte Optimierung zu sichern.

  Neben einigen Flags enthält die Datei die RF-Felder, die Zeitachse der RF-Felder und den zugehörigen Hamilton-Operator unter den symbolischen Namen `u[#]`, `u[#]_t` und `H[#]`, wobei `[#]` eine ganze natürliche Zahl ist. `[#]` ergibt sich durch die Reihenfolge, in der die Kontrollterme hinzugefügt wurden. Wurde die Optimierung im Liouville-Raum durchgeführt, so entsprechen die Hamilton-Operatoren ihren Super-Hamiltonians. In diesem Falle befindet sich zusätzlich die verwendete Relaxationsmatrix unter dem symbolischen Namen `red` in der Projektdatei.

- `void readSet(char* )`:
  Diese Routine lädt ein vorher mit `writeControlSet` gespeichertes Projekt. Der Name der Projektdatei wird der Methode als String übergeben. Hierbei wird dieselbe Struktur logischer RF-Kanäle wiederhergestellt, wie sie vor der Speicherung definiert wurde. Allerdings werden eventuell zuvor gesetzte RF-Fehlerverteilungen und RF-Limits entfernt, d.h. lädt man ein Projekt, so werden wieder ideale RF-Kanäle angenommen. In der Regel erfolgt dies, um anstatt zufälliger Start-Kontrollfelder einen definierten Ausgangspunkt für die Optimierung zu haben.

Optionale Methoden der Klasse `OCT_rho`

Die optionalen Methode der Unterklasse dienen dazu, die Qualität des durch eine Pulssequenz erreichten Kohärenztransfers $\sigma_{\text{Start}} \rightarrow \sigma_{\text{Ziel}}$ in Abhängigkeit verschiedener Parameter ein- oder zweidimensional zu berechnen. Die Daten werden hierbei binär in einer Matlab-Datei gespeichert. Mit Matlab kann man dann sehr einfach daraus die entsprechenden Plots bzw. Kontour-Plots erstellen (vergl. dazu die entsprechende Matlab Dokumentation).

- `void evalSeq( char*, int, double, double, int, int, double, double, int )`:
  Diese Methode dient zur Bewertung einer Sequenz im OCTANE-Format in Abhängigkeit zweier Offset-Terme. Der erste Parameter entspricht dem Namen der Ausgabedatei. Dann folgen die Parameter der beiden zu berücksichtigenden Offset-Terme, die zuvor mit `addOffset` hinzugefügt wurden. Hierbei können allerdings die Offset-Bereiche und deren Digitalisierung neu gewählt werden. Die Parameterliste `int, double, double, int` (zweiter bis fünfter bzw, sechster bis neunter Parameter) entsprechen dem Index des Offset-Hamiltonians (wie von `addOffset` zurückgegeben), der niedrigsten Offset-Frequenz in Hertz, der höchsten Offset-Frequenz in Hertz und der Digitaliserung der Achse. Die Ausgabedatei enthält drei Matrizen. Hierbei entspricht die Matrix `TE` dem Qualitätsfaktor und somit der z-Achse des Plots. Die Matrizen `off0` und `off1` enthalten die entpsrechenden Offsetfrequenzen, die zur Darstellung der x- und y-Achse verwendet werden.

- `void evalSeq( char*, int, double, double, int )`:
  Diese Methode berechnet den Qualitätsfaktor der aktuellen Pulssequenz in Abhängigkeit eines Offset-Hamiltonians. Hierbei entspricht der erste Parameter dem Namen der Ausgabedatei. Die restliche Liste definiert den zu verwendenden Offset-Term durch seinen Index `int`, die niedrigste und höchste zu verwendende Offset-Frequenz `double, double` und die für die Bewertung zu verwendende Digitalisierung `int`. Die Ausgabedatei enthält zwei Matrizen. Hierbei entspricht die Matrix `TE` dem Qualitätsfaktor und somit der Ordinate des Plots. Die Matrix `off0` enthält die entpsrechenden Offsetfrequenzen, die zur Darstellung der Abszisse verwendet werden.

- `void evalSeq( char*, int, int, double, int, double, double, int )`:
  Diese Methode dient zur Berechnung des Qualitätsfaktors in Abhängigkeit eines Offset-Terms und der Fehlerverteilung eines logischen RF-Kanals. Hierbei wird der Name der Ausgabedatei als erster Parameter übergeben. Es folgt die zu verwendende Fehlerverteilung des logischen RF-Kanals mit dem Index `int` (Rückgabewert von `addCoil`). Die in `int` Punkte digitalisierte Fehlerverteilung beschreibt gewichtete Pulswinkel-Anteile $\alpha/\alpha_{\text{ideal}}$ in einem Intervall von `double`. Die letzten Parameter definieren den zu verwendenden Offset-Term durch seinen Index `int`, die niedrigste und höchste zu verwendende Offset-Frequenz `double, double` und die für die Bewertung zu verwendende Digitalisierung `int`. Die Ausgabedatei enthält drei Matrizen. Hierbei entspricht die Matrix `TE` dem Qualitätsfaktor und somit der z-Achse des Plots. Die Matrizen `B1err0` und `off0` enthalten die entpsrechende Fehlerverteilung und Offsetfrequenzen, die zur Darstellung der x- und y-Achse verwendet werden.

Optionale Methoden der Klasse `OCT_u`

Die optionalen Methode dieser Unterklasse dienen dazu, die Qualität des durch eine Pulssequenz implementierten Propagators mit dem als Optimierungsziel formulierten Propagators in Abhängigkeit verschiedener Parameter ein- oder zweidimensional zu berechnen. Die Daten werden hierbei binär in einer Matlab-Datei gespeichert. Mit Matlab kann man dann sehr einfach daraus die entsprechenden Plots bzw. Kontour-Plots erstellen.

- `void evalSeq( char*, int, int, double )`:
  Diese Methode wird dazu verwendet die Güte der Implementation eines Propagstors in Abhängigkeit der Fehlerverteilung eines logischen RF-Kanals zu berechnen. Die Daten werden in der durch den ersten Parameter spezifizierten Ausgabedatei gespeichert. Die Fehlerverteilung wird für den Kanal `int` angenommen, wobei diese in `int` Punkte digitalisiert wird und einen Winkelanteils-Bereich $\alpha/\alpha_{\text{ideal}}$ von `double` abdeckt. Die Ausgabedatei enthält zwei Matrizen. Hierbei entspricht die Matrix `TE` dem Qualitätsfaktor und somit der Ordinate des Plots.

Die Matrix `B1err0` enthält die entsprechende Fehlerverteilung, die zur Darstellung der Abszisse verwendet wird.

- `void evalSeq( char*, int, int, double, int, int, double )`: Diese Methode wird dazu verwendet die Güte der Implementation eines Propagstors in Abhängigkeit der Fehlerverteilung zweier logischer RF-Kanäle zu berechnen. Die Daten werden in der durch den ersten Parameter spezifizierten Ausgabedatei gespeichert. Es folgt die Defintion der ersten Fehlerverteilung. Diese beschreibt für den Kanal `int` einen in `int` Punkte digitalisierten Winkelanteils-Bereich $\alpha/\alpha_{\text{ideal}}$ von `double` vewendet. Die letzten drei Parameter definieren analog die zweite Fehlerverteilung. Die Ausgabedatei enthält drei Matrizen. Hierbei entspricht die Matrix `TE` dem Qualitätsfaktor und somit der z-Achse des Plots. Die Matrizen `B1err0` und `B1err1` enthalten die entpsrechenden Fehlerverteilungen, die zur Darstellung der x- und y-Achse verwendet werden.

Kapitel 12

Templat-Klasse `Matrix`

Die Grundlage aller Berechnungen, die im Rahmen der Behandlung von Spinsystemen im Dichtematrix-Formalismus durchgeführt werden, bildet das mathematische Konstrukt der Matrix. Daher ist es notwendig, eine Bibliothek zur Verfügung zu haben, die die Arithmetik dieser mathematischen Objekte umsetzt. Im Rahmen dieser Arbeit wurde die Templat-Klasse `Matrix` entwickelt, die die mathematische Grundlage der Optimierungs-Bibliothek Octane (s. Kapitel 11) bildet. Diese Bibliothek stellt zwar einen integralen Bestandteil der Optimierungs-Bibliothek Octane dar, kann jedoch davon losgelöst benutzt werden.

Das prinzipielle Design dieser Klasse enthält zum einen die Typen-Unabhängigkeit der Matrix-Elemente, d.h. es lassen sich Matrizen jeden Typs damit abbilden. Zum anderen werden Operatoren und Funktionen der in C++ für Variablen anderen Typs definierten Syntax nachempfunden, um die Konsistenz der Programmiersprache zu erhalten. Darüberhinaus wurden die Routinen aufgrund von Rechenleistungs-Vorteilen wenn möglich als `inline`-Funktionen implementiert, was die Sprünge im Hauptspeicher des Computers und somit die Laufzeit minimiert.

12.1 Verwaltung von Matrix-Objekten

In diesem Abschnitt werden die Verwaltungs-Routinen der Klasse beschrieben. Zunächst sollen die notwendigen Konstruktoren und Destruktoren der Klasse besprochen werden, durch welche Objekte dieser Klasse deklariert bzw. gelöscht weden können. Im Anschluß werden Mehtoden vorgestellt, mit denen sich die Größe einer Matrix für ein allokiertes Objekt setzen bzw.

auslesen lassen. Der dritte Abschnitt wird sich mit den zur Verfügung stehenden Subskript-Operatoren beschäftigen, über die einzelne Matrix-Elemte addressiert werden können.

12.1.1 Allgemeine Objekt-Verwaltung

Zu den Routinen der allgemeinen Verwaltung von Objekten zählen in der Regel Konstruktoren und Destruktoren. Die Konstruktoren werden verwendet, um Speicherplatz für ein Objekt der Klasse `Matrix` zu allokieren, mittels des Destruktors kann dieser wieder freigegeben werden. Die Klasse `Matrix` stellt zwei Varianten von Konstruktoren zur Verfügung, um die Allokierung von Matrizen möglichst flexibel zu gestalten. Mit dem Default-Konstruktor können leere Matrizen bestimmter Größe angelegt werden, mit dem Copy-Constructor kann eine Objekt-Kopie erstellt werden.

Konstruktoren

- `Matrix<Type> (size_t = 6, size_t = 6)`:
  Dieser Konstruktor stellt den Default-Konstruktor dar. Durch Verwendung wird eine Matrix mit Matrix-Elemten vom Typ `Type` angelegt, der jedem von C++ bereit gestellten Typ entsprechen kann. Die Größe der anzulegenden Matrix wird durch die übergebenen Parameter vom Typ `size_t` bestimmt. Hierbei entspricht der erste Parameter der Anzahl der Zeilen und der zweite Parameter den Spalten der zu allokierenden Matrix. Wird die Parameter-Liste ausgelassen, so wird eine (6×6)-Matrix angelegt.

  Beispiel:

  1. Anlegen einer (6×6)-Matrix mit Elementen vom Typ `int` $\in \Re$:
     `Matrix<int> myMatrix2();`
  2. Anlegen einer (7×5)-Matrix mit Elementen vom Typ `double` $\in \Im$:
     `Matrix<complex<double> > myMatrix3(7,5);`

- `Matrix<Type> (const Matrix<Type> )`:
  Diese überladene Variante des Konstruktors stellt den sog. "Copy-Constructor" dar. Durch Verwendung dieses Konstruktors wird eine

Kopie eines bereits allokierten Matrix-Objektes erstellt. Das Quell-Objekt wird hierbei als Parameter übergeben.

Beispiel:

1. Anlegen einer Matrix-Kopie:
   `Matrix<double> myMatrix2( myMatix1 );`

Destruktor

- `~Matrix ()`:
  Der Destruktor löscht ein bereits angelegte Objekt der Klasse `Matrix` und gibt den allokierten Hauptspeicher wieder frei. Normalerweise wird dieser bei Verlassen des Objekt-Scopes [1] automatisch ausgeführt. Möchte man ein Objekt jedoch innerhalb seines Scopes löschen, kann man den Destruktor manuell auffrufen.

  Beispiel:

  1. Manuelles Löschen eines Matrix-Objektes:
     `~myMatrix();`

12.1.2 Verwaltung der Matrix-Größe

Die charakteristische Eigenschaft einer Matrix ist ihre Größe. Die Klasse `Matrix` stellt Methoden zur Verfügung, mit denen man die Größe einer Matrix setzen bzw. auslesen kann.

Setzen der Größe

- `void setSize(size_t, size_t)`:
  Um die Größe einer Matrix nach ihrer Allokierung zu verändern, kann man diese Methode benutzen. Die neue Größe der Matrix wird als Parameter vom Typ `size_t` übergeben. Hierbei entspricht der erste Parameter der Anzahl der gewünschten Zeilen und der zweite Parameter

[1] Unter Scope versteht man den Gültigkeitsbereich einer Variablen, in diesem Fall den eines Objektes. Normalerweise wird der Scope durch den Ort der Deklaration definiert. Allokiert man eine Variable lokal z.B. innerhalb einer Schleife, so ist der Scope nur auf diese Schleife begrenzt. Wird die Schleife verlassen, wird automatisch der Destruktor ausgeführt. Global allokierte Objekte werden vor Beenden der Haupt-Routine vom Compiler abgebaut.

den gewünschten Spalten. Wird eine Matrix vergrößert, bleibt die ursprüngliche Matrix in ihren alten Grenzen als Submatrix vollständig erhalten. Soll die Matrixgröße verkleinert werden, so bleiben nur die alten Werte innerhalb dieser neuen Grenzen erhalten.

Beispiel:

1. Vergrößern einer Matrix:
   ```
   Matrix<int> myMatrix(3,3);
   myMatrix.setSize(4,4);
   ```
   $$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \rightarrow \begin{pmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} & a_{03} \\ a_{13} \\ a_{23} \\ a_{30} \quad a_{31} \quad a_{32} & a_{33} \end{pmatrix}$$

2. Verkleinern einer Matrix:
   ```
   Matrix<int> myMatrix(3,3);
   myMatrix.setSize(2,2);
   ```
   $$\begin{pmatrix} \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} & a_{02} \\ a_{12} \\ a_{20} \quad a_{21} & a_{22} \end{pmatrix} \rightarrow \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}$$

Auslesen der Größe

- `size_t rows()`:
  Mit dieser Methode kann die Anzahl der in einer Matrix vorhandenen Zeilen ausgelesen werden. Der Rückgabewert ist vom Typ `size_t`.

  Beispiel:

  1. Auslesen der Anzahl Zeilen einer Matrix:
     ```
     size_t rows = myMatrix.rows();
     ```

- `size_t cols()`:
  Mit dieser Methode kann die Anzahl der in einer Matrix vorhandenen Spalten ausgelesen werden. Der Rückgabewert ist vom Typ `size_t`.

Beispiel:

1. Auslesen der Anzahl Spalten einer Matrix:
   `size_t cols = myMatrix.cols();`

12.1.3 Verwaltung von Matrix-Elementen

Um die Elementen eines Matrix-Objektes addressieren zu können, wurde der "Bracket-Operator" definiert. Unter Verwendung dieses und des Zuweisungs-Operators "=" können einzelne Matrix-Elemente ausgelesen oder gesetzt werden. Hierbei ist zu beachten, daß die Elemente in informatischer Zählweise indiziert werden, d.h. die Zählung beginnt mit dem Element $(0, 0)$ und endet bei $(rows - 1, cols - 1)$.

- `Type &operator () (size_t, size_t)`:
  Um den Wert eines Matrix-Elementes zu setzen, wird dieser Operator verwendet. Über die Parameter vom Typ `size_t` in der Reihenfolge (Reihe, Spalte) wird dieses Element in informatischer Zählweise (s.o.) addressiert. Der zu setzende Wert muß vom Typ `Type` sein, der bei der Allokierung des Objektes vereinbart wurde, unter Umständen muß Type-Casting [2] eingesetzt werden.

  Beispiel:

  1. Setzen eines Matrix-Elements:
     ```
     Matrix<complex<double> > myMatrix(2,2);
     myMatrix(0,0) = (complex<double>)1.0;
     ```

- `Type operator () (size_t, size_t) const`:
  Um den Wert eines Matrix-Elementes auszulesen, wird dieser Operator verwendet. Über die Parameter vom Typ `size_t` in der Reihenfolge (Reihe, Spalte) wird dieses Element in informatischer Zählweise (s.o.) addressiert. Der Rückgabewert ist vom Type `Type`, der bei der Allokierung des Objektes gewählt wurde.

[2] Unter "Type-Casting" versteht man das Maskieren eines Variablen-Typs als ein anderer Typ. Zu beachten ist, daß bei Type-Casts in einen Typ niedriger Präzision Informationen verloren gehen, z.B. wenn ein Fließkomma-Typ als Ganzzahl-Typ maskiert wird verliert man die Mantissen-Information.

Beispiel:

1. Auslesen eines Matrix-Elements:
   ```
   Matrix<complex<double> > myMatrix(2,2);
   complex<double> myElement = myMatrix(0,0);
   ```

12.1.4 Generierung spezieller Matrizen

Zwei Matrizen besitzen gruppentheoretisch besondere Bedeutung. Wegen ihres häufigen Auftretens in der Matrix-Algebra wurde eine einfache Generierung dieser Matrizen in die Klasse `Matrix` aufgenommen Dies ist zum einen die Nullmatrix als neutrales Element der Matrix-Addition und zum anderen die Einheitsmatrix, die das neutrale Element der Matrix-Multiplikation darstellt. Werden die in diesem Abschnitt vorgestellten Methoden verwendet, so beachte man, daß die ursprünglich im Objekt gespeicherten Daten mit den Werten der gewünschten Matrix überschrieben werden.

Nullmatrix

- `void null()`:
  Um eine allokierte Matrix unter Beibehaltung aller weiteren Parameter mit den Werten der Nullmatrix zu füllen, kann man diese Methode verwenden. Hierbei wirkt die Methode direkt auf die Datenelemente des Objekts, d.h. es wird weder eine Parameter-Liste noch ein Rückgabewert benötigt.

  Beispiel:

  1. Erzeugung einer Nullmatrix:
     ```
     Matrix<complex<double> > myMatrix(2,2);
     myMatrix.null();
     ```

- `void null(const size_t, const size_t)`:
  Um ein bereits angelegtes Matrix-Objekt in eine Nullmatrix neuer Größe umzuwandlen, kann man diese Methode der Klasse verwenden. Hierbei wird die neue Matrix-Größe als Parameter-Liste (Reihe, Spalte) vom Typ `size_t` übergeben. Diese Methode verfügt über keinen Rückgabewert, weil sie direkt auf die Datenelemente des Objektes wirkt.

Beispiel:

1. Erzeugung einer Nullmatrix neuer Größe:
```
Matrix<complex<double> > myMatrix(2,2);
myMatrix.null(4,4);
```

Einheitsmatrix

- `void unit()`:
  Um eine allokierte Matrix unter Beibehaltung aller weiteren Parameter mit den Werten der Einheitsmatrix zu füllen, kann man diese Methode verwenden. Hierbei wirkt die Methode direkt auf die Datenelemente des Objekts, d.h. es wird weder eine Parameter-Liste noch ein Rückgabewert benötigt. Ist die Matrix vor Auffruf der Methode nicht quadratisch, so bestimmt die kleinste Dimension der Quellmatrix die Dimension der Einheitsmatrix. War die Matrix vorher z.B. eine (3×7)-Matrix, so hat sie nach Aufruf dieser Methode die Dimension (3×3).

  Beispiel:

  1. Erzeugung einer Einheitsmatrix:
  ```
  Matrix<complex<double> > myMatrix(2,2);
  myMatrix.unit();
  ```

- `void unit(const size_t)`:
  Um ein bereits angelegtes Matrix-Objekt in eine Einheitsmatrix neuer Größe umzuwandlen, kann man diese Methode der Klasse verwenden. Hierbei wird die neue Matrix-Größe in einem Parameter des Typs `size_t` übergeben, weil eine Einheitsmatrix in jedem Falle eine quadratische Matrix sein muß. Diese Methode verfügt über keinen Rückgabewert, weil sie direkt auf die Datenelemente des Objektes wirkt.

  Beispiel:

  1. Erzeugung einer Einheitsmatrix neuer Größe:
  ```
  Matrix<complex<double> > myMatrix(2,2);
  myMatrix.unit(4);
  ```

12.2 Operatoren

Um die Algebra von Matrizen in C++ darstellen zu können, müssen die entsprechenden in der Programmiersprache definierten Operatoren überladen werden und für die Verwendung innerhalb der Klasse angepaßt werden. Hierbei sollte der in C++ übliche Umfang an Operationen definiert werden, vor allem die häufig eingesetzten mit Berechnungen kombinierten Zuweisungsoperatoren.

Die mit Zuweisungen verküpften Operationen werden unter dem entsprechenden Operator beschrieben. Wie bei einer Zuweisungs-Operation üblich, wird stets der rechts vom Operator stehende Variablen-Wert der links vom Operator stehenden Variable zugewiesen. Hierbei gilt stets zu beachten, daß die ursprünglichen Objekt-Daten überschrieben werden.

Die Wirkung des Zuweisungsoperators unter Verwendung des Bracket-Operators wurde bereits in Abschnitt 12.1.3 besprochen.

12.2.1 Der Zuweisungsoperator =

- `Matrix<Type> = (const Matrix<Type>)`:
  Diese Zuweisungs-Operation weist der linken Matrix den Wert der rechten Matrix zu, so daß für die Operation $B = A$ die Elementbeziehung $b_{ij} = a_{ij}$ gilt. Neben den Element-Werten übernimmt die linke Matrix ebenfalls die Größe der rechten Matrix, falls beide Matrizen vor Anwendung des Operators unterschiedliche Dimensionen aufwiesen.

  Beispiel:

  1. Zuweisen von Matrizen:
     ```
     myMatrix2 = myMatrix1;
     ```

12.2.2 Der Additions-Operator +

- `Matrix<Type> + ()`:
  Diese Operation stellt den Spezial-Fall der Multiplikation einer Matrix mit dem Skalar +1 dar. Diese Operation ist prinzipiell ohne Wirkung, mußte aber in den Sprachumfang aufgenommen werden, damit C++-Compiler keine Probleme bereiten.

- `Matrix<Type> + (Matrix<Type>, Matrix<Type>)`:
  Dieser Operator stellt die Matrix-Addition dar. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrizen bereitgestellt.

  Beispiel:

  1. Addieren von Matrizen ohne Verwertung des Rückgabewertes:
     `( myMatrix1 + myMatrix2 );`
  2. Addieren von Matrizen mit Übergabe des Rückgabewertes an eine Funktion:
     `myFunction( myMatrix1 + myMatrix2 );`
  3. Addieren von Matrizen mit anschließender Zuweisung:
     `myMatrix3 = myMatrix1 + myMatrix2;`

- `Matrix<Type> += (const Matrix<Type> )`:
  Diese Operation stellt die mit einer Zuweisung kombinierte Matrix-Addition dar. Diese Operation ist äquivalent zur Formulierung $A = A + B$.

  Beispiel:

  1. Kombiniertes Zuweisen und Addieren von Matrizen:
     `myMatrix1 += myMatrix2;`

12.2.3 Der Subtraktions-Operator -

- `Matrix<Type> - ()`:
  Diese Operation stellt die Negierung aller Matrix-Elemente dar. Es entspricht dem Spezial-Fall der Multiplikation einer Matrix mit dem Skalar -1. Das Ergebnis der Operation wird als Matrix gleichen Typs wie die Eingabematrix bereitgestellt.

  Beispiel:

  1. Negieren von Matrix-Elementen ohne Verwertung des Rückgabewertes:

```
( -myMatrix2 );
```

2. Negieren von Matrix-Elementen mit Übergabe des Rückgabewertes an eine Funktion:
   `myFunction( -myMatrix1 );`
3. Negieren von Matrix-Elementen mit anschließender Zuweisung:
   `myMatrix2 = -myMatrix1;`

- `Matrix<Type> - (Matrix<Type>, Matrix<Type>)`:
  Dieser Operator stellt die Matrix-Subtraktion dar. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrizen bereitgestellt.

  Beispiel:

  1. Subtrahieren von Matrizen ohne Verwertung des Rückgabewertes:
     `( myMatrix1 - myMatrix2 );`
  2. Subtrahieren von Matrizen mit Übergabe des Rückgabewertes an eine Funktion:
     `myFunction( myMatrix1 - myMatrix2 );`
  3. Subtrahieren von Matrizen mit anschließender Zuweisung:
     `myMatrix3 = myMatrix1 - myMatrix2;`

- `Matrix<Type> -= (const Matrix<Type> )`:
  Diese Operation stellt die mit einer Zuweisung kombinierte Matrix-Subtraktion dar. Diese Operation ist äquivalent zur Formulierung $A = A - B$.

  Beispiel:

  1. Kombiniertes Zuweisen und Subtrahieren von Matrizen:
     `myMatrix1 -= myMatrix2;`

12.2.4 Der Multiplikations-Operator *

- `Matrix<Type> * (Matrix<Type>, Matrix<Type>)`:
  Dieser Operator stellt die Matrix-Multiplikation dar. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen

Typ wie die Eingabe-Matrizen bereitgestellt.

Beispiel:

1. Multiplikation von Matrizen ohne Verwertung des Rückgabewertes:
   `( myMatrix1 * myMatrix2 );`
2. Multiplikation von Matrizen mit Übergabe des Rückgabewertes an eine Funktion:
   `myFunction( myMatrix1 * myMatrix2 );`
3. Multiplikation von Matrizen mit anschließender Zuweisung:
   `myMatrix3 = myMatrix1 * myMatrix2;`

- `Matrix<Type> * (Type, Matrix<Type> )`:
  Dieser Operator stellt die Multiplikation einer Matrix mit einem Skalar gemäß der Formulierung $s \cdot A$ dar. Hierbei muß der Skalar vom selben Typ sein, wie die Elemente der Matrix. Ist dies nicht der Fall, muß der Skalar mittels Type-Cast (s. Fußnote in Abschnitt 12.1.3) maskiert werden. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrizen bereitgestellt.

  Beispiel:

  1. Multiplikation einer Matrix mit einem Skalar ohne Verwertung des Rückgabewertes:
     `( 2.0 * myMatrix );`
  2. Multiplikation einer Matrix mit einem Skalar mit Übergabe des Rückgabewertes an eine Funktion:
     `myFunction( 2.0 * myMatrix );`
  3. Multiplikation einer Matrix mit einem Skalar mit anschließender Zuweisung:
     `myMatrix2 = 2.0 * myMatrix1;`

- `Matrix<Type> * (Matrix<Type>, Type )`:
  Dieser Operator stellt die Multiplikation einer Matrix mit einem Skalar

gemäß der Formulierung $A \cdot s$ dar. Diese Operation ist die permutierte Formulierung von `Matrix<Type> * (Type, Matrix<Type> )`, daher gilt das dort gesagte auch für diesen Fall.

- `Matrix<Type> *= (const Type )`:
  Diese Operation stellt die mit einer Zuweisung kombierte Multiplikation einer Matrix mit einem Skalar dar. Diese Operation ist äquivalent zur Formulierung $A = s \cdot A$.

  Beispiel:

  1. Kombiniertes Zuweisen und Multiplizieren einer Matrix mit einem Skalar:
     `myMatrix1 *= 2.0;`

- `Matrix<Type> *= (const Matrix<Type> )`:
  Diese Operation stellt die mit einer Zuweisung kombinierte Matrizen-Multiplikation dar. Diese Operation ist äquivalent zur Formulierung $A = A \cdot B$.

  Beispiel:

  1. Kombiniertes Zuweisen und Multiplizieren einer Matrix mit einem Skalar:
     `myMatrix2 *= myMatrix1;`

12.2.5 Der Divisions-Operator /

- `Matrix<Type> / (Matrix<Type>, Matrix<Type>)`:
  Dieser Operator stellt die Matrix-Division gemäß $A \cdot B^{-1}$ dar. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrizen bereitgestellt.

  Beispiel:

  1. Division von Matrizen ohne Verwertung des Rückgabewertes:
     `( myMatrix1 / myMatrix2 );`

2. Division von Matrizen mit Übergabe des Rückgabewertes an eine Funktion:
   `myFunction( myMatrix1 / myMatrix2 );`
3. Division von Matrizen mit anschließender Zuweisung:
   `myMatrix3 = myMatrix1 / myMatrix2;`

- `Matrix<Type> / (Type, Matrix<Type>)`:
  Dieser Operator stellt die Division eines Skalars durch eine Matrix gemäß der Formulierung $s \cdot A^{-1}$ dar. Hierbei muß der Skalar vom selben Typ sein, wie die Elemente der Matrix. Ist dies nicht der Fall, muß der Skalar mittels Type-Cast (s. Fußnote in Abschnitt 12.1.3) maskiert werden. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrizen bereitgestellt.

  Beispiel:

  1. Division eines Skalars durch eine Matrix ohne Verwertung des Rückgabewertes:
     `( 2.0 / myMatrix );`
  2. Division eines Skalars durch eine Matrix mit Übergabe des Rückgabewertes an eine Funktion:
     `myFunction( 2.0 / myMatrix );`
  3. Division eines Skalars durch eine Matrix mit anschließender Zuweisung:
     `myMatrix2 = 2.0 / myMatrix1;`

- `Matrix<Type> / (Matrix<Type>, Type )`:
  Im Gegensatz zur Multiplikation ist die skalare Division nicht permutativ. Daher stellt dieser Operator die Division einer Matrix durch einen Skalar gemäß der Formulierung $A \cdot \frac{1}{s}$ dar. Hierbei muß der Skalar vom selben Typ sein, wie die Elemente der Matrix. Ist dies nicht der Fall, muß der Skalar mittels Type-Cast (s. Fußnote in Abschnitt 12.1.3) maskiert werden. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrizen bereitgestellt.

Beispiel:

1. Division einer Matrix durch einen Skalar ohne Verwertung des Rückgabewertes:
   `( myMatrix / 2.0 );`
2. Division einer Matrix durch einen Skalar mit Übergabe des Rückgabewertes an eine Funktion:
   `myFunction( myMatrix / 2.0 );`
3. Division einer Matrix durch einen Skalar mit anschließender Zuweisung:
   `myMatrix2 = myMatrix1 / 2.0;`

- `Matrix<Type> /= (const Type )`:
  Diese Operation stellt die mit einer Zuweisung kombinierte Division einer Matrix durch einen Skalar dar. Diese Operation ist äquivalent zur Formulierung $A = \frac{1}{s} \cdot A$.

  Beispiel:

  1. Kombiniertes Zuweisen und Dividieren einer Matrix durch einen Skalar:
     `myMatrix /= 2.0;`

12.2.6 Logische Operatoren

- `bool == (const Matrix<Type>, Matrix<Type> )`:
  Dieser Operator entspricht der positiven relationalen Vergleichsoperation für zwei Matrizen gleichen Typs. Die Routine vergleicht die Werte der Matrix-Elemente gleicher Position beider Matrizen und hat den Rückgabewert `true` für identische Matrizen bzw. `false` für unterschiedliche.

  Beispiel:

  1. Verleich von Matrizen ohne Verwertung des Rückgabewertes:
     `( myMatrix1 == myMatrix2 );`

2. Verleich von Matrizen mit Verwertung des Ergebnisses in einer konditionalen Anweisung:
   `if ( myMatrix1 == myMatrix2 ){ ... };`

- `bool != (const Matrix<Type>, Matrix<Type> )`:
  Dieser Operator entspricht der negativen relationalen Vergleichsoperation für zwei Matrizen gleichen Typs. Die Routine vergleicht die Werte der Matrix-Elemente gleicher Positionen beider Matrizen und hat den Rückgabewert `true` für unterschiedliche Matrizen bzw. `false` für identische.

  Beispiel:

  1. Verleich von Matrizen ohne Verwertung des Rückgabewertes:
     `( myMatrix1 != myMatrix2 );`
  2. Verleich von Matrizen mit Verwertung des Ergebnisses in einer konditionalen Anweisung:
     `if ( myMatrix1 != myMatrix2 ){ ... };`

12.2.7 Operatoren mit funktionaler Bedeutung

- `Matrix<Type> ! (Matrix<Type>)`:
  Durch Verwendung dieser Methode kann man die Inverse A^{-1} einer Matrix berechnen. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrix bereitgestellt.

  Beispiel:

  1. Berechnung der Inversen einer Matrix ohne Verwertung des Rückgabewertes:
     `( !myMatrix );`
  2. Berechnung der Inversen einer Matrix mit Übergabe des Rückgabewertes an eine Funktion:
     `myFunction( !myMatrix );`

3. Berechnung der Inversen einer Matrix mit anschließender Zuweisung:
   `myMatrix2 = !myMatrix1;`

- `Matrix<Type> ^(Matrix<Type>, size_t)`:
  Diese Operation berechnet die skalare Potenz $B = (A)^n$ einer Matrix, so daß die Beziehung $B = \prod_n A$ gilt. Hierbei ist die anzugebende Potenz vom Typ `size_t`. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrizen bereitgestellt.

  Beispiel:

  1. Berechnung der sklaren Potenz einer Matrix ohne Verwertung des Rückgabewertes:
     `( myMatrix^3 );`
  2. Berechnung der skalaren Potenz einer Matrix mit anschließender Zuweisung:
     `myMatrix2 = myMatrix1^3;`

- `Matrix<Type> ^= (const size_t )`:
  Dieser Operator entspricht der mit einer Zuweisung kombinierten Berechnung der skalaren Potenz $A = (A)^n$ einer Matrix, so daß die Beziehung $A = \prod_n A$ gilt. Hierbei ist die anzugebenden Potenz vom Typ `size_t`.

  Beispiel:

  1. Berechnung der skalaren Potenz einer Matrix mit anschließender Zuweisung:
     `myMatrix ^= 3;`

- `Matrix<Type> ~ (Matrix<Type>)`:
  Die Verwendung dieses Operators berechnet die Transponierte der Eingabe-Matrix $B = A^T$ gemäß der Beziehung $b_{ij} = a_{ji}$. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrix bereitgestellt.

Beispiel:

1. Berechnung der Transponierten einer Matrix ohne Verwertung des Rückgabewertes:
   `( ~myMatrix );`
2. Berechnung der Transponierten einer Matrix mit anschließender Zuweisung:
   `myMatrix2 = ~myMatrix1;`

12.3 Funktionen

12.3.1 Funktionale Matrix-Operationen

Die im Folgenden beschriebenen Routinen sind in zwei Klassen eingeteilt. Zunächst werden die innerhalb des Objektes als Methoden implementierten Operationen (*Members*) besprochen. Daran schließt sich eine Auflistung der außerhalb des Objektes als Funktionen (*Non-Members*) definierte Operationen an. Diese Aufteilung führt in einigen Fällen zu renundanten Definitionen, die jedoch zu synthaktischen Vorteilen in der Verwendung der Klasse führen.

Members

- `Matrix<Type> adj( void )`:
  Diese Methode berechnet die Adjungierte einer Matrix $B = A^{\dagger}$ für $A \in \Im$ gemäß der Beziehung $b_{ij} = a^*_{ji}$. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrix bereitgestellt.

  Beispiel:

  1. Berechnen der Adjungierten einer Matrix ohne Verwertung des Rückgabewertes:
     `( myMatrix.adj() );`
  2. Berechnung der Adjungierten einer Matrix mit anschließender Zuweisung:
     `myMatrix2 = myMatrix1.adj();`

- `Matrix<Type> inv( void )`:
  Diese Methode berechnet die Inverse einer Matrix $B = A^{-1}$ gemäß der Beziehung $A \cdot A^{-1} = 1$. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrix bereitgestellt.

  Beispiel:

  1. Berechnen der Inversen einer Matrix ohne Verwertung des Rückgabewertes:
     `( myMatrix.inv() );`
  2. Berechnung der Inversen einer Matrix mit anschließender Zuweisung:
     `myMatrix2 = myMatrix1.inv();`

- `Type norm( void )`:
  Diese Methode berechnet die Norm einer Matrix gemäß

$$\| A \| = \sqrt{\sum_i \sum_j |a_{ij}|}.$$

  Die Norm wird für das aktuelle Objekt bestimmt und als Rückgabewert vom Typ `Type` bereitgestellt, der bei der Allokierung des Objektes vereinbart wurde.

  Beispiel:

  1. Berechnen der Norm einer Matrix ohne Verwertung des Rückgabewertes:
     `( myMatrix.norm() );`
  2. Berechnung der Norm einer Matrix mit anschließender Zuweisung:
     `double myNorm = myMatrix.norm();`

- `Type normFrobenius( void )`:
  Diese Methode berechnet die Fröbenius-Norm einer Matrix gemäß

$$\| A \| = \sqrt{\mathrm{Sp}\left\{A^{\dagger} \cdot A\right\}}.$$

Die Fröbenius-Norm wird für das aktuelle Objekt bestimmt und als Rückgabewert vom Typ `Type` bereitgestellt, der bei der Allokierung des Objektes vereinbart wurde.

Beispiel:

1. Berechnen der Fröbenius-Norm einer Matrix ohne Verwertung des Rückgabewertes:
   `( myMatrix.normFrobenius() );`
2. Berechnung der Fröbenius-Norm einer Matrix mit anschließender Zuweisung:
   `double myNorm = myMatrix.normFrobenius();`

- `double normInfinity( void )`:
  Diese Methode berechnet die Unendlichkeits-Norm einer Matrix, d.h. die maximale Reihensumme, gemäß

$$\| A \| = \max_i \left\{ \sum_j |a_{ij}| \right\}.$$

  Die Unendlichkeits-Norm wird für das aktuelle Objekt bestimmt und als Rückgabewert vom Typ `double` bereitgestellt.

  Beispiel:

  1. Berechnen der Unendlichkeits-Norm einer Matrix ohne Verwertung des Rückgabewertes:
     `( myMatrix.normInfinity() );`
  2. Berechnung der Unendlichkeits-Norm einer Matrix mit anschließender Zuweisung:
     `double myNorm = myMatrix.normInfinity();`

- `Matrix<Type> kron( Matrix<Type> )`:
  Diese Methode berechnet das direkte Matrix-Produkt (das sog. Kronecker-Produkt) gemäß $C = A \otimes B$, wobei die Matrix A durch das aufrufende Objekt referenziert wird und die Matrix B als Parameter gleichen Typs `Type` wie das Objekt. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrizen

bereitgestellt.

Beispiel:

1. Berechnen des direkten Produktes zweier Matrizen ohne Verwertung des Rückgabewertes:
   `( myMatrix1.kron( myMatrix2 ) );`
2. Berechnung des direkten Produktes zweier Matrizen mit anschließender Zuweisung:
   `myMatrix3 = myMatrix1.kron( myMatrix2 );`

- `Type trace( void )`:
  Diese Methode berechnet die Spur einer quadratischen Matrix gemäß $\mathrm{Sp}(A) = \sum_i a_{ii}$. Dies ist die zuweisungsfreie Variante, das Ergebnis wird als Rückgabewert vom gleichen Typ wie die Eingabe-Matrizen bereitgestellt.

  Beispiel:

  1. Berechnen der Spur einer Matrix ohne Verwertung des Rückgabewertes:
     `( myMatrix.trace() );`
  2. Berechnung der Spur einer Matrix mit anschließender Zuweisung:
     `double myTrace = myMatrix.trace();`

Non-Members

- `Matrix<Type> expm( Matrix<Type> )`:
  Diese Funtion berechnet unter Verwendung der Padé-Näherung [88, 89] die Exponierte einer Matrix gemäß $B = \exp(A)$. Die Funktion erwartet eine Matrix vom Typ `Type` als Eingabe-Parameter und stellt die Exponierte als Rückgabewert gleichen Typs bereit.

  Beispiel:

  1. Exponieren einer Matrix:
     `myMatrix2 = expm( myMatrix1 );`

- `Matrix<Type> commute(Matrix<Type>, Matrix<Type>)`:
  Diese Funktion berechnet den Kommutator zweier Matrizen gemäß $[A, B] = A \cdot B - B \cdot A$. Die Funktion erwartet die zu kommutierenden Matrizen vom Typ `Type` als Eingabe-Parameter und stellt die kommutierte Matrix als Rückgabewert gleichen Typs bereit.

  Beispiel:

  1. Berechnung des Kommutators einer Matrix:
     `myMatrix3 = commute( myMatrix1, myMatrix2 );`

- `Matrix<Type> kroncommute(Matrix<Type>, Matrix<Type>)`:
  Diese Funktion berechnet den Kronecker-Kommutator zweier Matrizen gemäß $[A, B] = A \otimes B - B \otimes A^T$. Die Funktion erwartet die zu kommutierenden Matrizen vom Typ `Type` als Eingabe-Parameter und stellt die kommutierte Matrix als Rückgabewert gleichen Typs bereit.

  Beispiel:

  1. Berechnung des Kronecker-Kommutators einer Matrix:
     `myMatrix3 = kroncommute( myMatrix1, myMatrix2 );`

- `Type trace( Matrix<Type> )`:
  Diese Funktion berechnet die Spur einer quadratischen Matrix gemäß $\mathrm{Sp}(A) = \sum_i a_{ii}$. Die Funktion erwartet die Matrix, deren Spur bestimmt werden soll, als Eingabe-Parameter. Das Ergebnis wird als Rückgabewert vom gleichen Typ `Type` wie die Elemente Eingabe-Matrix bereitgestellt.

  Beispiel:

  1. Berechnung der Spur einer Matrix :
     `double myTrace = trace( myMatrix );`

- `Matrix<Type> adj( Matrix<Type> )`:
  Diese Funktion berechnet die Adjungierte einer Matrix $B = A^\dagger$ für $A \in \Im$ gemäß der Beziehung $b_{ij} = a^*_{ji}$. Die Funktion erwartet die Matrix, deren Adjungierte bestimmt werden soll, als Eingabe-Parameter.

Das Ergebnis wird als Rückgabewert vom gleichen Typ `Type` wie die Eingabe-Matrix bereitgestellt.

Beispiel:

1. Berechnen der Adjungierten einer Matrix:
   `myMatrix2 = adj( myMatrix1 );`

- `Matrix<Type> sub( size_t,size_t,size_t,size_t,Matrix<Type> )`:
  Diese Funktion liest eine Untermatrix aus einer gegebenen Matrix aus. Die Elemente, die in der Quellmatrix die Sub-Matrix bestimmen, werden über die ersten vier Parameter vom Typ `size_t` in der Form (Zeile (Start), Spalte (Start), Zeile (Ende), Spalte (Ende)) definiert. Hierbei werden Elemente in informatischer Zählweise addressiert. Der fünfte Parameter ist die Matrix, aus der die Untermatrix ausgelesen werden soll. Das Ergebnis wird als Rückgabewert vom gleichen Typ `Type` wie die Eingabe-Matrix bereitgestellt.

  Beispiel:

  1. Auslese einer Sub-Matrix:
     ```
     Matrix<int> myMatrix(3,3);
     sub(0,0, 1,1, myMatrix);
     ```

$$\left(\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{matrix} a_{02} \\ a_{12} \end{matrix} \right) \rightarrow \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}$$

12.3.2 I/O-Routinen

- `writeMAT4( char*, char* )`:
  Diese Methode schreibt eine Matrix unter einem symbolischen Namen in eine Datei, deren Format dem binären Level 1.0 Format von Matlab [90] entspricht. Hierbei werden die Parameter in der Reihenfolge *Dateiname, Symbolname* als String übergeben.

Beispiel:

1. Speichern einer Matrix:
   `myMatrix.writeMAT4(''file.mat'', ''myMatrix'');`

- `readMAT4( char*, char* )`:
  Diese Methode liest eine Matrix unter einem symbolischen Namen aus einer Datei, deren Format dem binären Level 1.0 Format von Matlab [90] entspricht. Hierbei werden die Parameter in der Reihenfolge *Dateiname, Symbolname* als String übergeben.

  Beispiel:

  1. Einlesen einer Matrix:
     `myMatrix.readMAT4(''file.mat'', ''myMatrix'');`

- `ostream& << (ostream, Matrix<Type>)`:
  Dieser Operator definiert den Outstream für eine Matrix. Hierbei wird die Matrix im Klartext (ASCII) über den definierte Outstream ausgegeben.

  Beispiel:

  1. Ausgabe einer Matrix auf die Standard-Ausgabe:
     `cout << myMatrix;`

- `istream& >> (istream, Matrix<Type> )`:
  Dieser Operator definiert den Instream für eine Matrix. Hierbei wird die Matrix über den definierte Instream eingelesen.

  Beispiel:

  1. Eingabe einer Matrix über die Standard-Eingabe:
     `cin >> myMatrix;`

Kapitel 13

`Wtest`-Programme

In diesem Kapitel werden die im Rahmen dieser Arbeit geschriebnen `Wtest`-Skripte gezeigt. Dabei handelt es sich um reine dipolare Relaxations-Operatoren (DD und DD/DD-kreuzkorreliert). In den hier gezeigten Versionen wird dabei die spektrale Dichtefunktion des starren Rotors verwendet (vergl. Kapitel 3.2.2).

13.1 2-Spinsystem

```
##############################################################################
#                                                                            #
# File: redfield.2spin.wtest                                                 #
# Project: create a matlab file containing the Redfield relaxation matrix for  #
#     a 2-spin system
# Date: Sep 13, 2000                                                         #
#                                                                            #
##############################################################################


##############################################################################
# Constants - parameters - defintions                                        #
#                                                                            #
##############################################################################


# DEFINITION: spin system: is (i=1H, s=13C)
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

spins ab
setisp a,b

# gyromagnetic ratio of i (bcd)= 1H - [s*A/kg]
setvar g_i = 2.6752e8

# gyromagnetic ratio of s (a) = 13C - [s*A/kg]
setvar g_s = 6.728e7

# distances between two nuclei - [m]
setvar r_ii = 1.75e-10
setvar r_si = 1.07e-10

# correlation time - [s]
setvar tau_c = 5e-9

# CONSTANTS: somes usefull constants
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Planck's h/2pi - [J*s]
setvar hq = 1.0546e-34

# permeability of vacuum - [V*s/(A*m)]
```

```
setvar m0 = 1.256637061e-6

# gyromagnetic ratio of 1H - [s*A/kg]
setvar g_1H = 2.6752e8

# CONSTANTS: magnetic field strength
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# spectrometer frequency - [1/s]
setvar v0 = 600e6

# CALCULATED: spin frequencies
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
evalu v_i = v0 * g_i / g_1H
evalu v_s = v0 * g_s / g_1H

evalu w_i = 2*pi * v_i
evalu w_s = 2*pi * v_s


###############################################################################
# spectral densities                                                          #
#                                                                             #
###############################################################################

# spectral density functions
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
evalu J_0 = 2 * tau_c / (5 * ( 1 + pow(0*tau_c, 2) ) )
evalu J_i = 2 * tau_c / (5 * ( 1 + pow(w_i*tau_c, 2) ) )
evalu J_s = 2 * tau_c / (5 * ( 1 + pow(w_s*tau_c, 2) ) )
evalu J_iPi = 2 * tau_c / (5 * ( 1 + pow((w_i + w_i)*tau_c, 2) ) )
evalu J_iMs = 2 * tau_c / (5 * ( 1 + pow((w_i - w_s)*tau_c, 2) ) )
evalu J_iPs = 2 * tau_c / (5 * ( 1 + pow((w_i + w_s)*tau_c, 2) ) )

# spatial functions for dipolar relaxation
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
evalu c0dp_si = -sqrt(6) * m0 * hq * g_s * g_i / ( 4 * pi * pow(r_si, 3) )
evalu c0dp_ii = -sqrt(6) * m0 * hq * g_i * g_i / ( 4 * pi * pow(r_ii, 3) )


###############################################################################
# Legendre polynoms                                                           #
#                                                                             #
###############################################################################

# legendre polynoms
# ~~~~~~~~~~~~~~~~~

evalu P_auto = 1


###############################################################################
# tensor operators for the dipolar interaction: A_q_w_spins                   #
# e.g. p0iMsAB: A for (q = +0), (w = wi-ws) and (spins = ab)                  #
#      m2iPsAC: A for (q = -2), (w = wi+ws) and (spins = ac)                  #
#                                                                             #
###############################################################################

# q = 0, w = 0, hetero nuclear
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
seto p0OAB = (2/sqrt(6)) * az*bz


# q = 0, w = +-(w_i - w_s)
# ~~~~~~~~~~~~~~~~~~~~~~~~
seto p0iMsAB = (-0.5/sqrt(6)) * ( b+*a- + b-*a+ )


# q = 1, w = w_s
# ~~~~~~~~~~~~~~
seto p1sAB = -0.5 * bz*a+


# q = -1, w = w_s
# ~~~~~~~~~~~~~~~
seto m1sAB = 0.5 * bz*a-


# q = 1, w = w_i
# ~~~~~~~~~~~~~~
seto p1iBA = -0.5 * b+*az


# q = -1, w = w_i
# ~~~~~~~~~~~~~~~
seto m1iBA = 0.5 * b-*az


# q = 2, w = +-(w_i + w_s)
```

```
# ~~~~~~~~~~~~~~~~~~~~~~
seto p2iPsAB = 0.5 * a+*b+


# q = -2, w = +-(w_i + w_s)
# ~~~~~~~~~~~~~~~~~~~~~~~~
seto m2iPsAB = 0.5 * a-*b-


##############################################################################
# relaxation operator GAMMA: AUTO-CORRELATED                                 #
#                                                                            #
##############################################################################


# q = -2 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoM2 \
= 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [p2iPsAB, [m2iPsAB, ]]


# q = -1 => pow(-1, q) = -1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoM1 \
= -0.5 * c0dp_si * c0dp_si * P_auto * J_s * [p1sAB, [m1sAB, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [p1iBA, [m1iBA, ]]


# q = 0 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~
sets Lauto0 \
= 0.5 * c0dp_si * c0dp_si * P_auto * J_0 * [p00AB, [p00AB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iMs * [p0iMsAB, [p0iMsAB, ]]


# q = +1 => pow(-1, q) = -1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoP1 \
= -0.5 * c0dp_si * c0dp_si * P_auto * J_s * [m1sAB, [p1sAB, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [m1iBA, [p1iBA, ]]


# q = +2 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoP2 \
= 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [m2iPsAB, [p2iPsAB, ]]


# complete L_auto.
# ~~~~~~~~~~~~~~~
sets L_auto \
= LautoM2 + LautoM1 + Lauto0 + LautoP1 + LautoP2


##############################################################################
# relaxation matrix.                                                         #
#                                                                            #
##############################################################################

# relaxation operator GAMMA - [1/s^2]
#
sets red \
= L_auto
save redfield.J0.5ns.mat, red

quit
```

13.2 3-Spinsystem

```
##############################################################################
#                                                                            #
# File: redfield.3spin.wtest                                                 #
# Project: create a matlab file containing the Redfield relaxation matrix for  #
#      a 3-spin system
# Date: Sep 13, 2000                                                         #
#                                                                            #
##############################################################################


##############################################################################
# Constants - parameters - defintions                                        #
#                                                                            #
##############################################################################
```

```
# DEFINITION: spin system: i2s (i=1H, s=13C, tetraeder geometry ala C(R2)(H2))
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

spins abc
setisp a,bc

# gyromagnetic ratio of i (bcd)= 1H - [s*A/kg]
setvar g_i = 2.6752e8

# gyromagnetic ratio of s (a) = 13C - [s*A/kg]
setvar g_s = 6.728e7

# distances between two nuclei - [m]
setvar r_ii = 1.75e-10
setvar r_si = 1.07e-10

# bond's angle - []
setvar theta_isi = 109.44

# correlation time - [s]
setvar tau_c = 1e-9

# CONSTANTS: somes usefull constants
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Planck's h/2pi - [J*s]
setvar hq = 1.0546e-34

# permeability of vacuum - [V*s/(A*m)]
setvar m0 = 1.256637061e-6

# gyromagnetic ratio of 1H - [s*A/kg]
setvar g_1H = 2.6752e8

# CONSTANTS: magnetic field strength
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# spectrometer frequency - [1/s]
setvar v0 = 600e6

# CALCULATED: spin frequencies
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
evalu v_i = v0 * g_i / g_1H
evalu v_s = v0 * g_s / g_1H

evalu w_i = 2*pi * v_i
evalu w_s = 2*pi * v_s


##############################################################################
# spectral densities                                                         #
#                                                                            #
##############################################################################

# spectral density functions
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
evalu J_0 = 2 * tau_c / (5 * ( 1 + pow(0*tau_c, 2) ) )
evalu J_i = 2 * tau_c / (5 * ( 1 + pow(w_i*tau_c, 2) ) )
evalu J_s = 2 * tau_c / (5 * ( 1 + pow(w_s*tau_c, 2) ) )
evalu J_iPi = 2 * tau_c / (5 * ( 1 + pow((w_i + w_i)*tau_c, 2) ) )
evalu J_iMs = 2 * tau_c / (5 * ( 1 + pow((w_i - w_s)*tau_c, 2) ) )
evalu J_iPs = 2 * tau_c / (5 * ( 1 + pow((w_i + w_s)*tau_c, 2) ) )

# spatial functions for dipolar relaxation
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
evalu c0dp_si = -sqrt(6) * m0 * hq * g_s * g_i / ( 4 * pi * pow(r_si, 3) )
evalu c0dp_ii = -sqrt(6) * m0 * hq * g_i * g_i / ( 4 * pi * pow(r_ii, 3) )


##############################################################################
# Legendre polynoms                                                          #
#                                                                            #
##############################################################################

# angles between dipoles
# ~~~~~~~~~~~~~~~~~~~~~~
setvar t_ab_ac = 109.44
setvar t_ab_bc = 35.28

setvar t_ac_ab = 109.44
setvar t_ac_bc = 35.28

setvar t_bc_ab = 35.28
setvar t_bc_ac = 35.28


# legendre polynoms
```

```
# ~~~~~~~~~~~~~~~~

evalu P_auto = 1

evalu P_ab_ac = 0.5 * ( 3*pow(cos(t_ab_ac), 2) - 1 )
evalu P_ab_bc = 0.5 * ( 3*pow(cos(t_ab_bc), 2) - 1 )

evalu P_ac_ab = 0.5 * ( 3*pow(cos(t_ac_ab), 2) - 1 )
evalu P_ac_bc = 0.5 * ( 3*pow(cos(t_ac_bc), 2) - 1 )

evalu P_bc_ab = 0.5 * ( 3*pow(cos(t_bc_ab), 2) - 1 )
evalu P_bc_ac = 0.5 * ( 3*pow(cos(t_bc_ac), 2) - 1 )


##############################################################################
# tensor operators for the dipolar interaction: A_q_w_spins                  #
# e.g. p0iMsAB: A for (q = +0), (w = wi-ws) and (spins = ab)                 #
#      m2iPsAC: A for (q = -2), (w = wi+ws) and (spins = ac)                 #
#                                                                            #
##############################################################################

# q = 0, w = 0, hetero nuclear
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
seto p00AB = (2/sqrt(6)) * az*bz
seto p00AC = (2/sqrt(6)) * az*cz


# q = 0, w = +-(w_i - w_i) = 0, homo nuclear
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
seto p00BC = (2/sqrt(6)) * ( bz*cz - 0.25*b+*c- - 0.25*b-*c+ )


# q = 0, w = +-(w_i - w_s)
# ~~~~~~~~~~~~~~~~~~~~~~~~
seto p0iMsAB = (-0.5/sqrt(6)) * ( b+*a- + b-*a+ )
seto p0iMsAC = (-0.5/sqrt(6)) * ( c+*a- + c-*a+ )


# q = 1, w = w_s
# ~~~~~~~~~~~~~~
seto p1sAB = -0.5 * bz*a+
seto p1sAC = -0.5 * cz*a+


# q = -1, w = w_s
# ~~~~~~~~~~~~~~~
seto m1sAB = 0.5 * bz*a-
seto m1sAC = 0.5 * cz*a-


# q = 1, w = w_i
# ~~~~~~~~~~~~~~
seto p1iBA = -0.5 * b+*az
seto p1iBC = -0.5 * b+*cz

seto p1iCA = -0.5 * c+*az
seto p1iCB = -0.5 * c+*bz


# q = -1, w = w_i
# ~~~~~~~~~~~~~~~
seto m1iBA = 0.5 * b-*az
seto m1iBC = 0.5 * b-*cz

seto m1iCA = 0.5 * c-*az
seto m1iCB = 0.5 * c-*bz


# q = 2, w = +-(w_i + w_i)
# ~~~~~~~~~~~~~~~~~~~~~~~~
seto p2iPiBC = 0.5 * b+*c+


# q = -2, w = +-(w_i + w_i)
# ~~~~~~~~~~~~~~~~~~~~~~~~~
seto m2iPiBC = 0.5 * b-*c-


# q = 2, w = +-(w_i + w_s)
# ~~~~~~~~~~~~~~~~~~~~~~~~
seto p2iPsAB = 0.5 * a+*b+
seto p2iPsAC = 0.5 * a+*c+


# q = -2, w = +-(w_i + w_s)
# ~~~~~~~~~~~~~~~~~~~~~~~~~
seto m2iPsAB = 0.5 * a-*b-
seto m2iPsAC = 0.5 * a-*c-
```

```
##############################################################################
# relaxation operator GAMMA: AUTO-CORRELATED                                 #
#                                                                            #
##############################################################################


# q = -2 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoM2 \
= 0.5 * c0dp_ii * c0dp_ii * P_auto * J_iPi * [p2iPiBC, [m2iPiBC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [p2iPsAB, [m2iPsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [p2iPsAC, [m2iPsAC, ]]


# q = -1 => pow(-1, q) = -1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoM1 \
= -0.5 * c0dp_si * c0dp_si * P_auto * J_s * [p1sAB, [m1sAB, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_s * [p1sAC, [m1sAC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [p1iBA, [m1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iBC, [m1iBC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [p1iCA, [m1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iCB, [m1iCB, ]]


# q = 0 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~
sets Lauto0 \
= 0.5 * c0dp_si * c0dp_si * P_auto * J_0 * [p0OAB, [p0OAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_0 * [p0OAC, [p0OAC, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_auto * J_0 * [p0OBC, [p0OBC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iMs * [p0iMsAB, [p0iMsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iMs * [p0iMsAC, [p0iMsAC, ]]


# q = +1 => pow(-1, q) = -1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoP1 \
= -0.5 * c0dp_si * c0dp_si * P_auto * J_s * [m1sAB, [p1sAB, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_s * [m1sAC, [p1sAC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [m1iBA, [p1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iBC, [p1iBC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [m1iCA, [p1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iCB, [p1iCB, ]]


# q = +2 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoP2 \
= 0.5 * c0dp_ii * c0dp_ii * P_auto * J_iPi * [m2iPiBC, [p2iPiBC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [m2iPsAB, [p2iPsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [m2iPsAC, [p2iPsAC, ]]


# complete L_auto.
# ~~~~~~~~~~~~~~~
sets L_auto \
= LautoM2 + LautoM1 + Lauto0 + LautoP1 + LautoP2


##############################################################################
# relaxation operator GAMMA: CROSS-CORRELATED                                #
#                                                                            #
##############################################################################


# q = -2 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets LcrosM2 \
= 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_iPs * [p2iPsAB, [m2iPsAC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_iPs * [p2iPsAC, [m2iPsAB, ]]


# q = -1 => pow(-1, q) = -1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets L1 \
= -0.5 * c0dp_si * c0dp_si * P_ab_ac * J_s * [p1sAB, [m1sAC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_s * [p1sAC, [m1sAB, ]]

sets L2 \
= -0.5 * c0dp_si * c0dp_ii * P_ab_bc * J_i * [p1iBA, [m1iBC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_i * [p1iBA, [m1iCA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_bc * J_i * [p1iBA, [m1iCB, ]]

sets L3 \
= -0.5 * c0dp_ii * c0dp_si * P_bc_ab * J_i * [p1iBC, [m1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ac * J_i * [p1iBC, [m1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iBC, [m1iCB, ]]
```

```
sets L4\
= -0.5 * c0dp_si * c0dp_si * P_ac_ab * J_i * [p1iCA, [m1iBA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bc * J_i * [p1iCA, [m1iBC, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bc * J_i * [p1iCA, [m1iCB, ]]

sets L5 \
= -0.5 * c0dp_ii * c0dp_si * P_bc_ab * J_i * [p1iCB, [m1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iCB, [m1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ac * J_i * [p1iCB, [m1iCA, ]]

sets LcrosM1 \
= L1 + L2 + L3 + L4 + L5


# q = 0 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~
sets L1 \
= 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_0 * [p0OAB, [p0OAC, ]] \
+ 0.5 * c0dp_si * c0dp_ii * P_ab_bc * J_0 * [p0OAB, [p0OBC, ]]

sets L2 \
= 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_0 * [p0OAC, [p0OAB, ]] \
+ 0.5 * c0dp_si * c0dp_ii * P_ac_bc * J_0 * [p0OAC, [p0OBC, ]]

sets L3 \
= 0.5 * c0dp_ii * c0dp_si * P_bc_ab * J_0 * [p0OBC, [p0OAB, ]] \
+ 0.5 * c0dp_ii * c0dp_si * P_bc_ac * J_0 * [p0OBC, [p0OAC, ]]

sets L4 \
= 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_iMs * [p0iMsAB, [p0iMsAC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_iMs * [p0iMsAC, [p0iMsAB, ]]

sets Lcros0 \
= L1 + L2 + L3 + L4


# q = +1 => pow(-1, q) = -1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets L1 \
= -0.5 * c0dp_si * c0dp_si * P_ab_ac * J_s * [m1sAB, [p1sAC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_s * [m1sAC, [p1sAB, ]]

sets L2 \
= -0.5 * c0dp_si * c0dp_ii * P_ab_bc * J_i * [m1iBA, [p1iBC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_i * [m1iBA, [p1iCA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_bc * J_i * [m1iBA, [p1iCB, ]]

sets L3 \
= -0.5 * c0dp_ii * c0dp_si * P_bc_ab * J_i * [m1iBC, [p1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ac * J_i * [m1iBC, [p1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iBC, [p1iCB, ]]

sets L4 \
= -0.5 * c0dp_si * c0dp_si * P_ac_ab * J_i * [m1iCA, [p1iBA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bc * J_i * [m1iCA, [p1iBC, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bc * J_i * [m1iCA, [p1iCB, ]]

sets L5 \
= -0.5 * c0dp_ii * c0dp_si * P_bc_ab * J_i * [m1iCB, [p1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iCB, [p1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ac * J_i * [m1iCB, [p1iCA, ]]

sets LcrosP1 \
= L1 + L2 + L3 + L4 + L5


# q = +2 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets LcrosP2 \
= 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_iPs * [m2iPsAB, [p2iPsAC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_iPs * [m2iPsAC, [p2iPsAB, ]]


# complete L_cross.
# ~~~~~~~~~~~~~~~~~
sets L_cross \
= LcrosM2 + LcrosM1 + Lcros0 + LcrosP1 + LcrosP2


##############################################################################
# relaxation matrix.                                                         #
#                                                                            #
##############################################################################

# relaxation operator GAMMA
#
sets red \
= L_auto + L_cross
```

```
save Redfield.i2s.Palmer.v8.mat, red

quit
```

13.3 4-Spinsystem

```
##############################################################################
#                                                                            #
# File: redfield.4spin.wtest                                                 #
# Project: create a matlab file containing the Redfield relaxation matrix for  #
#     a 4-spin system
# Date: Sep 13, 2000                                                         #
#                                                                            #
##############################################################################


##############################################################################
# Constants - parameters - defintions                                        #
#                                                                            #
##############################################################################


# DEFINITION: spin system: i3s (i=1H, s=13C, tetraeder geometry ala CR(H3))
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

spins abcd
setisp a,bcd

# gyromagnetic ratio of i (bcd)= 1H - [s*A/kg]
setvar g_i = 2.6752e8

# gyromagnetic ratio of s (a) = 13C - [s*A/kg]
setvar g_s = 6.728e7

# distances between two nuclei - [m]
setvar r_ii = 1.75e-10
setvar r_si = 1.07e-10

# bond's angle - []
setvar theta_isi = 109.44

# correlation time - [s]
setvar tau_c = 1e-9

# CONSTANTS: somes usefull constants
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Planck's h/2pi - [J*s]
setvar hq = 1.0546e-34

# permeability of vacuum - [V*s/(A*m)]
setvar m0 = 1.256637061e-6

# gyromagnetic ratio of 1H - [s*A/kg]
setvar g_1H = 2.6752e8

# CONSTANTS: magnetic field strength
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# spectrometer frequency - [1/s]
setvar v0 = 600e6

# CALCULATED: spin frequencies
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
evalu v_i = v0 * g_i / g_1H
evalu v_s = v0 * g_s / g_1H

evalu w_i = 2*pi * v_i
evalu w_s = 2*pi * v_s


##############################################################################
# spectral densities                                                         #
#                                                                            #
##############################################################################

# spectral density functions
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
evalu J_0 = 2 * tau_c / (5 * ( 1 + pow(0*tau_c, 2) ) )
evalu J_i = 2 * tau_c / (5 * ( 1 + pow(w_i*tau_c, 2) ) )
evalu J_s = 2 * tau_c / (5 * ( 1 + pow(w_s*tau_c, 2) ) )
evalu J_iPi = 2 * tau_c / (5 * ( 1 + pow((w_i + w_i)*tau_c, 2) ) )
evalu J_iMs = 2 * tau_c / (5 * ( 1 + pow((w_i - w_s)*tau_c, 2) ) )
evalu J_iPs = 2 * tau_c / (5 * ( 1 + pow((w_i + w_s)*tau_c, 2) ) )
```

```
# spatial functions for dipolar relaxation
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
evalu c0dp_si = -sqrt(6) * m0 * hq * g_s * g_i / ( 4 * pi * pow(r_si, 3) )
evalu c0dp_ii = -sqrt(6) * m0 * hq * g_i * g_i / ( 4 * pi * pow(r_ii, 3) )


###############################################################################
# Legendre polynoms                                                           #
#                                                                             #
###############################################################################

# angles between dipoles
# ~~~~~~~~~~~~~~~~~~~~~~
setvar t_ab_ac = 109.44
setvar t_ab_ad = 109.44
setvar t_ab_bc = 35.28
setvar t_ab_bd = 35.28
setvar t_ab_cd = 90.0

setvar t_ac_ab = 109.44
setvar t_ac_ad = 109.44
setvar t_ac_bc = 35.28
setvar t_ac_bd = 90.0
setvar t_ac_cd = 35.28

setvar t_ad_ab = 109.44
setvar t_ad_ac = 109.44
setvar t_ad_bc = 90.0
setvar t_ad_bd = 35.28
setvar t_ad_cd = 35.28

setvar t_bc_ab = 35.28
setvar t_bc_ac = 35.28
setvar t_bc_ad = 90.0
setvar t_bc_bd = 60.0
setvar t_bc_cd = 60.0

setvar t_bd_ab = 35.28
setvar t_bd_ac = 90.0
setvar t_bd_ad = 35.28
setvar t_bd_bc = 60.0
setvar t_bd_cd = 60.0

setvar t_cd_ab = 90.0
setvar t_cd_ac = 35.28
setvar t_cd_ad = 35.28
setvar t_cd_bc = 60.0
setvar t_cd_bd = 60.0

# legendre polynoms
# ~~~~~~~~~~~~~~~~~

evalu P_auto = 1

evalu P_ab_ac = 0.5 * ( 3*pow(cos(t_ab_ac), 2) - 1 )
evalu P_ab_ad = 0.5 * ( 3*pow(cos(t_ab_ad), 2) - 1 )
evalu P_ab_bc = 0.5 * ( 3*pow(cos(t_ab_bc), 2) - 1 )
evalu P_ab_bd = 0.5 * ( 3*pow(cos(t_ab_bd), 2) - 1 )
evalu P_ab_cd = 0.5 * ( 3*pow(cos(t_ab_cd), 2) - 1 )

evalu P_ac_ab = 0.5 * ( 3*pow(cos(t_ac_ab), 2) - 1 )
evalu P_ac_ad = 0.5 * ( 3*pow(cos(t_ac_ad), 2) - 1 )
evalu P_ac_bc = 0.5 * ( 3*pow(cos(t_ac_bc), 2) - 1 )
evalu P_ac_bd = 0.5 * ( 3*pow(cos(t_ac_bd), 2) - 1 )
evalu P_ac_cd = 0.5 * ( 3*pow(cos(t_ac_cd), 2) - 1 )

evalu P_ad_ab = 0.5 * ( 3*pow(cos(t_ad_ab), 2) - 1 )
evalu P_ad_ac = 0.5 * ( 3*pow(cos(t_ad_ac), 2) - 1 )
evalu P_ad_bc = 0.5 * ( 3*pow(cos(t_ad_bc), 2) - 1 )
evalu P_ad_bd = 0.5 * ( 3*pow(cos(t_ad_bd), 2) - 1 )
evalu P_ad_cd = 0.5 * ( 3*pow(cos(t_ad_cd), 2) - 1 )

evalu P_bc_ab = 0.5 * ( 3*pow(cos(t_bc_ab), 2) - 1 )
evalu P_bc_ac = 0.5 * ( 3*pow(cos(t_bc_ac), 2) - 1 )
evalu P_bc_ad = 0.5 * ( 3*pow(cos(t_bc_ad), 2) - 1 )
evalu P_bc_bd = 0.5 * ( 3*pow(cos(t_bc_bd), 2) - 1 )
evalu P_bc_cd = 0.5 * ( 3*pow(cos(t_bc_cd), 2) - 1 )

evalu P_bd_ab = 0.5 * ( 3*pow(cos(t_bd_ab), 2) - 1 )
evalu P_bd_ac = 0.5 * ( 3*pow(cos(t_bd_ac), 2) - 1 )
evalu P_bd_ad = 0.5 * ( 3*pow(cos(t_bd_ad), 2) - 1 )
evalu P_bd_bc = 0.5 * ( 3*pow(cos(t_bd_bc), 2) - 1 )
evalu P_bd_cd = 0.5 * ( 3*pow(cos(t_bd_cd), 2) - 1 )

evalu P_cd_ab = 0.5 * ( 3*pow(cos(t_cd_ab), 2) - 1 )
evalu P_cd_ac = 0.5 * ( 3*pow(cos(t_cd_ac), 2) - 1 )
evalu P_cd_ad = 0.5 * ( 3*pow(cos(t_cd_ad), 2) - 1 )
evalu P_cd_bc = 0.5 * ( 3*pow(cos(t_cd_bc), 2) - 1 )
```

```
evalu P_cd_bd = 0.5 * ( 3*pow(cos(t_cd_bd), 2) - 1 )


###############################################################################
# tensor operators for the dipolar interaction: A_q_w_spins                   #
# e.g. p0iMsAB: A for (q = +0), (w = wi-ws) and (spins = ab)                  #
#      m2iPsAC: A for (q = -2), (w = wi+ws) and (spins = ac)                  #
#                                                                             #
###############################################################################

# q = 0, w = 0, hetero nuclear
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
seto p0OAB = (2/sqrt(6)) * az*bz
seto p0OAC = (2/sqrt(6)) * az*cz
seto p0OAD = (2/sqrt(6)) * az*dz


# q = 0, w = +-(w_i - w_i) = 0, homo nuclear
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
seto p0OBC = (2/sqrt(6)) * ( bz*cz - 0.25*b+*c- - 0.25*b-*c+ )
seto p0OBD = (2/sqrt(6)) * ( bz*dz - 0.25*b+*d- - 0.25*b-*d+ )
seto p0OCD = (2/sqrt(6)) * ( cz*dz - 0.25*c+*d- - 0.25*c-*d+ )


# q = 0, w = +-(w_i - w_s)
# ~~~~~~~~~~~~~~~~~~~~~~~~
seto p0iMsAB = (-0.5/sqrt(6)) * ( b+*a- + b-*a+ )
seto p0iMsAC = (-0.5/sqrt(6)) * ( c+*a- + c-*a+ )
seto p0iMsAD = (-0.5/sqrt(6)) * ( d+*a- + d-*a+ )


# q = 1, w = w_s
# ~~~~~~~~~~~~~~
seto p1sAB = -0.5 * bz*a+
seto p1sAC = -0.5 * cz*a+
seto p1sAD = -0.5 * dz*a+


# q = -1, w = w_s
# ~~~~~~~~~~~~~~~
seto m1sAB = 0.5 * bz*a-
seto m1sAC = 0.5 * cz*a-
seto m1sAD = 0.5 * dz*a-



# q = 1, w = w_i
# ~~~~~~~~~~~~~~
seto p1iBA = -0.5 * b+*az
seto p1iBC = -0.5 * b+*cz
seto p1iBD = -0.5 * b+*dz

seto p1iCA = -0.5 * c+*az
seto p1iCB = -0.5 * c+*bz
seto p1iCD = -0.5 * c+*dz

seto p1iDA = -0.5 * d+*az
seto p1iDB = -0.5 * d+*bz
seto p1iDC = -0.5 * d+*cz


# q = -1, w = w_i
# ~~~~~~~~~~~~~~~
seto m1iBA = 0.5 * b-*az
seto m1iBC = 0.5 * b-*cz
seto m1iBD = 0.5 * b-*dz

seto m1iCA = 0.5 * c-*az
seto m1iCB = 0.5 * c-*bz
seto m1iCD = 0.5 * c-*dz

seto m1iDA = 0.5 * d-*az
seto m1iDB = 0.5 * d-*bz
seto m1iDC = 0.5 * d-*cz


# q = 2, w = +-(w_i + w_i)
# ~~~~~~~~~~~~~~~~~~~~~~~~
seto p2iPiBC = 0.5 * b+*c+
seto p2iPiBD = 0.5 * b+*d+
seto p2iPiCD = 0.5 * c+*d+


# q = -2, w = +-(w_i + w_i)
# ~~~~~~~~~~~~~~~~~~~~~~~~~
seto m2iPiBC = 0.5 * b-*c-
seto m2iPiBD = 0.5 * b-*d-
seto m2iPiCD = 0.5 * c-*d-
```

```
# q = 2, w = +-(w_i + w_s)
# ~~~~~~~~~~~~~~~~~~~~~~~
seto p2iPsAB = 0.5 * a+*b+
seto p2iPsAC = 0.5 * a+*c+
seto p2iPsAD = 0.5 * a+*d+


# q = -2, w = +-(w_i + w_s)
# ~~~~~~~~~~~~~~~~~~~~~~~
seto m2iPsAB = 0.5 * a-*b-
seto m2iPsAC = 0.5 * a-*c-
seto m2iPsAD = 0.5 * a-*d-


##############################################################################
# relaxation operator GAMMA: AUTO-CORRELATED                                 #
#                                                                            #
##############################################################################


# q = -2 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoM2 \
= 0.5 * c0dp_ii * c0dp_ii * P_auto * J_iPi * [p2iPiBC, [m2iPiBC, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_auto * J_iPi * [p2iPiBD, [m2iPiBD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_auto * J_iPi * [p2iPiCD, [m2iPiCD, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [p2iPsAB, [m2iPsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [p2iPsAC, [m2iPsAC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [p2iPsAD, [m2iPsAD, ]]


# q = -1 => pow(-1, q) = -1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoM1 \
= -0.5 * c0dp_si * c0dp_si * P_auto * J_s * [p1sAB, [m1sAB, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_s * [p1sAC, [m1sAC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_s * [p1sAD, [m1sAD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [p1iBA, [m1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iBC, [m1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iBD, [m1iBD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [p1iCA, [m1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iCB, [m1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iCD, [m1iCD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [p1iDA, [m1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iDB, [m1iDB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iDC, [m1iDC, ]]


# q = 0 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets Lauto0 \
= 0.5 * c0dp_si * c0dp_si * P_auto * J_0 * [p00AB, [p00AB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_0 * [p00AC, [p00AC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_0 * [p00AD, [p00AD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_auto * J_0 * [p00BC, [p00BC, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_auto * J_0 * [p00BD, [p00BD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_auto * J_0 * [p00CD, [p00CD, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iMs * [p0iMsAB, [p0iMsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iMs * [p0iMsAC, [p0iMsAC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iMs * [p0iMsAD, [p0iMsAD, ]]


# q = +1 => pow(-1, q) = -1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoP1 \
= -0.5 * c0dp_si * c0dp_si * P_auto * J_s * [m1sAB, [p1sAB, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_s * [m1sAC, [p1sAC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_s * [m1sAD, [p1sAD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [m1iBA, [p1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iBC, [p1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iBD, [p1iBD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [m1iCA, [p1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iCB, [p1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iCD, [p1iCD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_auto * J_i * [m1iDA, [p1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iDB, [p1iDB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iDC, [p1iDC, ]]


# q = +2 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
sets LautoP2 \
= 0.5 * c0dp_ii * c0dp_ii * P_auto * J_iPi * [m2iPiBC, [p2iPiBC, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_auto * J_iPi * [m2iPiBD, [p2iPiBD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_auto * J_iPi * [m2iPiCD, [p2iPiCD, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [m2iPsAB, [p2iPsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [m2iPsAC, [p2iPsAC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_auto * J_iPs * [m2iPsAD, [p2iPsAD, ]]
```

```
# complete L_auto.
# ~~~~~~~~~~~~~~~~
sets L_auto \
= LautoM2 + LautoM1 + Lauto0 + LautoP1 + LautoP2


##############################################################################
# relaxation operator GAMMA: CROSS-CORRELATED                                #
#                                                                            #
##############################################################################


# q = -2 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
sets LcrosM2 \
= 0.5 * c0dp_ii * c0dp_ii * P_bc_bd * J_iPi * [p2iPiBC, [m2iPiBD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_bc_cd * J_iPi * [p2iPiBC, [m2iPiCD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_bd_bc * J_iPi * [p2iPiBD, [m2iPiBC, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_bd_cd * J_iPi * [p2iPiBD, [m2iPiCD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_cd_bc * J_iPi * [p2iPiCD, [m2iPiBC, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_cd_bd * J_iPi * [p2iPiCD, [m2iPiBD, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_iPs * [p2iPsAB, [m2iPsAC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ab_ad * J_iPs * [p2iPsAB, [m2iPsAD, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_iPs * [p2iPsAC, [m2iPsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ac_ad * J_iPs * [p2iPsAC, [m2iPsAD, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ad_ab * J_iPs * [p2iPsAD, [m2iPsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ad_ac * J_iPs * [p2iPsAD, [m2iPsAC, ]]


# q = -1 => pow(-1, q) = -1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
sets L1 \
= -0.5 * c0dp_si * c0dp_si * P_ab_ac * J_s * [p1sAB, [m1sAC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ab_ad * J_s * [p1sAB, [m1sAD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_s * [p1sAC, [m1sAB, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ac_ad * J_s * [p1sAC, [m1sAD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ad_ab * J_s * [p1sAD, [m1sAB, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ad_ac * J_s * [p1sAD, [m1sAC, ]]

sets L2 \
= -0.5 * c0dp_si * c0dp_ii * P_ab_bc * J_i * [p1iBA, [m1iBC, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_bd * J_i * [p1iBA, [m1iBD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_i * [p1iBA, [m1iCA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_bc * J_i * [p1iBA, [m1iCB, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_cd * J_i * [p1iBA, [m1iCD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ab_ad * J_i * [p1iBA, [m1iDA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_bd * J_i * [p1iBA, [m1iDB, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_cd * J_i * [p1iBA, [m1iDC, ]]

sets L3 \
= -0.5 * c0dp_ii * c0dp_si * P_bc_ab * J_i * [p1iBC, [m1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_bd * J_i * [p1iBC, [m1iBD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ac * J_i * [p1iBC, [m1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iBC, [m1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_cd * J_i * [p1iBC, [m1iCD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ad * J_i * [p1iBC, [m1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_bd * J_i * [p1iBC, [m1iDB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_cd * J_i * [p1iBC, [m1iDC, ]]

sets L4 \
= -0.5 * c0dp_ii * c0dp_si * P_bd_ab * J_i * [p1iBD, [m1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_bc * J_i * [p1iBD, [m1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bd_ac * J_i * [p1iBD, [m1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_bc * J_i * [p1iBD, [m1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_cd * J_i * [p1iBD, [m1iCD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bd_ad * J_i * [p1iBD, [m1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iBD, [m1iDB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_cd * J_i * [p1iBD, [m1iDC, ]]

sets L5\
= -0.5 * c0dp_si * c0dp_si * P_ac_ab * J_i * [p1iCA, [m1iBA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bc * J_i * [p1iCA, [m1iBC, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bd * J_i * [p1iCA, [m1iBD, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bc * J_i * [p1iCA, [m1iCB, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_cd * J_i * [p1iCA, [m1iCD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ac_ad * J_i * [p1iCA, [m1iDA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bd * J_i * [p1iCA, [m1iDB, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_cd * J_i * [p1iCA, [m1iDC, ]]

sets L6 \
= -0.5 * c0dp_ii * c0dp_si * P_bc_ab * J_i * [p1iCB, [m1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iCB, [m1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_bd * J_i * [p1iCB, [m1iBD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ac * J_i * [p1iCB, [m1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_cd * J_i * [p1iCB, [m1iCD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ad * J_i * [p1iCB, [m1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_bd * J_i * [p1iCB, [m1iDB, ]] \
```

```
- 0.5 * c0dp_ii * c0dp_ii * P_bc_cd * J_i * [p1iCB, [m1iDC, ]]

sets L7 \
= -0.5 * c0dp_ii * c0dp_si * P_cd_ab * J_i * [p1iCD, [m1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bc * J_i * [p1iCD, [m1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bd * J_i * [p1iCD, [m1iBD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_cd_ac * J_i * [p1iCD, [m1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bc * J_i * [p1iCD, [m1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_cd_ad * J_i * [p1iCD, [m1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bd * J_i * [p1iCD, [m1iDB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iCD, [m1iDC, ]]

sets L8 \
= -0.5 * c0dp_si * c0dp_si * P_ad_ab * J_i * [p1iDA, [m1iBA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ad_bc * J_i * [p1iDA, [m1iBC, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ad_bd * J_i * [p1iDA, [m1iBD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ad_ac * J_i * [p1iDA, [m1iCA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ad_bc * J_i * [p1iDA, [m1iCB, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ad_cd * J_i * [p1iDA, [m1iCD, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ad_bd * J_i * [p1iDA, [m1iDB, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ad_cd * J_i * [p1iDA, [m1iDC, ]]

sets L9 \
= -0.5 * c0dp_ii * c0dp_si * P_bd_ab * J_i * [p1iDB, [m1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_bc * J_i * [p1iDB, [m1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iDB, [m1iBD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bd_ac * J_i * [p1iDB, [m1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_bc * J_i * [p1iDB, [m1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_cd * J_i * [p1iDB, [m1iCD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bd_ad * J_i * [p1iDB, [m1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_cd * J_i * [p1iDB, [m1iDC, ]]

sets L10 \
= -0.5 * c0dp_ii * c0dp_si * P_cd_ab * J_i * [p1iDC, [m1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bc * J_i * [p1iDC, [m1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bd * J_i * [p1iDC, [m1iBD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_cd_ac * J_i * [p1iDC, [m1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bc * J_i * [p1iDC, [m1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [p1iDC, [m1iCD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_cd_ad * J_i * [p1iDC, [m1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bd * J_i * [p1iDC, [m1iDB, ]]

sets LcrosM1 \
= L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10


# q = 0 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~
sets L1 \
= 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_0 * [p00AB, [p00AC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ab_ad * J_0 * [p00AB, [p00AD, ]] \
+ 0.5 * c0dp_si * c0dp_ii * P_ab_bc * J_0 * [p00AB, [p00BC, ]] \
+ 0.5 * c0dp_si * c0dp_ii * P_ab_bd * J_0 * [p00AB, [p00BD, ]] \
+ 0.5 * c0dp_si * c0dp_ii * P_ab_cd * J_0 * [p00AB, [p00CD, ]]

sets L2 \
= 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_0 * [p00AC, [p00AB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ac_ad * J_0 * [p00AC, [p00AD, ]] \
+ 0.5 * c0dp_si * c0dp_ii * P_ac_bc * J_0 * [p00AC, [p00BC, ]] \
+ 0.5 * c0dp_si * c0dp_ii * P_ac_bd * J_0 * [p00AC, [p00BD, ]] \
+ 0.5 * c0dp_si * c0dp_ii * P_ac_cd * J_0 * [p00AC, [p00CD, ]]

sets L3 \
= 0.5 * c0dp_si * c0dp_si * P_ad_ab * J_0 * [p00AD, [p00AB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ad_ac * J_0 * [p00AD, [p00AC, ]] \
+ 0.5 * c0dp_si * c0dp_ii * P_ad_bc * J_0 * [p00AD, [p00BC, ]] \
+ 0.5 * c0dp_si * c0dp_ii * P_ad_bd * J_0 * [p00AD, [p00BD, ]] \
+ 0.5 * c0dp_si * c0dp_ii * P_ad_cd * J_0 * [p00AD, [p00CD, ]]

sets L4 \
= 0.5 * c0dp_ii * c0dp_si * P_bc_ab * J_0 * [p00BC, [p00AB, ]] \
+ 0.5 * c0dp_ii * c0dp_si * P_bc_ac * J_0 * [p00BC, [p00AC, ]] \
+ 0.5 * c0dp_ii * c0dp_si * P_bc_ad * J_0 * [p00BC, [p00AD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_bc_bd * J_0 * [p00BC, [p00BD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_bc_cd * J_0 * [p00BC, [p00CD, ]]

sets L5 \
= 0.5 * c0dp_ii * c0dp_si * P_bd_ab * J_0 * [p00BD, [p00AB, ]] \
+ 0.5 * c0dp_ii * c0dp_si * P_bd_ac * J_0 * [p00BD, [p00AC, ]] \
+ 0.5 * c0dp_ii * c0dp_si * P_bd_ad * J_0 * [p00BD, [p00AD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_bd_bc * J_0 * [p00BD, [p00BC, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_bd_cd * J_0 * [p00BD, [p00CD, ]]

sets L6 \
= 0.5 * c0dp_ii * c0dp_si * P_cd_ab * J_0 * [p00CD, [p00AB, ]] \
+ 0.5 * c0dp_ii * c0dp_si * P_cd_ac * J_0 * [p00CD, [p00AC, ]] \
+ 0.5 * c0dp_ii * c0dp_si * P_cd_ad * J_0 * [p00CD, [p00AD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_cd_bc * J_0 * [p00CD, [p00BC, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_cd_bd * J_0 * [p00CD, [p00BD, ]]
```

```
sets L7 \
= 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_iMs * [pOiMsAB, [pOiMsAC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ab_ad * J_iMs * [pOiMsAB, [pOiMsAD, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_iMs * [pOiMsAC, [pOiMsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ac_ad * J_iMs * [pOiMsAC, [pOiMsAD, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ad_ab * J_iMs * [pOiMsAD, [pOiMsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ad_ac * J_iMs * [pOiMsAD, [pOiMsAC, ]]

sets Lcros0 \
= L1 + L2 + L3 + L4 + L5 + L6 + L7


# q = +1 => pow(-1, q) = -1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets L1 \
= -0.5 * c0dp_si * c0dp_si * P_ab_ac * J_s * [m1sAB, [p1sAC, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ab_ad * J_s * [m1sAB, [p1sAD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_s * [m1sAC, [p1sAB, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ac_ad * J_s * [m1sAC, [p1sAD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ad_ab * J_s * [m1sAD, [p1sAB, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ad_ac * J_s * [m1sAD, [p1sAC, ]]

sets L2 \
= -0.5 * c0dp_si * c0dp_ii * P_ab_bc * J_i * [m1iBA, [p1iBC, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_bd * J_i * [m1iBA, [p1iBD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_i * [m1iBA, [p1iCA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_bc * J_i * [m1iBA, [p1iCB, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_cd * J_i * [m1iBA, [p1iCD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ab_ad * J_i * [m1iBA, [p1iDA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_bd * J_i * [m1iBA, [p1iDB, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ab_cd * J_i * [m1iBA, [p1iDC, ]]

sets L3 \
= -0.5 * c0dp_ii * c0dp_si * P_bc_ab * J_i * [m1iBC, [p1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_bd * J_i * [m1iBC, [p1iBD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ac * J_i * [m1iBC, [p1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iBC, [p1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_cd * J_i * [m1iBC, [p1iCD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ad * J_i * [m1iBC, [p1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_bd * J_i * [m1iBC, [p1iDB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_cd * J_i * [m1iBC, [p1iDC, ]]

sets L4 \
= -0.5 * c0dp_ii * c0dp_si * P_bd_ab * J_i * [m1iBD, [p1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_bc * J_i * [m1iBD, [p1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bd_ac * J_i * [m1iBD, [p1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_bc * J_i * [m1iBD, [p1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_cd * J_i * [m1iBD, [p1iCD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bd_ad * J_i * [m1iBD, [p1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iBD, [p1iDB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_cd * J_i * [m1iBD, [p1iDC, ]]

sets L5 \
= -0.5 * c0dp_si * c0dp_si * P_ac_ab * J_i * [m1iCA, [p1iBA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bc * J_i * [m1iCA, [p1iBC, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bd * J_i * [m1iCA, [p1iBD, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bc * J_i * [m1iCA, [p1iCB, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_cd * J_i * [m1iCA, [p1iCD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ac_ad * J_i * [m1iCA, [p1iDA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_bd * J_i * [m1iCA, [p1iDB, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ac_cd * J_i * [m1iCA, [p1iDC, ]]

sets L6 \
= -0.5 * c0dp_ii * c0dp_si * P_bc_ab * J_i * [m1iCB, [p1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iCB, [p1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_bd * J_i * [m1iCB, [p1iBD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ac * J_i * [m1iCB, [p1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_cd * J_i * [m1iCB, [p1iCD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bc_ad * J_i * [m1iCB, [p1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_bd * J_i * [m1iCB, [p1iDB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bc_cd * J_i * [m1iCB, [p1iDC, ]]

sets L7 \
= -0.5 * c0dp_ii * c0dp_si * P_cd_ab * J_i * [m1iCD, [p1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bc * J_i * [m1iCD, [p1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bd * J_i * [m1iCD, [p1iBD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_cd_ac * J_i * [m1iCD, [p1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bc * J_i * [m1iCD, [p1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_cd_ad * J_i * [m1iCD, [p1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bd * J_i * [m1iCD, [p1iDB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iCD, [p1iDC, ]]

sets L8 \
= -0.5 * c0dp_si * c0dp_si * P_ad_ab * J_i * [m1iDA, [p1iBA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ad_bc * J_i * [m1iDA, [p1iBC, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ad_bd * J_i * [m1iDA, [p1iBD, ]] \
- 0.5 * c0dp_si * c0dp_si * P_ad_ac * J_i * [m1iDA, [p1iCA, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ad_bc * J_i * [m1iDA, [p1iCB, ]] \
```

```
- 0.5 * c0dp_si * c0dp_ii * P_ad_cd * J_i * [m1iDA, [p1iCD, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ad_bd * J_i * [m1iDA, [p1iDB, ]] \
- 0.5 * c0dp_si * c0dp_ii * P_ad_cd * J_i * [m1iDA, [p1iDC, ]]

sets L9 \
= -0.5 * c0dp_ii * c0dp_si * P_bd_ab * J_i * [m1iDB, [p1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_bc * J_i * [m1iDB, [p1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iDB, [p1iBD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bd_ac * J_i * [m1iDB, [p1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_bc * J_i * [m1iDB, [p1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_cd * J_i * [m1iDB, [p1iCD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_bd_ad * J_i * [m1iDB, [p1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_bd_cd * J_i * [m1iDB, [p1iDC, ]]

sets L10 \
= -0.5 * c0dp_ii * c0dp_si * P_cd_ab * J_i * [m1iDC, [p1iBA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bc * J_i * [m1iDC, [p1iBC, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bd * J_i * [m1iDC, [p1iBD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_cd_ac * J_i * [m1iDC, [p1iCA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bc * J_i * [m1iDC, [p1iCB, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_auto * J_i * [m1iDC, [p1iCD, ]] \
- 0.5 * c0dp_ii * c0dp_si * P_cd_ad * J_i * [m1iDC, [p1iDA, ]] \
- 0.5 * c0dp_ii * c0dp_ii * P_cd_bd * J_i * [m1iDC, [p1iDB, ]]

sets LcrosP1 \
= L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10


# q = +2 => pow(-1, q) = +1.
# ~~~~~~~~~~~~~~~~~~~~~~~~~
sets LcrosP2 \
= 0.5 * c0dp_ii * c0dp_ii * P_bc_bd * J_iPi * [m2iPiBC, [p2iPiBD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_bc_cd * J_iPi * [m2iPiBC, [p2iPiCD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_bd_bc * J_iPi * [m2iPiBD, [p2iPiBC, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_bd_cd * J_iPi * [m2iPiBD, [p2iPiCD, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_cd_bc * J_iPi * [m2iPiCD, [p2iPiBC, ]] \
+ 0.5 * c0dp_ii * c0dp_ii * P_cd_bd * J_iPi * [m2iPiCD, [p2iPiBD, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ab_ac * J_iPs * [m2iPsAB, [p2iPsAC, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ab_ad * J_iPs * [m2iPsAB, [p2iPsAD, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ac_ab * J_iPs * [m2iPsAC, [p2iPsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ac_ad * J_iPs * [m2iPsAC, [p2iPsAD, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ad_ab * J_iPs * [m2iPsAD, [p2iPsAB, ]] \
+ 0.5 * c0dp_si * c0dp_si * P_ad_ac * J_iPs * [m2iPsAD, [p2iPsAC, ]]


# complete L_cross.
# ~~~~~~~~~~~~~~~~
sets L_cross \
= LcrosM2 + LcrosM1 + Lcros0 + LcrosP1 + LcrosP2


##############################################################################
# relaxation matrix.                                                         #
#                                                                            #
##############################################################################

# relaxation operator GAMMA
#
sets red \
= L_auto + L_cross
save Redfield.i3s.Palmer.v8.mat, red

quit
```

Kapitel 14

Hartpuls-Näherung für ROPE-INEPT

Wie in Kapitel 6 angemerkt, wurde in [42] eine analytische Lösung des Transfer-Schemas (6.16) erarbeitet. Durch diese ist die Berechnung der geforderten Rotationen möglich. Da bei NMR-Spektrometern ohne lineare Verstärker solche stetigen Rotationen schwierig zu implementieren sind, wurde ein Näherungs-Verfahren entwickelt, durch das eine Sequenz aus harten Pulsen berechnet werden kann. Dieses Verfahren wurde in einen Satz von Matlab-Programmen umgesetzt, die hier kurz vorgestellt werden sollen.

14.1 Erzeugung einer Hartpuls-Näherung

Die in diesem Abschnitt beschriebenen Skripte können dazu benutzt werden, eine Hartpuls-Näherung für ein ROPE-INEPT-Experiment zu berechnen. Die berechnete Pulssequenz ist nicht breitbandig. Die Berechnung wird über die Routine `psa4` gestartet, alle weiteren sind Subroutinen, die intern aufgerufen werden.

Die Funktions-Parameter sind die Kopplungskonstante `J` in Hertz, die transversale Relaxationsrate `T2` in Sekunden, die Länge einer Rotationsperiode `tau` (in erster Näherung die $\tau_{\mathrm{mix}}/2$) und die Anzahl der zu erzeugenden Pulse pro Rotationsphase.

Die Rückgabewerte entsprechen mit `p1` bzw. `p3` den Pulswinkeln der genäherten Sequenz in Einheiten von $\pi/2$. In `d1`, `d2` und `d3` werden die entsprechenden Delays in μs zurückgegeben. Die hierbei verwendete Indizierung der Variablen entspricht den drei Phasen der Sequenz. Z.B. wird Phase I

durch `p1` und `d1` beschrieben. Die Rückgabewerte können dazu verwendet werden, aus dieser schmalbandigen Näherung eine breitbandige zu berechnen (s. dazu Abschnitt 14.2).

Umrechnung der diskretisierten Kontroll-Werte in Pulswinkel:

```
function [p1,p3,d1,d2,d3] = psa4( J, T2, tau, N )

    D = 1/(pi*T2);

    [t1,u1,t2,u2] = ps4(J,D,tau,N);

    [tmp, maxj] = size(u1);
    idx = 1;

    for j = 1:maxj
        alpha_y(idx) = real( asin( u1(idx) ) );
        alpha_x(idx) = real( acos( u2(idx) ) );
        idx = idx + 1;
    end;

    disp(sprintf('# of pulses: %d',2.*maxj));

    disp('Testing pulse sequence on spin matrices:');

    % definition of the pauli-matrices
    ix = 0.5 * [0 1; 1 0];
    iy = 0.5 * [0 -i; i 0];
    iz = 0.5 * [1 0; 0 -1];
    id = [1 0; 0 1];

    % build up 2-spin system
    Ix = kron(ix,id);
    Sx = kron(id,ix);
    Iy = kron(iy,id);
    Sy = kron(id,iy);
    Iz = kron(iz,id);
    Sz = kron(id,iz);

    H0 = sOp( 2*pi * J * Iz*Sz );
    sIz = sKet( Iz );
    sIx = sKet( Ix );
    sIySz = sKet( 2*Iy*Sz );
    sIzSz = sKet( 2*Iz*Sz );

    % build relaxation operator
    A = 2*Iz*Sz;
    Id = kron(id,id);
    red = pi*D*( kron(A*A, Id) + kron(Id, A'*A') - kron(A, A') - kron(A,A') );

    % define spin states (2-spin system's norm is 1, thus no normalization)
    rho_start = sKet( Iz );
    rho_target = sKet( 2*Iz*Sz );

    % run sequence
    disp( sprintf('\n*** PHASE I ...') );
    idx = 1;
    rho_current = rho_start;
    olIz(idx) = real(rho_current' * sIz);
    olIx(idx) = real(rho_current' * sIx);
    olISz(idx) = real(rho_current' * sISz);
    olIzSz(idx) = real(rho_current' * sIzSz);
    idx = idx + 1;
    k = 1;
    alpha = alpha_y(1);
    for j = 1:(maxj-1)
        disp( sprintf( '\tpulse: (phase: y   alpha: %f [deg] == %f [pi/2])', alpha .* (360/(2*pi)), alpha*2/pi) );
        p1(k) = alpha*2/pi;
        rho_current = expm( -i* sOp(alpha * Iy)  ) * rho_current;
        delta_t = t1(j+1)-t1(j);
        disp( sprintf( '\tdelay: %f [usec]', delta_t*1e6 ) );
        d1(k) = delta_t*1e6;
        rho_current = revolveKet( rho_current, H0, red, delta_t );
        olIz(idx) = real(rho_current' * sIz);
        olIx(idx) = real(rho_current' * sIx);
        olISz(idx) = real(rho_current' * sISz);
        olIzSz(idx) = real(rho_current' * sIzSz);
        % alpha of next pulse = alpha (as should be) - alpha (which we have)
        alpha = alpha_y(j+1) - ( atan( olIx(idx) / olIz(idx) ) );
        idx = idx + 1;
        k = k + 1;
    end;
```

```
    % alpha of next pulse = alpha (as should be) - alpha (which we have)
    alpha = alpha_y(maxj) - ( atan( olIx(idx-1) / olIz(idx-1) ) );
    disp( sprintf( '\tpulse: (phase: y   alpha: %f [deg] == %f [pi/2])', alpha .* (360/(2*pi)), alpha*2/pi) );
    p1(k) = alpha*2/pi;
    rho_current = expm( -i* sOp(alpha * Iy)  ) * rho_current;

    disp( sprintf('\n*** PHASE OF VOID ...') );
    delta_t = t2(1)-t1(maxj);
    d2 = delta_t*1e6;
    disp( sprintf( '\tdelay: %f [usec]', delta_t*1e6 ) );
    rho_current = revolveKet( rho_current, H0, red, delta_t );
    olIz(idx) = real(rho_current' * sIz);
    olIx(idx) = real(rho_current' * sIx);
    olIySz(idx) = real(rho_current' * sIySz);
    olIzSz(idx) = real(rho_current' * sIzSz);
    idx = idx + 1;

   disp( sprintf('\n*** PHASE II ...') );
   alpha = alpha_x(1);
   k = 1;
    for j = 1:(maxj-1)
        disp( sprintf( '\tpulse: (phase: x   alpha: %f [deg] == %f [pi/2])', alpha .* (360/(2*pi)), alpha*2/pi) );
        p3(k) = alpha*2/pi;
        rho_current = expm( -i* sOp(alpha * Ix)  ) * rho_current;
        delta_t = t2(j+1)-t2(j);
        disp( sprintf( '\tdelay: %f [usec]', delta_t*1e6 ) );
        d3(k) = delta_t*1e6;
        rho_current = revolveKet( rho_current, H0, red, delta_t );
        olIz(idx) = real(rho_current' * sIz);
        olIx(idx) = real(rho_current' * sIx);
        olIySz(idx) = real(rho_current' * sIySz);
        olIzSz(idx) = real(rho_current' * sIzSz);
        alpha = alpha_x(j+1) - atan( olIzSz(idx) / olIySz(idx) );
        idx = idx + 1;
        k = k + 1;
    end;

    % correction in alpha via last u2...
    alpha = alpha_x(maxj) - atan( olIzSz(idx-1) / olIySz(idx-1) );
    % ... concat w/ last 90-alpha_soll(max)
    alpha = 0.5*pi-alpha_x(maxj);
    disp( sprintf( '\tpulse: (phase: x   alpha: %f [deg] == %f [pi/2])', alpha .* (360/(2*pi)), alpha*2/pi) );
    p3(k) = alpha*2/pi;
    rho_current = expm( -i* sOp(alpha * Ix)  ) * rho_current;
    olIz(idx) = real(rho_current' * sIz);
    olIx(idx) = real(rho_current' * sIx);
    olIySz(idx) = real(rho_current' * sIySz);
    olIzSz(idx) = real(rho_current' * sIzSz);

    % calculate transfer amplitude
    te = rho_current' * rho_target;
    disp( sprintf( 'Pulse sequence gives transfer amplitude of: %f', te) );
```

Umrechnung der analytischen Kontroll-Vorschrift in diskrete Kontroll-Werte:

```
function [t1, u1, t2, u2] = ps4( J, D, tau, N )

    alpha = D / J;
    beta = asinh( alpha );

    kappa = 1 + 2*alpha^2 - 2*alpha * sqrt( 1 + alpha^2 ) * coth( pi*J* sqrt( 1 + alpha^2 )*tau + 2*beta );

    theta1 = acot( (1-kappa) / (2*alpha*kappa) );
    theta2 = atan( (1-kappa) / (2*alpha) );

    T = 2*tau + (theta2 - theta1)/(pi*J);
    disp( sprintf( 'Total time of sequence: %f [sec]', T) );

    eta = ( exp( alpha * ( theta1-theta2 ) ) * ( 1 - alpha * sin( 2*theta2 ) ) ) / sin( theta1 + theta2 );

    A = sinh( phi(alpha, J, 0.5*tau) );
    B = cosh( phi(alpha, J, tau) );

    R1 = eta / sqrt( (tan(theta2))^2 + kappa );
    R2 = eta / sqrt( 1 + kappa/( (tan(theta2))^2 ) );

    idx = 1; idxmax = N+1;
    for t = 0:tau/(N-1):tau
        Phi=phi(alpha,J,t);

        u1(idx) = sqrt( R1^2*( 1 + cosh(Phi) ) / ( B*R1^2 + 2*A^2*R2^2 - R1^2*cosh(Phi) ) );
        t1(idx) = t;

        u2(idxmax-idx) = u1(idx);
```

```
        t2(idxmax-idx) = (T - t);

        idx = idx + 1;
    end;

    disp( sprintf( '\tTransfer amplitude: %f [-]', eta) );
    etaINEPT = aINEPTmax(J,alpha);
    disp( sprintf('\tAnalytical Transfer Amplitude INEPT: %f [-]', etaINEPT ) );
    disp( sprintf('\tGain of OCT: %f [%%]', 100*eta/etaINEPT - 100) );
    etaINEPT = sINEPTmax(J,alpha);
    disp( sprintf('\tSimulated Transfer Amplitude INEPT: %f [-]', etaINEPT ) );
    disp( sprintf('\tGain of OCT: %f [%%]', 100*eta/etaINEPT - 100) );
```

Analytische Berechnung der Transferamplitude von INEPT:

```
function [eta] = aINEPTmax( J, alpha )

    tmax = 1/(2*J);
    N = 1024;
    dt = tmax/(N-1);

    idx = 1;
    for t = 0:dt:tmax
        etatmp(idx) = exp( -alpha.*t *pi*J).*sin(t *pi*J);
        idx = idx + 1;
    end;

    eta = max(etatmp);
```

Berechnung der Transferamplitude von INEPT durch Simulation:

```
function [eta] = sINEPTmax( J, alpha )

    tmax = 1/(2*J);
    N = 1024;
    dt = tmax/(N-1);

    D = alpha*J;

    % definition of the pauli-matrices
    ix = 0.5 * [0 1; 1 0];
    iy = 0.5 * [0 -i; i 0];
    iz = 0.5 * [1 0; 0 -1];
    id = [1 0; 0 1];
    % build up 2-spin system
    Ix = kron(ix,id);
    Sx = kron(id,ix);
    Iy = kron(iy,id);
    Sy = kron(id,iy);
    Iz = kron(iz,id);
    Sz = kron(id,iz);

    H0 = sOp( 2*pi * J * Iz*Sz );
    sIz = sKet( Iz );
    sIx = sKet( Ix );
    sIySz = sKet( 2*Iy*Sz );
    sIzSz = sKet( 2*Iz*Sz );

    % build relaxation operator
    DD = 2*Iz*Sz;
    Id = kron(id,id);
    red = pi*D*( kron(DD*DD, Id) + kron(Id, DD'*DD') - kron(DD, DD') - kron(DD,DD') );

    % define states (2-spin system's norm is 1, thus no normalization)
    rho_start = sKet( Iz );
    rho_target = sKet( 2*Iy*Sz );

    idx = 1;
    for t = 0:dt:tmax
        rho_current = expm( -i* sOp(0.5*pi * Iy)  ) * rho_start;
        rho_current = revolveKet( rho_current, H0, red, t );

        etatmp(idx) = rho_current' * rho_target;
        time(idx) = t;

        idx = idx + 1;
    end;

    eta = max(etatmp);
```

14.2 Erzeugung einer breitbandigen Hartpuls-Näherung

Hat man mit `psa4` eine Hartpuls-Näherung erhalten, so kann man diese unter Verwendung des Skriptes `simseqbb` in eine breitbandiger Version umrechnen lassen. Hierzu werden die mit `psa4` berechneten Rückgabewerte `p1, p3, d1, d2, d3,` als Teil der Eingabewerte verwendet. Darüberhinaus muß wiederum die Kopplungskonstante `J` in Hertz und die transversale Relaxationsrate `T2` in Sekunden definiert werden. Neben der Erzeugung der breitbandigen Variante erzeugt das Programm auch Daten für einen Offset-Plot, für den bestimmte Bedingungen angenommen werden können. `B1Err` ist ein Flag, über das B_1-Inhomogenitäten während der Simulation vernachlässigt (`B1Err = 0`) oder berücksichtigt (`B1Err = 1`) werden können. In letzterem Fall wird eine Gauss-Verteilung berechnet, deren Parameter `nB1err` und `interval` im Quell-Code anzupassen sind (standardmäßig wird Fehlerverteilung mit 10% Halbwertsbreite mit einer Digitalisierung von fünf Datenpunkten berechnet). Mit `tau90` wird die Länge eines harten $\pi/2$-Pulses in μs übergeben, aus der intern die entsprechende RF-Frequenz berechnet wird. Die letzten beiden Parameter bestimmen den Umfang des simulierten Offset-Plots. Dieser ist mit $\nu_{I,S} = [-\texttt{maxOff}, \texttt{maxOff}]$ stets quadratisch und besteht aus ($\texttt{ResOff} \times \texttt{ResOff}$) Datenpunkten.

Die Rückgabewerte enthalten den berechneten Offset-Plot. In den Matrizen `offI` bzw. `offS` sind die x- bzw. y-Achsen (d.h. Offsetfrequenzen in Hertz) gespeichert. Die Matrix `transfer` enthält die entsprechenden Daten der z-Achse (Transferamplitude η).

Im Quellcode kann man außerdem die zu verwendenden Phasenzyklen für die Refokussierungs-Pulse angeben. In der hier gezeigten Version wird ein MLEV-8-Zyklus verwendet. `Phase1` bezieht sich auf die erste Rotationsperiode, während `Phase3` in der zweiten Rotationsperiode verwendet wird.

Umrechnung der schmalbandingen Hartpuls-Näherung in eine breitbandige mit gleichzeitiger Berechnugn eines Offset-Plots:

```
function [offI, offS, transfer] = simseqbb( p1, p3, d1, d2, d3, J, T2, B1err, tau90, maxOff, ResOff )
% define the sequence, pulses in [pi/2], delays in [usec], tau90 in [usec]

    [dummy, np1] = size(p1);
    [dummy, np3] = size(p3);

    [dummy, nd1] = size(d1);
    [dummy, nd3] = size(d3);
```

```
D = 1/(pi*T2);

% Define the phase cycles for the 180's
% MLEV-8
Phase1 = [1 1 -1 -1 1 -1 -1 1];
Phase3 = [1 1 -1 -1 1 -1 -1 1];

% definition of the pauli-matrices
ix = 0.5 * [0 1; 1 0];
iy = 0.5 * [0 -i; i 0];
iz = 0.5 * [1 0; 0 -1];
id = [1 0; 0 1];

% build up 2-spin system
Ix = kron(ix,id);
Sx = kron(id,ix);
Iy = kron(iy,id);
Sy = kron(id,iy);
Iz = kron(iz,id);
Sz = kron(id,iz);

sIz = sKet( Iz );
sIx = sKet( Ix );
sIySz = sKet( 2*Iy*Sz );
sIzSz = sKet( 2*Iz*Sz );

% build relaxation operator
A = 2*Iz*Sz;
Id = kron(id,id);
red = pi*D*( kron(A*A, Id) + kron(Id, A'*A') - kron(A, A') - kron(A,A') );

% define states (2-spin system's norm is 1, thus no normalization)
rho_start = sKet( Iz );
rho_target = sKet( 2*Iz*Sz );

% pulse field inhomogenity: ideal or real?
if (B1err == 1)

    % define the parameters for B1-inhomogeneity
    nB1err = 5;
    interval = 0.16;

    % calculate the distribution
    sigma = 0.04246;
    mu = 1.0;
    wSum = 0.0;
    for  idx=1:1:nB1err
            B1err(idx,1) = (mu-interval) + (idx-1)*( (2*interval)/(nB1err-1) );
            wB1err(idx,1) = 1 / ( sigma*sqrt(2*pi) ) * exp( - power( B1err(idx,1)-mu, 2) / (2*power(sigma,2)) );
            wSum = wSum + wB1err(idx,1);
    end;
    for idx = 1:1:nB1err
            wB1err(idx,1) = wB1err(idx,1) / wSum;
    end;

else
    nB1err = 1;
    B1err = 1.0;
    wB1err = 1.0;
end;

if (ResOff == 1)
    incDeltaOffI = 1;
    maxOff = 0;
else
    incDeltaOffI = (2*maxOff) / (ResOff - 1);
end;

% calculate the rf-amplitude of the 90deg pulse
nu_rf_I = 0.5*pi / (tau90*1e-6);
disp( sprintf('RF-amplitude: %f [kHz]', nu_rf_I/(2*pi)*1e-3));
nu_rf_S = nu_rf_I; % erst mal gleichwertige pulse fuer I und S

if (p3(1)==0)
    disp( sprintf('First pulse of phase III is zero, dropping two pi-pulses'));
else
    disp( sprintf('First pulse of phase III is unequal to zero, including two pi-pulses'));
    if (d2<2*tau90)
        disp( sprintf('WARNING: pi-pulse exceeds delay in phase II'));
    end;
end;

count_i = 1;
for deltaOffI =  -maxOff:incDeltaOffI:maxOff

    count_s = 1;
    for deltaOffS =  -maxOff:incDeltaOffI:maxOff

        H0 = sOp( 2*pi * (J * Iz*Sz + deltaOffI*Iz + deltaOffS*Sz) );
```

```
        TE = 0;
        for countI = 1:nB1err

            for countS = 1:nB1err
                % set up rf-Hamiltonians
                HrfIy = sOp( nu_rf_I * Iy * B1err(countI));
                HrfIx = sOp( nu_rf_I * Ix * B1err(countI));
                HrfIySy = sOp(nu_rf_I * Iy * B1err(countI) + nu_rf_S * Sy * B1err(countS));
                HrfIxSx = sOp(nu_rf_I * Ix * B1err(countI) + nu_rf_S * Sx * B1err(countS));

                rho_current = rho_start;

                for j=1:np1
                    rho_current = revolveKet( rho_current, HrfIy+H0, red, p1(j)*tau90*1e-6 );
                    if (j<=nd1)
                        rho_current = revolveKet( rho_current, H0, red, 0.5*(d1(j)-2*tau90)*1e-6 );
                        rho_current = revolveKet( rho_current, Phase1(j)*HrfIySy + H0, red, 2*tau90*1e-6 ); % 180y
                        rho_current = revolveKet( rho_current, H0, red, 0.5*(d1(j)-2*tau90)*1e-6 );
                    end;
                end;

                 if (p3(1)==0)
                    rho_current = revolveKet( rho_current, H0, red, 0.5*(d2+d3(1)-2*tau90)*1e-6 );
                    rho_current = revolveKet( rho_current, Phase3(1)*HrfIxSx + H0, red, 2*tau90*1e-6 ); % 180x
                    rho_current = revolveKet( rho_current, H0, red, 0.5*(d2+d3(1)-2*tau90)*1e-6 );
                    for j=2:np3
                        rho_current = revolveKet( rho_current, HrfIx+H0, red, p3(j)*tau90*1e-6 );
                       if (j<=nd3)
                            rho_current = revolveKet( rho_current, H0, red, 0.5*(d3(j)-2*tau90)*1e-6 );
                            rho_current = revolveKet( rho_current, Phase3(j)*HrfIxSx + H0, red, 2*tau90*1e-6 ); % 180x
                            rho_current = revolveKet( rho_current, H0, red, 0.5*(d3(j)-2*tau90)*1e-6 );
                        end;
                    end;
                else
                    rho_current = revolveKet( rho_current, H0, red, 0.5*(d2-2*tau90)*1e-6 );
                    rho_current = revolveKet( rho_current, HrfIxSx + H0, red, 2*tau90*1e-6 ); % 180x
                    rho_current = revolveKet( rho_current, H0, red, 0.5*(d2-2*tau90)*1e-6 );
                    rho_current = revolveKet( rho_current, -HrfIxSx + H0, red, 2*tau90*1e-6 ); % 180-x
                    for j=1:np3
                        rho_current = revolveKet( rho_current, HrfIx+H0, red, p3(j)*tau90*1e-6 );
                       if (j<=nd3)
                            rho_current = revolveKet( rho_current, H0, red, 0.5*(d3(j)-2*tau90)*1e-6 );
                            rho_current = revolveKet( rho_current, Phase3(j)*HrfIxSx + H0, red, 2*tau90*1e-6 ); % 180x
                            rho_current = revolveKet( rho_current, H0, red, 0.5*(d3(j)-2*tau90)*1e-6 );
                        end;
                    end;
              end;

                TE = TE + wB1err(countI)*wB1err(countS)*(rho_current' * rho_target);

            end; % countS (B1err)

        end; % countI (B1err)

        transfer(count_i,count_s) = real(TE);
        offI(count_i, count_s) = deltaOffI;
        offS(count_i, count_s) = deltaOffS;

        count_s = count_s + 1;

    end; % count_s (Offset)

    count_i = count_i + 1;

end; % count_i (Offset)


% Display sequence as is
for j=1:np1
    disp( sprintf( '\tpulse: (phase: y   %f [pi/2])', p1(j) ) );
    if (j<=nd1)
        disp( sprintf( '\tdelay: %f [usec]', 0.5*(d1(j)-2*tau90) ) );
        disp( sprintf( '\tpulse: (phase: (%i)y   2 [pi/2])', Phase1(j) ) );
        disp( sprintf( '\tdelay: %f [usec]', 0.5*(d1(j)-2*tau90) ) );
    end;
end;
if (p3(1)==0)
    disp( sprintf( '\tdelay: %f [usec]', 0.5*(d2+d3(1)-2*tau90) ) );
    disp( sprintf( '\tpulse: (phase: (%i)x   2 [pi/2])', Phase3(1) ) );
    disp( sprintf( '\tdelay: %f [usec]', 0.5*(d2+d3(1)-2*tau90) ) );
    for j=2:np3
        disp( sprintf( '\tpulse: (phase: x   %f [pi/2])', p3(j) ) );
        if (j<=nd3)
            disp( sprintf( '\tdelay: %f [usec]', 0.5*(d3(j)-2*tau90) ) );
            disp( sprintf( '\tpulse: (phase: (%i)x   2 [pi/2])', Phase1(j) ) );
            disp( sprintf( '\tdelay: %f [usec]', 0.5*(d3(j)-2*tau90) ) );
        end;
    end;
```

```
else
    disp( sprintf( '\tdelay: %f [usec]', 0.5*(d2-2*tau90) ) );
    disp( sprintf( '\tpulse: (phase: (1)x   2 [pi/2])' ) );
    disp( sprintf( '\tdelay: %f [usec]', 0.5*(d2-2*tau90) ) );
    disp( sprintf( '\tpulse: (phase: (-1)x   2 [pi/2])' ) );
    for j=1:np3
        disp( sprintf( '\tpulse: (phase: x   %f [pi/2])', p3(j) ) );
        if (j<=nd3)
            disp( sprintf( '\tdelay: %f [usec]', 0.5*(d3(j)-2*tau90) ) );
            disp( sprintf( '\tpulse: (phase: (%i)x   2 [pi/2])', Phase1(j) ) );
            disp( sprintf( '\tdelay: %f [usec]', 0.5*(d3(j)-2*tau90) ) );
        end;
    end;
end;
```

Kapitel 15

Zusammenfassung und Ausblick

Mit der vorliegenden Arbeit konnte gezeigt werden, daß sich die Prinzipien der Steurungs-Theorie zur Kontrolle der quantenmechanischen Dynamik von Spinsystemen in der kernmagnetischen Resonanzspektroskopie einsetzen lassen. Diese Systeme sind steuerbar, weil

- sie durch eine Bewegungsgleichung beschrieben werden können:
  Differentialgleichung der Bewegung nach Liouville-von Neumann
- diese in der NMR um äußere systemunabhängige Terme (Kontrollen) erweiterbar ist:
  Kontrolle der Bewegung durch RF-Felder

Da die erzeugbare Amplitude der RF-Felder in der Praxis begrenzt ist, bezeichnet man die Kontrollen als beschränkt. Daher erfolgt die Beschreibung des Stuerungs-Problems durch die nicht-klassischen Variationstheorie (Maximum Prinzip nach Pontryagin).

In Teil I wurden die theoretischen Grundlagen dieser Arbeit beschrieben. Hierbei wurde in Kapitel 2 die Quantenmechanik von N-Spinsystemen im Rahmen des Dichtematrix-Formalismus beschrieben. Diese Beschreibung wurde in Kapitel 3 um den Effekt der Kernspin-Relaxation erweitert. Abschließend wurde in Kapitel 4 eine kurze Einführung in die Steuerungs-Theorie vorgestellt.

In Teil II wurden numerische Optimierungsergebnisse vorgestellt, die unter Verwendung der im Rahmen dieser Arbeit entwickelten Optimierungs-Software OCTANE erhalten wurden.

Kapitel 5 zielte auf die Untersuchung idealer Spinsysteme ab. Es wurden Aufbaukurven für Anti- und Inphase-Transfer für Spinsystemen aus zwei bis fünf Kernspins für verschiedene einfache Kopplungstopologien durchgeführt (I_nS mit $n \in [1,3]$ und Spinketten I_n mit $n \in [2,5]$). Da man diese Kurven den für jede Zeit (im Rahmen der steuerungstheoretischen Beschreibung) optimalen Transfer entnehmen kann, werden diese als TOP-Kurven (Time Optimal Pulses) bezeichnet. Wegen der Annahme idealer Bedingungen kann man neben der quantenmechanisch gegebenen maximalen Transferamplitude (die sog. unitäre Grenze) auch die minimale Zeit identifizieren, in der diese noch durch eine Pulssequenz implementiert werden kann.

In Kapitel 6 wurden erste Untersuchungen zur optimalen Steuerung unter Einschluß von Relaxation vorgestellt. Zunächst wurde an einem Pseudo-1-Spinsystem die Gültigkeit des Ansatzes nachvollzogen. Im Anschluß daran wurde ein INEPT-Schritt für ein dipolar relaxierendes 2-Spinsystem optimiert. Aus den Ergebnissen wurde ein charakteristisches Transferschema gewonnen, welches gegenüber dem INEPT-Experiment verbesserte Eigenschaften aufweist. Dieses konnte inzwischen durch anlytische Betrachtungen bestätigt und erfolgreich in ein NMR-Experiment umgesetzt werden. Solche durch Anwendung der Steuerungstheorie relaxations-optimierte Pulssequenz-Elemente werden allgemein als ROPE (Relaxation Optimized Pulse Elements) bezeichnet. Da sich in diesem Kapitel ausschließlich auf diplorare Relaxation beschränkt wurde, die aufgrund der Größe der verwendeten Spinsysteme nur autokorreliert sein konnte, ist eine Erweiterung des Relaxationsmechanismus für zukünftige Arbeiten interessant, vor allem auf die Berücksichtigung von kreuzkorrelierter DD/CSA-Relaxation. Darüberhinaus wird es für die Optimierung von Pulssequenzen mit dem Einsatzziel großer Moleküle von Wichtigkeit sein, einen "Störspin" in das Spinsystem aufzunehmen. Durch eine entsprechende Wahl der Parameter dieses Kernspins ist es möglich, die Relaxationseigenschaften großerer Moleküle zu emulieren. Diese beiden Erweiterungen ermöglichen es, prinzipielle Einsichten in ROPE-Sequenzen für makromolekulare biologische Systeme zu gewinnen.

In Kapitel 7 wurden die weiteren in Octane implementierten Störeffekte (Radiofrequenz-Offsets, Resonanz-Offsets) am Beispiel der Optimierung eines breitbandigen und fehlertoleranten Anregungspulses getestet. Diese wurde für eine Anregungsbandbreite von 40 kHz und eine Fehlertoleranz vom

$\pm 5\%$ durchgeführt. Die numerisch ezeugte RF-Shape (BEBOP, Broadband Excitation By Optimized Pulses) konnte direkt auf einem Spektrometer implementiert werden, wobei sich die Güte der Anregung bestätigen ließ. Durch Rekalibierung der Radiofrequenz-Parameter konnte die Bandbreite gar auf 64 kHz erweitert werden. Da die gewählte Pulsdauer noch nicht im Bereich der Zeitoptimalität liegt, ist es zunächst interessant, einen zeitoptimalen Anregungspuls numerisch zu bestimmen . Neben diesem Parameter ist es darüber hinaus ebenso interessant, die Grenzen der erreichbaren Bandbreite und Fehlertoleranz auszuloten.

Mit den in Teil II beschriebenen Ergebnissen konnte gezeigt werden, daß die numerische Optimierung von NMR-Experimente mittels der kontrolltheoretischen Bibliothek OCTANE in den verschiedensten Bereich funktionsfähig ist und die numerisch gewonnenen Ergebnisse von theoretischen Betrachtungen bishin zur automatisierten Generierung realer Experimente reichen. Durch die allgemeine Formulierung und die Korrelationsmöglichkeiten der implementierten Effekte sind die Anwendungsbereiche mannigfaltig. In dieser Dissertation konnten nicht mehr als einige Beispiele dafür gezeigt werden, hierbei konnte eine Grenze der Optimierungsmöglichkeiten nicht identifiziert werden. Derzeit wird das Einsatzgebiet lediglich durch die zur Verfügung stehende Rechenzeit limitiert. Durch eine weitere Optimierung und Parallelisierung des Programm-Codes sind auch in diesem Bereich noch Möglichkeiten für zukünftige Weiterentwicklungen gegeben.

In Teil III wurden experimentelle Implementierungen von optimierten Pulssequenzen vorgestellt, deren analytische Beschreibung auf geometrischen Betrachtungen der Steuerungstheorie beruhen. Im Einzelnen handelt es sich dabei um zeitoptimalen Inphase-Transfer in IS-Spinsystemen (Kapitel 8), ein zeitoptimales SWAP(1,3)-Gatter (Kapitel 9) und die effiziente Erzeugung und Propagation effektiver Soliton-Operatoren in Ising-Ketten (Kapitel 10). Durch die mathematische Analyse konnte für alle Sequenzen die Erfüllung der Optimalitätsbedingung gezeigt werden. Bei den in dieser Arbeit gezeigten Implementierungen stand die Übersetzung der mathematisch formulierten Kontrollgesetzte in praxistaugliche NMR-Experimente im Vordergrund.

Die im Rahmen dieser Arbeit entwickelte Software wurde in Teil IV besprochen. Zunächst in Kapitel 11 die Umsetzung des theoretischen Rahmenwerks in die Computer-Optimierungsbibliothek OCTANE vorgestellt und sowohl die Design-Prinzipien als auch die Verwendung detailliert diskutiert. Diese Bibliothek ermöglicht die automatisierte Erzeugung optimaler RF-Formen unter dem Einschluß der in der NMR typischen Störeffekte (Relaxation, Radiofrequenz-Offsets, Resonanz-Offsets). Ein wichtiges Element der Bibliothek stellt das mathematische Konstrukt der Matrix dar. Hierfür wurde die Templat-Klasse `Matrix` entwickelt, deren Sprachumfang in Kapitel 12 erläutert wurde.

Literaturverzeichnis

[1] F. Bloch, W.W. Hansen, M. Packard, *Phys. Rev.* **69**, 127 (1946).

[2] E.M. Purcell, H.C. Torrey, R.V. Pound, *Phys. Rev.* **69**, 37 (1946).

[3] T.R. Brown, B.M. Kincaid, K. Ugurbil, *Proc. Natl. Acad. Sci.* **79**, 3523 (1982).

[4] D.C. Roe, P.M. Kating, P.J. Krusic, B.E. Smart, High Resolution NMR Techniques in Catalysis, *Topics in Catalysis* **5** 133-147 (1998).

[5] A.E. Ferentz, G. Wagner, NMR spectroscopy: a multifaceted approach to macromolecular structure, *Q. Rev. Biophys.* **33**, 29-36 (2000).

[6] N.A. Gershenfeld, I.L. Chuang, Bulk Spin-Resoance Quantum Computation, *Science* **275**, 350-356 (1997).

[7] O.W. Sørensen, G.W. Eich, M.H. Levitt, G. Bodenhausen, R.R. Ernst, *Prog. Nucl. Magn. Reson. Spectros.* **16**, 163-192 (1983).

[8] O.W. Sørensen, *Prog. Nucl. Magn. Reson. Spectros.* **21**, 503-669 (1989).

[9] K. Pervushin, R. Riek, G. Wider, K. Wüthrich, Attenuated T2 relaxation by mutual cancellation of dipole-dipole coupling and chemical shift anisotropy indicates an avenue to NMR structures of very large biological macromolecules in solution, *Proc. Natl. Acad. Sci. USA* **94**, 12366-12371 (1997).

[10] R. Riek, G. Wider, K. Pervushin, K. Wüthrich, Polarization transfer by cross-correlated relaxation in solution NMR, *Proc. Natl. Acad. Sci. USA* **96**, 4918-4923 (1999).

[11] G.F. Franklin, J.D. Powell, A. Emami-Naeini, "Feedback Control of Dynamic Systems" (2. Ausgabe), Adison Wesley, Reading (1991).

[12] K. Blum, "Density Matrix Theory and Applications", Seiten 1-217. Plenum Press, New York (1981).

[13] F. Bloch, *Phys. Rev.* **70**, 460-474 (1946).

[14] J. Cavanagh, W.J. Fairbrother, A.G. Palmer III, N.J. Skelton, "Protein NMR Spectroscopy", Seiten 1-587. Academic Press, San Diego (1996).

[15] W. Kutzelnigg, "Einführung in die Theoretische Chemie", Band 1, Seiten 1-287. Verlag Chemie, Weinheim (1992).

[16] I.N. Levine, "Quantum Chemistry", Seiten 1-566. Allyn & Bacon, Boston (1983).

[17] N. Bazley, D.W. Fox, *J. Math. Physics* **4**, 1147 (1963).

[18] N. Bazley, D.W. Fox, *Rev. mod. Phys.* **35**, 712 (1963).

[19] J. Reinhold, "Quantentheorie der Moleküle", Seiten 1-372. B.G. Teubner, Stuttgart (1994).

[20] P.A.M. Dirac, "The Principles of Quantum Mechanics", Seiten 1-314. Oxford University Press, New York (1967).

[21] A. Abgragam, "Principles of Nuclear Magnetism", Seiten 1-599. Clarendon Press, Oxford (1961).

[22] R.R. Ernst, G. Bodenhausen, A. Wokaun, "Principles of Nuclear Magnetic Resonance in One and Two Dimensions", Seiten 1-596. Oxford Science Publications, Oxford (1988).

[23] F. Lorenz, "Lineare Algebra", Band 2, Seiten 1-189. BI-Wissenschaftsverlag, Mannheim (1992).

[24] L.E. Kay, R.E.D. McClung, *J. Magn. Reson.* **77**, 258-273 (1988).

[25] P.L. Corio, "Structure of High Resolution NMR Spectra", Seiten 1-548. Academic Press, New York (1967).

[26] R.K. Wangsness, F. Bloch, *Phys. Rev.* **89**, 728-739 (1953).

[27] A.G. Redfield, *Adv. Magn. Reson.* **9**, 1-32 (1965).

[28] N.C. Pyper, *Mol. Phys.* **22**, 433-458 (1971).

[29] P.S. Hubbard, *Phys. Rev.* **180**, 319-326 (1969).

[30] D.M. Brink, G.R. Satchler, "Angular Momentum", Seiten 1-170. Clarendon Press, Oxford (1993).

[31] J. McConnell, "The theory of nuclear magnetic relaxation in liquids", Seiten 1-191. Cambridge University Press, Cambridge (1987).

[32] D. Neuhaus, M. Williamson, "The Nuclear Overhauser Effect in Structural and Conformational Analysis", Seiten 1-514. VCH Publishers, New York (1989).

[33] I.M. Gelfand, S.V. Fomin "Calculus of Variations", Prentice Hall, London (1963).

[34] E.R. Pinch "Optimal Control and the Calculus of Variations", Oxford University Press, New York (1997).

[35] I.M. Bomze, W. Grossmann "Optimierung - Theorie und Algorithmen", BI Wissenschaftsverlag, Mannheim (1993).

[36] J. Varga "Angewandte Optimierungs", BI Wissenschaftsverlag, Mannheim (1991).

[37] L.S. Pontryagin, Optimal control processes, *Usp. Nat. Nauk.* **14**, 3 (1959).

[38] V.G. Boltyanskii, R.V. Gamkrelidze, L.S. Pontryagin, The Theory of Optimal processes - The Maximum Principle, *Izv. Akad. Nauk SSSR, Ser. Mat.* **24**, 3 (1960).

[39] L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze, E.F. Mishchenko "The mathematical theory of optimal processes", Pergamon Press, Oxford (1964).

[40] A.J. Shaka, C.L. Lee, A.J. Pines, Iterative Schemes for Bilinear Operators: Applications to Spin Decoupling, *J. Magn. Reson.* **77**, 274-293 (1988).

[41] S. Szymanksi, A.M. Gryff-Keller, G. Binsch, A Liouville Space Formulation of Wagness-Bloch-Redfield Theory of Nuclear Spin Relaxation Suitable for Machine Computation, *J. Magn. Reson.* **68**, 399-432 (1986).

[42] Navin Khaneja, Timo O. Reiss, Burhard Luy, Steffen J. Glaser, Optimal Control of Spin Dynamics in the Presence of Relaxation, arXiv:quant-ph/0208050

[43] B. Blümich, H.W. Spiess, Quaternions as a Practical Tool for the Evaluation of Composite Rotations, *J. Magn. Reson.* **61**, 356-362 (1985).

[44] A.J. Shaka, A.J. Pines, Symmetric phase-alternating composite pulses, *J. Magn. Reson.* **71**, 495-503 (1987).

[45] R. R. Ernst, G. Bodenhausen, A. Wokaun, "Principles of Nuclear Magnetic Resonance in One and Two Dimensions", Clarendon Press, Oxford (1987).

[46] O. W. Sørensen, Polarization Transfer Experiments in High-Resolution NMR Spectroscopy, *Prog. NMR Spectrosc.* **21**, 503-569 (1989).

[47] O. W. Sørensen, A Universal Bound on Spin Dynamics, *J. Magn. Reson.* **86**, 435-440 (1990).

[48] S. J. Glaser, T. Schulte-Herbrüggen, M. Sieveking, O. Schedletzky, N. C. Nielsen, O. W. Sørensen, C. Griesinger, Unitary Control in Quantum Ensembles, Maximizing Signal Intensity in Coherent Spectroscopy, *Science* **208**, 421-424 (1998).

[49] T. Untidt, T. Schulte-Herbrüggen, B. Luy, S. J. Glaser, C. Griesinger, O.W. Sørensen, N. C. Nielsen, Design of NMR Pulse Experiments with Optimum Sensitivity: Coherence-Order-Selective Transfer in I_2S and I_3S Spin Systems, *Mol. Phys.* **95**, 787-796 (1998).

[50] N. Khaneja, R. W. Brockett, S. J. Glaser, Time Optimal Control in Spin Systems, *Phys. Rev. A* **63**, 03208 (2001).

[51] R. W. Brockett, "System Theory on Group Manifolds and Coset Spaces," *SIAM J. Control* **10**, 2 (1972).

[52] D. P. Weitekamp, J. R. Garbow, and A. Pines, Determination of Dipole Coupling Constants Using Heteronuclear Multiple Quantum NMR, *J. Chem. Phys.* **77**, 2870-2883 (1982); Erratum, *J. Chem. Phys.* **80**, 1372 (1984).

[53] P. Caravatti, L. Braunschweiler, and R. R. Ernst, Heteronuclear Correlation Spectroscopy in Rotating Solids, *Chem. Phys. Lett.* **100**, 305-310 (1983).

[54] S. J. Glaser and J. J. Quant, Homonuclear and heteronuclear Hartmann-Hahn transfer in isotropic liquids, *in* "Advances in Magnetic and Optical Resonance" (W. S. Warren, Ed.), Vol. 19, pp. 59 - 252, Academic Press, San Diego (1996).

[55] A. A. Maudsley, A. Wokaun, R. R. Ernst, Coherence Transfer Echoes, *Chem. Phys. Lett.* **55**, 9-14 (1978).

[56] R. E. Hurd. B. K. John, Gradient-Enhanced Proton-Detected Heteronuclear Multiple-Quantum Coherence Spectroscopy, *J. Magn. Reson.* **91**, 648-653 (1991).

[57] M. Sattler, P. Schmidt, J. Schleucher, O. Schedletzky, S. J. Glaser, C. Griesinger, Novel Pulse Sequences with Sensitivity Enhancement for In-phase Coherence Transfer Employing Pulsed Field Gradients, *J. Magn. Reson. B* **108**, 235-242 (1995).

[58] S. J. Glaser, T. Schulte-Herbrüggen, M. Sieveking, O. Schedletzky, N. C. Nielsen, O. W. Sørensen, C. Griesinger, Unitary Control in Quantum Ensembles, Maximizing Signal Intensity in Coherent Spectroscopy, *Science* **208**, 421-424 (1998).

[59] T. Untidt, T. Schulte-Herbrüggen, B. Luy, S. J. Glaser, C. Griesinger, O. W. Sørensen, N. C. Nielsen, Design of NMR Pulse Experiments

with Optimum Sensitivity: Coherence-Order-Selective Transfer in I_2S and I_3S Spin Systems, *Mol. Phys.* **95**, 787-796 (1998).

[60] N. Khaneja, R. W. Brockett, S. J. Glaser, Time Optimal Control in Spin Systems, *Phys. Rev. A* **63**, 03208 (2001).

[61] T. O. Reiss, N. Khaneja, S. J. Glaser, Time-Optimal Coherence-Order-Selective Transfer of In-Phase Coherence in Heteronuclear IS Spin Systems, *J. Magn. Reson.* **154**, 192-195 (2002).

[62] N. Khaneja, R. W. Brockett, S. J. Glaser, Sub-Riemannian Geometry and Time Optimal Control of Three Spin Systems: Quantum Gates and Coherence Transfer, *Phys. Rev. A*, in press (2001).

[63] R. R. Ernst, G. Bodenhausen, A. Wokaun, Principles of Nuclear Magnetic Resonance in One and Two Dimensions, Clarendon Press, Oxford (1987).

[64] N. Gershenfeld, I, L. Chuang, Bulk spin-resonance quantum computation, *Science* **275**, 350-356 (1997).

[65] D. G. Cory, A. F. Fahmy, T. F. Havel, Ensemble quantum computing by NMR spectroscopy, *Proc. Natl. Acad. Sci* **94**, 1634-1639 (1997).

[66] C. H. Bennet, D. P. DiVinvenzo, Quantum information and computation, *Nature* **404**, 247-255 (2000).

[67] A. Barenco, C.H. Bennett, R. Cleve, D.P. Divincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin and H. Weinfurter, Elementary gates for quantum computation, *Phys. Rev. A* **28** (1996).

[68] T. Schulte-Herbrüggen, O. W. Sørensen, The Relationship between Ensemble Quantum Computing Logical Gates and NMR Pulse Sequences Engineering Exemplified by the SWAP operation, *Conc. Magn. Reson.* **12(6)**, 389-395 (2000).

[69] N. Linden, H. Barjat, E. Kupče, R. Freeman, How to exchange information between twp coupled nuclear spins: the universal SWAP operation, *Chem. Phys. Lett.* **307**, 198-204 (1999).

[70] Z. L. Mádi, R. Brüschweiler, R. R. Ernst, One-and two-dimensional ensemble quantum computing in spin Liouville space, *J. Chem. Phys.* **109**, 10603-10611 (1998).

[71] S. J. Glaser and J. J. Quant, Homonuclear and heteronuclear Hartmann-Hahn transfer in isotropic liquids, *in* "Advances in Magnetic and Optical Resonance" (W. S. Warren, Ed.), Vol. 19, pp. 59 - 252, Academic Press, San Diego (1996).

[72] N. Khaneja, S. J. Glaser, Spin Solitons and Quantum Control of Spin Chain Dynamics, preprint quant-ph/0202013 (2002).

[73] E. Ising, *Z. Physik* **31**, 253 (1925).

[74] Z.L. Madi, B. Brutscher, T. Schulte-Herbrüggen, R. Brüschweiler, R.R.Ernst, Time-resolved observation of spin waves in a linear chain of nuclear spins, *Chem. Phys. Lett* **268**, 300-305 (1997).

[75] T. Schulte-Herbrüggen, Z.L. Madi, O.W. Sørensen, R.R. Ernst, *Mol. Phys.* **72**, 847 (1991),

[76] The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, U.S.A.

[77] B. Stroustrup, "The C++ Programming Language", Seiten 1-632. Addison-Wesley, Reading (1991).

[78] G. Willms, "C++", Seiten 1-1164, Data Becker, Düsseldorf (1997).

[79] G. Satir, D. Brown, "C++: The Core Language", O'Reilly, Sebastopol (1995).

[80] S. Oualline, "Practical C++ Programming", Seiten 1-581, O'Reilly, Sebastopol (1995).

[81] B.W. Kernighan, D.M. Richie "Programmieren in C", Seiten 1-262. Carl Hanser Verlag, München (1990).

[82] W.R. Stevens, "Advanced Programming in the UNIX Environment", Seiten 1-712. Addison-Wesley, Reading (1997).

[83] G. Booch, J. Rombaugh, I- Jacobson, "The Unified Modelling Language - User Guide", Seiten 45-202, Addison-Wesley, Reading (1999).

[84] W.H. Press, S.A. Teukolsky, "Numerical Recipes in C" (2. Ausgabe), Cambridge University Press, Cambridge (1992).

[85] E. Polak "Computational Methods in Optimization", Academic Press, New York (1971).

[86] T. Grams "Simulation - Strukturiert und Objektorientiert programmiert", BI Wissenschaftsverlag, Mannheim (1992).

[87] Z. Madi "WTest - general purpose simulation of magnetic resonance pulse experiments in the time domain using numerical density matrix calculations", Eidgenössische Technische Hochschule Zürich (1989).

[88] G.H. Golub, C.F. Van Loan, "Matrix Computation", Seite 384, Algorithmus 11.3-1, Johns Hopkins University Press (1983).

[89] C.B. Moler, C.F. Van Loan, Nineteen Dubious Ways to Compute the Exponential of a Matrix, *SIAM Review* **20**, 801-836 (1979).

[90] Matlab Dokumentation, Band: "MAT-File Format", 1-39, The Mathworks Inc. (1999).

Danksagung

Ich danke Herrn Prof. Dr. Steffen J. Glaser für die Ermöglichung dieser Arbeit in seiner Arbeitsgruppe. Seine Unterstützung während dieser Zeit, seine Bereitschaft zur Diskussion und seine Anleitung zu wissenschaftlichen Arbeiten waren mir stets eine große Freude. Zudem möchte ich mich für die mir gewährten Freiheiten bei der Durchführung dieser Arbeit bedanken.

Ich danke Herrn Prof. Dr. Frank H. Köhler für die freundliche Übernahme des Korreferats.

Darüberhinaus gilt mein Dank: Gerd Hauser und Frank Kramer für die stets angenehme Atmosphäre in 42.102; dem Rest des AK Glaser: Dr. Thomas Schulte-Herbrüggen, Cindie Kehlet und Dr. Raimund Marx; Monika Goede und Dr. Rainer Haeßner für die Rechner- und Spektrometerwartung; Prof. Dr. Navin Khaneja (Harvard Universität) für seine Ideen und die Kooperation bezüglich allem, was mit dem Gebiet der Optimalen-Kontroll-Theoie zu tun hat; Prof. Dr. Thomas E. Skinner (Ohio State Universität) für die Zusammenarbeit während seines Sabbaticals, die in den Ergebnissen aus Kapitel 7 fruchtete; Dr. Burkhard Luy für seinen experimentellen Beitrag zu Kapitel 7; dem AK Kessler für alle gemeinschaftlichen Aktivitäten. Ein großes Dankeschön an die Lektoren: Dr. Jens Liermann (APAC GmbH) für seinen unermüdlichen Einsatz bei der Korrektur des informatischen Teils dieser Arbeit und der Hilfe bei der Erstellung des UML-Diagrammes; Gerd Hauser für seine Bereitschaft die experimentellen Kapitel Korrektur zu lesen. Abschließend möchte ich mich bei meiner Familie für ihre Unterstützung bedanken.