Beiträge zur geometrischen Integration und Anwendungen in der numerischen Simulation

Habilitationsschrift

zur Erlangung des akademischen Grades

Dr. rer. nat. habil.

vorgelegt der
Mathematisch-Naturwissenschaftlich-Technischen Fakultät

der Martin-Luther-Universität Halle-Wittenberg

von

Dr. rer. nat. Jörg Wensch
geb. am 29.9.68 in Köthen

Gutachter
1. Prof. Dr. Karl Strehmel
2. Prof. Dr. Peter Rentrop
3. Prof. Dr. Martin Arnold

Halle (Saale), den 7.10.2003.

Datum der Verteidigung: 24.6.2004.

Für Monika, Friederike und Leonhard.

Herrn Prof. Karl Strehmel und Herrn Prof. Rüdiger Weiner danke ich für die Förderung und engagierte Begleitung dieser Arbeit, für die langjährige Unterstützung meiner Tätigkeit und das wohlwollende Klima im Institut für Numerische Mathematik, für das beide verantwortlich zeichnen.

Teile dieser Arbeit entstanden aus der Zusammenarbeit mit anderen Forschern. Mein besonderer Dank gilt meinem langjährigen Kollegen Helmut Podhaisky für viele interessante und anregende Diskussionen, für das Klima kreativen Ideenaustauschs in unserem Arbeitsraum, und letztendlich auch für den Anstoß zur Thematik in Kapitel 5. In dieses flossen ebenso Ergebnisse aus einer Zusammenarbeit mit Herrn Stefan Hartmann von der Universität Kassel ein.

Meinem Kollegen Alf Gerisch möchte ich ebenso für einen angeregten Ideenaustausch sowie für die stete Bereitschaft, englischsprachigen Publikationen den letzten Schliff zu geben, danken.

Herrn Markus Däne und den Kollegen von der Arbeitsgruppe “Computational Physics” danke ich für die anregende Zusammenarbeit, deren Ergebnisse in Kapitel 4 dieser Arbeit Niederschlag finden. Ben Sommeijer war mein geschätzter Gastgeber bei einem Forschungsaufenthalt am CWI in Amsterdam im Herbst 2001, wo die Inhalte von Kapitel 6 entstanden sind.

Herrn Prof. Arnold von der Martin-Luther-Universität Halle-Wittenberg und Herrn Prof. Peter Rentrop von der Technischen Universität München möchte ich für das dieser Arbeit entgegengebrachte Interesse danken.

Großer Dank geht an meine ganze Familie, die die privaten Entbehrungen an Wochenenden und Abenden geduldig ertragen hat, besonders an meine Frau Monika, die mich auf meinem Weg stets bestärkt hat.

Halle, im Oktober 2003 Jörg Wensch

Inhaltsverzeichnis

Einleitung

Die Numerische Mathematik lebt von ihren Anwendungen. Komplexe Probleme aus Naturwissenschaft und Technik, Wirtschaft und Medizin führen nach geeigneter Modellierung zumeist auf mathematische Probleme in Funktionenräumen, die nicht exakt lösbar sind. Die Numerische Mathematik entwickelt Algorithmen zur Lösung dieser Probleme, sie überführt sie durch Diskretisierung in endlichdimensionale Probleme, sie untersucht (im Rahmen der numerischen Analysis mit fließendem Übergang zur Analysis selbst) die Lösbarkeit des Ausgangsproblems und die Konvergenzeigenschaften der Diskretisierung. Selbst für endlichdimensionale Probleme kann der optimale Lösungsalgorithmus nicht nur von direkter (endlicher), sondern von iterativer Natur sein. Für eine ganze Reihe von Verfahren sind durch theoretische Aussagen Existenz einer Lösung und die Konvergenz der Diskretisierung gesichert, andere Verfahren haben sich durch vielfache erfolgreiche Anwendung in der Praxis bewährt, harren aber noch der theoretischen Untermauerung.

Auch die zur Implementierung verwendete Rechnerarchitektur hat Einfluß auf die Wahl des optimalen Algorithmus. Erfordert die Größe des Problems eine parallele Architektur, so muß gute Parallelisierbarkeit bereits bei der Wahl des Algorithmus berücksichtigt werden, unter Umständen sogar schon bei der Wahl des mathematischen Modells. Dies erfordert genügende Sachkenntnis in der konkreten Anwendung, die Trennung von Anwendungsproblem und der daraus abgeleiteten mathematischen Aufgabenstellung führt oft zur Vernachlässigung entscheidender Details. Bei der Implementierung selbst ist zwischen Effizienz auf der einen Seite und leichter Implementierbarkeit/ Erweiterbarkeit auf der anderen Seite das Gleichgewicht zu wahren, die beabsichtigte Lebensdauer eines Codes spielt dabei eine große Rolle.

Diese Vielschichtigkeit von Problemstellungen des wissenschaftlichen Rechnens, das vom Numeriker abverlangte Wissen in vielen benachbarten Disziplinen wie Analysis/ numerischer Analysis, Informatik, Naturwissenschaften wie Physik und Chemie, Pharmazie und Medizin, oder ingenieurwissenschaftliche Disziplinen, schlägt sich auch in der Struktur dieser Arbeit nieder. Sowohl Teil I, als auch die im Teil II zusammengefaßten Kapitel 4,5 und 6 beschäftigen sich jeweils mit der Entwicklung und Analyse numerischer Verfahren unter Berücksichtigung konkreter Anwendungsprobleme.

Teil I ist dem großen Themenkomplex der geometrischen Integration gewidmet, er beschäftigt sich mit einer Verallgemeinerung von gewöhnlichen Differentialgleichungen im $\mathbb{R}^n$ auf Differentialgleichungen auf differenzierbaren Mannigfaltigkeiten. Einen Zugang zu solchen Problemen bietet der Lie-Gruppen-Ansatz. Bewegungen auf der Mannigfaltigkeit werden durch Transformationsgruppen beschrieben, und diese Transformationsgruppen sind eben Lie-Gruppen. Der Fall kommutativer Lie-Gruppen erweist sich als äquivalent zum $\mathbb{R}^n$, für nichtkommutative Gruppen sind hingegen Modifikationen notwendig, um die klassische Ordnung der Verfahren zu erhalten.

Kapitel 1 gibt einen Überblick über die verwendeten Hilfsmittel aus den Gebieten Analysis auf differenzierbaren Mannigfaltigkeiten sowie Lie-Gruppen und Lie-Algebren. Eine Einordnung des Materials in den Kapiteln 2 und 3 wird dadurch erleichtert. Wesentlich ist dabei zum einen die Parametrisierung nichtlinearer Mannigfaltigkeiten durch Transformationsgruppen (Lie-Gruppen), die wiederum

durch die Lie-Algebren dargestellt werden können. Die Vorgehensweise wird im ersten Abschnitt am Beispiel der Toda-Lattice-Gleichung ausführlich erläutert. Zum anderen führt eben die Ersetzung der kommutativen Vektorraumoperation Addition durch die (im allgemeinen) nichtkommutative Gruppenoperation zu speziellen Problemen bei der Konstruktion von Zeitintegrationsverfahren. Besondere Bedeutung kommt dabei den Abbildungen $\exp$, dexp sowie der Baker-Campbell-Hausdorff-Formel (BCH-Formel) zu. Für spezielle Lie-Gruppen können die Abbildungen mit effizienten Algorithmen berechnet werden, in einem gesonderten Abschnitt wird darauf eingegangen.

In Kapitel 2 werden die Grundideen bei der Übertragung von Runge-Kutta-Verfahren auf Lie-Gruppen, wie sie von Munthe-Kaas in den Arbeiten [72, 73, 74] entwickelt wurden, dargestellt. Neben intuitiven Zugängen in den ersten beiden Arbeiten findet sich in [74] auch die Nutzung der Lie-Algebra zur Parametrisierung der Lie-Gruppe. Klassische Grundbegriffe zur Charakterisierung von Einschrittverfahren werden entwickelt, man siehe auch [116].

Die Herleitung von Ordnungsbedingungen nach [73] wird dargestellt, verglichen wird dies im dritten Abschnitt mit einer Butcher-Baum-Theorie für die Matrixdarstellung der Lie-Grup-pen-Diffe-ren-ti-al-glei-chung. Außerdem gehen wir auf andere Parametrisierungen der Lie-Gruppe (Cayley-Abbildung) kurz ein.

Kapitel 3 beschreibt im Detail die Entwicklung von Extrapolationsverfahren. Gezeigt wird die Gültigkeit einer asymptotischen Entwicklung des globalen Fehlers auch in Lie-Gruppen. Auf dieser Basis können zunächst einfache Extrapolationsmethoden angegeben werden, die wie im ODE-Fall auch bei vorgegebener Ordnung deutlich aufwendiger als vergleichbare Runge-Kutta-Verfahren sind. Zu einem mächtigen Werkzeug wird die Extrapolationsidee erst, wenn eine quadratische Entwicklung des globalen Fehlers vorliegt.

Wir zeigen hier (man siehe auch [116]), daß diese zum einen für symmetrische Verfahren vorliegt, zum anderen beweisen wir, daß die explizite Mittelpunktregel bei entsprechender Interpretation symmetrisch ist. Im Gegensatz zum ODE-Fall erfordert die Herleitung geeigneter Verfahren hier zusätzlich eine Approximation der BCH-Formel, da die hergeleitete asymptotische Entwicklung in der Lie-Algebra gültig ist. Für Verfahren bis zur Ordnung 8 werden geeignete Approximationsformeln angegeben.

Zur Herleitung der Formeln wurde im Rahmen dieser Arbeit das Paket GLIEALG für Mathematica entwickelt und unter [119] der Allgemeinheit zur Verfügung gestellt. Dieses Paket erlaubt symbolische Rechnungen in bewerteten freien Lie-Algebren, so daß Potenzreihenentwicklungen der BCH-Formel einfach berechnet werden können. Insbesondere für höhere Approximationsordnungen wie 6 und 8 erweist sich die Anzahl der zu berücksichtigenden Ausdrücke als nicht mehr handhabbar. Der dritte Abschnitt widmet sich der Beschreibung dieses Paketes.

Abschließend wird die Zuverlässigkeit und Effizienz der hergeleiteten Verfahren an einem Beispiel aus der Physik – dem Toda-Lattice – unter Beweis gestellt. Die auf quadratischer Extrapolation beruhenden Methoden erweisen sich dabei als besonders geeignet.

Teil II stellt anhand von drei Beispielen die vielseitigen Einsatzmöglichkeiten effizienter und hochspezialisierter Zeitintegrationsverfahren zur Lösung von Anwenderproblemen aus den Naturwissenschaften dar. Die drei Teile eint, daß die eingesetzten Verfahren auf die jeweilige Problemklasse zugeschnitten sind – dies ist auch die Grundidee der geometrischen Integration. Bei den behandelten Problemen handelt es sich um lineare Differentialgleichungen mit stark oszillierenden Lösungen (Kapitel 4), um steife Probleme mittlerer Dimension (Kapitel 5) und um steife Systeme hoher Dimension (Kapitel 6). Zur numerischen Lösung kommen zum einen klassische Verfahren der Zeitintegration zum Einsatz, zum anderen wird demonstriert, wie der in Teil I dargestellte Lie-Gruppen-Ansatz zur Konstruktion neuer Zeitintegrationsverfahren – hier am Beispiel der von Iserles und Nørsett entwickelten

Magnus-Methoden – verwendet werden kann. Zum Vergleich muß gesagt werden, daß die numerische Zeitintegration mit Verfahren wie Runge-Kutta-Methoden auf eine mehr als hundertjährige Geschichte zurückblicken kann, so daß man mittlerweile über eine Vielzahl hochspezialisierter Methoden für verschiedene Anwendungsklassen verfügt. Die Idee der systematischen geometrischen Integration ist hingegen noch sehr jung, sie wurde in den 90er Jahren geboren, neue Verfahren wurden entwickelt und ältere Ideen wiederentdeckt und wiederbelebt. So basieren die Magnus-Methoden auf einer Entwicklung der Lösung linearer Differentialgleichungen nach W. Magnus aus dem Jahre 1954 [66]. Ideen zur symplektischen Integration finden sich in Arbeiten aus den späten 80er Jahren [36], [124], vor allem auf dem Gebiet der Astronomie wurden symplektische Verfahren auf der Basis generierender Funktionen entwickelt. Die Lie-Gruppen-Methoden, basierend auf Ideen von Munthe-Kaas, wurden in den 90er Jahren entwickelt. Auf dem Gebiet der partiellen Differentialgleichungen findet man eine Vielzahl problemangepaßter Diskretisierungen, die der geometrischen Integration zugeordnet werden können.

Insofern ist es auch nicht verwunderlich, daß klassische Verfahren wesentlich mehr Anwendung erfahren als geometrische Integratoren. Die vorliegende Arbeit möchte neben eigenen Beiträgen auch der Verbreitung der Idee der geometrischen Integration dienen. Daher haben wir im zweiten Teil zum einen Anwendungen ausgewählt, die mit klassischen Verfahren gelöst werden können, zum anderen wird in Kapitel 4 ein geometrisches Integrationsverfahren angewandt. Wir schlagen so eine Brücke zwischen geometrischer und klassischer Integration.

In Kapitel 4 werden Methoden aus der in Teil I beschriebenen Klasse der geometrischen Integratoren genutzt, um Problemstellungen der Quantenphysik zu lösen. Bei der Lösung der Schrödingergleichung zur Bestimmung der Wellenfunktion für Atome wird zumeist mit einer Separation des radialen Anteils und des sogenannten Drehimpulsanteils gearbeitet. Für den radialen Anteil führt dies zu gewöhnlichen Differentialgleichungen mit oszillierenden Lösungen.

Wir wenden auf diese Probleme Methoden, die wie die in Teil I beschriebenen zu den geometrischen Integratoren gezählt werden, an. Der Lie-Gruppen-Ansatz führt für lineare Probleme auf die von Iserles und Nørsett entwickelten Magnus-Methoden zurück. Grundlage ist die von W. Magnus 1954 hergeleitete Magnus-Entwicklung zur Lösungsdarstellung. Numerische Tests bestätigen die hohe Effizienz der geometrischen Integrationsmethoden für solche Probleme. Des weiteren testen wir verschiedene Varianten der Schrittweitensteuerung. Auch hier zeigt sich, daß solche Fehlerschätzer, die die Struktur des Problems berücksichtigen, hier eine zeitsymmetrische, am besten geeignet sind.

Kapitel 5 wendet sich einer Problemstellung aus der Mechanik zu, die auf ein differential-al-gebra-i-sches System führt. Bei der Zeitintegration solcher Systeme ist die Steifheit des Systems zu berücksichtigen. Eine effiziente Verfahrensklasse stellen linear-implizite Methoden dar, für einen Überblick verweisen wir auf die Monographie von Strehmel und Weiner [98]. Wir beschäftigen uns hier mit Rosenbrock-Typ Methoden. Die Anwendung von Krylovtechniken bei Rosenbrock-Methoden für gewöhnliche Differentialgleichungen findet man in den Arbeiten von Schmitt und Weiner [91], während die Anwendung von Rosenbrock-Methoden auf Index-1-DAEs selbst von verschiedenen Autoren untersucht wurde, wir verweisen exemplarisch auf Rentrop, Roche und Steinebach [82], [84]. Untersuchungen von Krylovtechniken bei BDF-Methoden für DAEs vom Index 1 findet man in der Arbeit von Brown, Hindmarsh und Petzold [7]. Wir wenden hier Rosenbrock-Methoden gekoppelt mit Krylov-Techniken auf die Index-1-Systeme an. Es zeigt sich dabei, daß die Berücksichtigung der algebraischen Gleichungen wesentlich für die Effizienz der Methode ist.

Getestet werden die im ersten Teil des Kapitels entwickelten Techniken bei der Lösung von Anwendungen aus der Kontinuumsmechanik. Geht man über die reine linear-elastische Deformation hinaus, verbleibt aber im Rahmen der hysteresefreien Deformationen, so gelangt man zum Phänomen der Viskoelastizität. Hierzu findet man eine Vielzahl von Modellen in der Literatur, wir haben uns für

ein dreielementiges Standard-Modell, das Hookesche Federn mit Newtonschen Dämpfern koppelt, entschieden. Die Bestimmung der zeitabhängigen Deformation bei Kriechprozessen führt nach einer Semidiskretisierung mit finiten Elementen zu einem differential-algebraischen System vom Index 1 von hoher Dimension. Bei impliziter Zeitintegration mit Rosenbrock-Methoden sind große lineare Gleichungssysteme zu lösen. Hierzu setzen wir Krylov-Techniken ein.

In Kapitel 6 wird eine Problemstellung aus der Neurobiologie betrachtet. Wir legen den Schwerpunkt neben der mathematischen Modellierung auch auf Implementierungsfragen, insbesondere interessieren wir uns für Zeitintegrationsverfahren, die eine einfache parallele Nutzung vieler Prozessoren gestatten.

Die Modellierung des Wachstums von Axonen im menschlichen Gehirn führt im betrachteten Fall auf ein gemischtes System aus parabolischen Gleichungen und Gradientengleichungen. Nach einer Semidiskretisierung mit der Linienmethode gelangt man zu einem steifen Anfangswertproblem. Dieses wird mit einem speziell adaptierten Zeitintegrationsverfahren gelöst, wir nutzen hier wegen der einfachen Parallelisierbarkeit ein explizites Verfahren, das aber dennoch über ausreichende Stabilitätseigenschaften verfügt.

Der resultierende sequentielle Algorithmus wird nun komponentenweise parallelisiert. Zum einen wird die Auswertung der rechten Seite der Differentialgleichung so parallelisiert (parallelism across the function), daß der Kommunikationsaufwand für die konkret genutzte CC-NUMA-Architektur minimiert wird. Das so erhaltene Schema muß nun auch auf die Methode übertragen werden. Hier zeigen sich die Vorteile der verwendeten expliziten Methode – da sie nur auf Level-1-BLAS-Routinen zugreift, ist die Parallelisierung der linearen Algebra mittels OpenMP sehr einfach zu realisieren.

Der parallele Algorithmus erweist sich insgesamt auch für eine große Anzahl von Prozessoren (128) als sehr effizient zur Lösung des Problems. Die Simulationsergebnisse stimmen mit theoretischen Vorhersagen überein. Durch Verwendung unterschiedlicher Parallelisierungsmodelle für die rechte Seite der Differentialgleichung konnten interessante und für weitere Simulationen hilfreiche Ergebnisse zur Parallelisierung auf der verwendeten Maschine (Teras, SGI Origin 3800) gewonnen werden.

Abschließend hofft der Verfasser, mit der vorliegenden Arbeit ein realistisches Bild der modernen numerischen Mathematik gegeben zu haben. Der engen Verbindung von Theorie und Anwendung wird durch die Tatsache Rechnung getragen, daß neben neuen theoretischen Resultaten in Kapitel 3, die in ein abstraktes Konzept (Kapitel 1 und 2) eingebettet sind, auch eine Darstellung typischer Praxisprobleme in den Kapiteln 4 bis 6 ihren Platz findet. Das im ersten Teil betrachtete Konzept der geometrischen Integration findet dabei durch die im Kapitel 4 angewandten Magnus-Methoden seine Würdigung, zum anderen werden in den Kapiteln 5 und 6 Beiträge zu den klassischen Zeitintegrationsverfahren geliefert, die hier - ganz im Geist der geometrischen Integration - problemangepaßte Verfahren für eine spezielle Anwendungsklasse darstellen.

Teil I

Differentialgleichungen in Lie-Gruppen – Ansätze, numerische Verfahren und Anwendungen

Kapitel 1

Theoretische Grundlagen

1.1 Ein motivierendes Beispiel – die Toda-Lattice-Gleichung

Eine Abstraktion ist stets durch ein Beispiel motiviert – im Falle der Lie-Gruppen-Integratoren ist die Toda-Lattice-Gleichung [100, 38, 25, 58] ein hervorragendes. Beim Toda-Lattice handelt es sich um ein eindimensionales Teilchengitter mit einem exponentiell abfallenden Potential, berücksichtigt werden nur die Wechselwirkungen zwischen benachbarten Teilchen. Die Dynamik eines solchen Systems ist durch Newtons Bewegungsgleichungen gegeben:

$$x_n'' = -\exp(-(x_{n+1} - x_n)) + \exp(-(x_n - x_{n-1})). \tag{1.1}$$

Man stellt dabei üblicherweise periodische Randbedingungen, die Teilchen befinden sich also in einem Ring. Die Gleichungen (1.1) gelten für $n = 1, \ldots, N$, wobei $x_0 := x_N$ und $x_{N+1} := x_1$. Möglich ist aber auch, am linken Rand $x_0 = -\infty$ und am rechten Rand $x_{N+1} = \infty$ zu fordern.

Nach Flaschka [38] setzt man $a_n = -\frac{1}{2}x_n'$ und $b_n = \frac{1}{2}\exp(\frac{1}{2}(x_n - x_{n+1}))$. Für a_n, b_n ergeben sich dann die Gleichungen

$$\begin{aligned} a_n' &= 2(b_n^2 - b_{n-1}^2) \\ b_n' &= b_n(a_{n+1} - a_n). \end{aligned}$$

Mit den zyklischen Tridiagonalmatrizen

$$L = \begin{pmatrix} a_1 & b_1 & & & & b_N \\ b_1 & a_2 & b_2 & & & \\ & b_2 & a_3 & b_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & b_{N-2} & a_{N-1} & b_{N-1} \\ b_N & & & & b_{N-1} & a_N \end{pmatrix} \tag{1.2}$$

$$B(L) = \begin{pmatrix} 0 & b_1 & & & & -b_N \\ -b_1 & 0 & b_2 & & & \\ & -b_2 & 0 & b_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -b_{N-2} & 0 & b_{N-1} \\ b_N & & & & -b_{N-1} & 0 \end{pmatrix} \tag{1.3}$$

ergibt sich die Matrix-Differentialgleichung

$$L' = B(L)L - LB(L). \tag{1.4}$$

Wie wir unten zeigen werden, läßt eine solche Differentialgleichung die Eigenwerte der Lösung L unverändert, sie bilden eine Invariante des Systems. Der von einer solchen Gleichung generierte Fluß wird daher als isospektral bezeichnet, man siehe auch [14, 15]. Standard-Integrationsverfahren wie Runge-Kutta-Verfahren oder lineare Mehrschrittverfahren erhalten für gewöhnlich nur lineare Invarianten, für quadratische Invarianten lassen sich spezielle Verfahren konstruieren (z.B. die Gauß-Methoden). Polynomiale Invarianten vom Grad 3 oder höher lassen sich nicht automatisch erhalten, siehe [47], p.102, Theorem 3.3.

Allgemein heißt eine Funktion $I(y)$ ein erstes Integral einer Differentialgleichung $y' = f(y)$, wenn I entlang einer Lösungstrajektorie konstant bleibt, also $I_y(y)f(y) = 0$. Nun sind die Eigenwerte der Matrix L erste Integrale. Um erste Integrale auch in der numerischen Lösung zu erhalten, kann nach jedem Integrationsschritt eine Projektion auf die durch $M = \{L : I(L) = I(L_0)\}$ definierte Mannigfaltigkeit vorgenommen werden.

Die geometrische Integration geht einen anderen Weg: Um eine Invariante zu erhalten, verwendet man einfach einen Ansatz, der automatisch die Invarianten unverändert läßt. Wir setzen hier

$$L(t) = T(t)L_0T(t)^{-1}, \quad T(0) = I \tag{1.5}$$

und bestimmen $T(t)$ so, daß die Differentialgleichung (1.4) erfüllt ist. Differentiation von $L(t)$ ergibt

$$\begin{aligned} L'(t) =& T'(t)L_0T(t)^{-1} - T(t)L_0T(t)^{-1}T'(t)T(t)^{-1} \\ =& T'(t)T(t)^{-1}L(t) - L(t)T'(t)T(t)^{-1}, \end{aligned} \tag{1.6}$$

mit $B = T'T^{-1}$ also gerade (1.4). Wir bestimmen daher T als Lösung der Differentialgleichung

$$T'(t) = B(L(t))T(t), \quad T(0) = I, \tag{1.7}$$

und somit erfüllt $L(t) = T(t)L_0T(t)^{-1}$ die Gleichung (1.4). Mit dem Existenz- und Eindeutigkeitssatz von Picard-Lindelöf folgt daraus gleichzeitig, daß für jede Lösung eine Darstellung (1.5) mit einem $T(t)$ nach (1.7) existiert. Damit ist gleichzeitig gezeigt, daß die Eigenwerte der Matrix L eine Invariante des Systems sind.

Diskretisieren wir die Gleichung (1.7) mit einem geeigneten numerischen Verfahren, und setzen $L_n = T_nL_0T_n^{-1}$, so sind in der Tat alle Eigenwerte der Matrix $L(t)$ in der numerischen Lösung L_n invariant.

Allerdings ist die exakte Matrix $L(t)$ auch symmetrisch, was für beliebige Startwerte L_0 nur für orthogonales $T(t)$ erfüllt werden kann. Auch die Gleichung (1.7) besitzt ein erstes Integral, es gilt

$$\begin{aligned} \frac{d}{dt}(T^TT) =& T'^TT + T^TT' \\ =& T^T(B^T + B)T = 0, \end{aligned}$$

da die Matrix B laut Konstruktion schiefsymmetrisch ist. Wollen wir die Symmetrie der Matrix L erhalten, so müssen wir die Orthogonalität der Matrix T sichern. Orthogonalität ist eine quadratische Invariante, hier kämen im Bereich der Runge-Kutta-Verfahren im wesentlichen Gauß-Methoden in Frage (siehe [47], S. 97ff.). Man kann aber analog zum vorhergehenden die Lösung mit der Cayley-Abbildung cay parametrisieren, so daß die Invariante automatisch erhalten wird. Wir setzen

$$T(t) = \mathrm{cay}(R(t)) := (I - R(t))^{-1}(I + R(t)), \tag{1.8}$$

wobei sich $R(t) = -(I - T(t))(I + T(t))^{-1}$ ergibt. Man überprüft leicht, daß $T(t)$ genau dann orthogonal ist, wenn $R(t)$ schiefsymmetrisch ist.

Der Ansatz (1.8) liefert nach Differentiation und Einsetzen in (1.7) eine explizite Differentialgleichung für $R'(t)$.

$$\begin{aligned} T'(t) =& 2(I - R(t))^{-1} R'(t)(I - R(t))^{-1} \quad \Rightarrow \\ R'(t) =& \frac{1}{2}(I - R(t))B(L(T(R(t))))(I + R(t)) \end{aligned} \tag{1.9}$$

Ist $R(t)$ schiefsymmetrisch, so gilt dies auch für die rechte Seite von (1.9), d.h., dann gilt $R'(t) + R'(t)^T = 0$. Somit ist der lineare Unterraum der schiefsymmetrischen Matrizen invariant gegenüber dem Fluß der Differentialgleichung. Ein invarianter linearer Unterraum wird von vielen gebräuchlichen Integrationsverfahren für gewöhnliche Differentialgleichungen erhalten, so von Runge-Kutta-Verfahren oder linearen Mehrschrittverfahren. Wir bemerken außerdem, daß wir von einer Differentialgleichung in einer nichtaffinen Menge (Gruppe der orthogonalen Matrizen) zu einer Differentialgleichung in einem Vektorraum (schiefsymmetrische Matrizen) übergegangen sind.

Etwas abstrakter können wir die obige Vorgehensweise wie folgt zusammenfassen: Gegeben ist eine Differentialgleichung $L' = B(L)L - LB(L)$, deren Lösung auf der Mannigfaltigkeit $M =: \{L : \mathrm{spec}(L) = \mathrm{spec}(L_0)\}$ verbleibt. Die Mannigfaltigkeit wird durch eine Gruppe G von Transformationen parametrisiert, hier wird G durch die orthogonalen Matrizen mit Determinante 1 (diejenige Zusammenhangskomponente der orthogonalen Matrizen, die die Einheitsmatrix enthält) dargestellt. Die einer Matrix T zugehörige Transformation $\Lambda(T)$ ergibt sich zu $\Lambda(T) : X \mapsto \Lambda(T)(X) = TXT^{-1}$. Die Transformationsgruppe (eine nichtlineare Struktur) wird wiederum geeignet durch eine lineare Struktur $\mathfrak{g}$ (Vektorraum der schiefsymmetrischen Matrizen) dargestellt. Die Differentialgleichung kann dabei sowohl in der Gruppe als auch im Vektorraum formuliert werden. Zur numerischen Lösung im Vektorraum kann ein beliebiges Standard-Verfahren genutzt werden, das die Vektorraumstruktur (also lineare erste Integrale) respektiert. Im folgenden Diagramm ist die gesamte Vorgehensweise noch einmal veranschaulicht.

$$\begin{array}{ccccc} \mathfrak{g} & \xrightarrow{\mathrm{cay}} & G & \xrightarrow{\Lambda} & M \\ \downarrow & & \downarrow & & \downarrow \\ \text{ODE} & \longleftarrow & g'(t) = f(g(t)) & \longleftarrow & y' = F(y) \\ (1.9) & \longleftarrow & (1.7) & \longleftarrow & (1.4) \end{array}$$

Die Transformationsgruppe ist eine Lie-Gruppe, der sie parametrisierende Vektorraum die Lie-Algebra. Die Parametrisierung mit cay ist nur für spezielle Lie-Gruppen möglich (quadratische Gruppen, siehe Abschnitt 1.3.1).

1.2 Analysis auf Mannigfaltigkeiten

Wir stellen hier kurz einige Grundbegriffe bereit. Für ausführlichere Darstellungen siehe man [4], [76], [67].

1.2.1 Differenzierbare Mannigfaltigkeiten

Anschaulich wollen wir uns hier unter einer m-dimensionalen differenzierbaren Mannigfaltigkeit M eine m-dimensionale Teilmenge des $\mathbb{R}^n$ ($n \geq m$) vorstellen, die genügend glatt ist. Lokal läßt sich diese Mannigfaltigkeit durch m krummlinige Koordinaten parametrisieren. Diese krummlinigen Koordinaten lassen sich zumeist nicht auf die gesamte Mannigfaltigkeit fortsetzen, sie erlauben nur die Parametrisierung eines bestimmten Bereichs der Mannigfaltigkeit. Diesen Bereich bezeichnen wir als Koordinatenkarte U. Die krummlinigen Koordinaten auf U sind durch eine Abbildung $\chi : U \mapsto V \subset \mathbb{R}^m$ gegeben.

Die gesamte Mannigfaltigkeit sei nun von solchen Karten U_α überdeckt. Eine wichtige Eigenschaft ist natürlich die Glattheit der krummlinigen Koordinaten. Betrachtet man M als eingebettet in den $\mathbb{R}^n$, so kann man die Glattheit der krummlinigen Koordinaten durch Glattheitsanforderungen an die Koordinatenfunktionen beschreiben.

Will man sich von der Vorstellung lösen, daß die Mannigfaltigkeit in den $\mathbb{R}^n$ eingebettet ist, so lassen sich natürlich keine direkten Glattheitsforderungen an die Koordinatenfunktionen stellen. Man fordert daher Glattheit der Koordinatentransformationen, die sich auf den Überlappungen zweier Karten ergeben.

Definition 1.2.1. *Eine m-dimensionale differenzierbare Mannigfaltigkeit besteht aus einer Menge M und einer Familie von Karten U_α mit bijektiven Koordinatenfunktionen $\chi_\alpha : U_\alpha \to V_\alpha \subset \mathbb{R}^m$, die folgende Eigenschaften besitzen:*

1. *Die Karten überdecken M, also $\bigcup_\alpha U_\alpha = M$.*

2. *Die V_α sind offene Mengen.*

3. *Auf der Überlappung $U_\alpha \cap U_\beta$ zweier Karten ist die Koordinatentransformation unendlich oft differenzierbar, d.h., die Abbildung*

$$\chi_\alpha \circ \chi_\beta^{-1} : \chi_\beta(U_\alpha \cap U_\beta) \to \chi_\alpha(U_\alpha \cap U_\beta)$$

   ist unendlich oft differenzierbar.

4. *Zu zwei verschiedenen Punkten $x_\alpha \in U_\alpha$, $x_\beta \in U_\beta$ existieren Umgebungen (offene Mengen) W_α, W_β von $\chi_\alpha(x_\alpha)$, $\chi_\beta(x_\beta)$ in V_α, V_β, so daß*

$$\chi_\alpha^{-1}(W_\alpha) \cap \chi_\beta^{-1}(W_\beta) = \emptyset.$$

Die vierte Forderung liefert uns eine Topologie auf M, die dem zweiten Trennungsaxiom genügt, M ist somit ein Hausdorffraum. Für viele Resultate ist diese Annahme nicht zwingend, sie vermeidet jedoch unnötige technische Komplikationen.

Die Glattheitsanforderungen lassen sich auch auf k-mal stetig differenzierbare Funktionen abschwächen. Wir werden im folgenden unter glatt stets unendlich oft differenzierbar verstehen.

Beispiel 1.2.1. *Die einfachste differenzierbare Mannigfaltigkeit ist der $\mathbb{R}^m$ selbst. Die Koordinaten sind hier durch eine einzige Karte gegeben. Jede andere differenzierbare Mannigfaltigkeit mit nur einer Karte U kann analog mit der Menge $V = \chi(U)$ identifiziert werden.*

Beispiel 1.2.2. *Die dreidimensionale Einheitskugel $S^2 = \{(x, y, z) : x^2 + y^2 + z^2 = 1\}$ stellt zusammen mit zwei stereographischen Projektionen eine differenzierbare Mannigfaltigkeit dar. Die Karten sind*

$$U_1 = S^2 \setminus \{(0, 0, 1)\} \text{ und } U_2 = S^2 \setminus \{(0, 0, -1)\}$$

mit den Koordinatenfunktionen

$$\chi_1 = (\xi_1, \eta_1) = (\frac{x}{1-z}, \frac{y}{1-z}) \text{ und } \chi_2 = (\xi_2, \eta_2) = (\frac{x}{1+z}, \frac{y}{1+z}).$$

Drückt man auf der Überlappung der beiden Karten die Koordinaten in U_1 durch die in U_2 aus, so ergibt sich

$$(\xi_1, \eta_1) = (\frac{\xi_2}{\xi_2^2 + \eta_2^2}, \frac{\eta_2}{\xi_2^2 + \eta_2^2}).$$

Wegen $(x, y, z) \in U_1 \cap U_2$ ist $(\xi_2, \eta_2) \neq (0, 0)$ und daher ist die Koordinatentransformation unendlich oft differenzierbar.

Der Tatsache, daß man für eine Karte verschiedene Koordinatenfunktionen angeben kann (nämlich all jene, die ineinander durch einen Diffeomorphismus überführbar sind), kann man auf zweierlei Art Rechnung tragen. Prinzipiell wäre es möglich, in einem (wie auch immer) ausgezeichneten Koordinatensystem mathematische Eigenschaften zu beschreiben, und bei mathematischen Aussagen jeweils die Invarianz bezüglich der Wahl des Koordinatensystems zu zeigen. Besser ist jedoch, koordinatenunabhängige Definitionen für mathematische Eigenschaften zu wählen. Dies erreicht man dadurch, daß jede Eigenschaft in ihrer Wirkung auf die Koordinaten beschrieben wird. Wir sprechen anstelle von Koordinaten allgemein von Funktionen, die die Mannigfaltigkeit nach $\mathbb{R}$ abbilden. Jedes mathematische Objekt wird durch seine Wirkung auf diese Funktionen definiert.

1.2.2 Tangentenvektoren und Vektorfelder

Sei $x \in M$ ein Punkt einer m-dimensionalen Mannigfaltigkeit M. M sei in den $\mathbb{R}^n$ durch $M = \{x \in \mathbb{R}^n : f(x) = 0\}$ eingebettet, wobei $f = (f_1, \ldots, f_{n-m})$. Die Gradienten $\nabla f_1, \ldots, \nabla f_{n-m}$ stehen dann senkrecht auf M, während ihr orthogonales Komplement tangential zu M ist. Wir bezeichnen dieses orthogonale Komplement $TM|_x$ als Tangentialraum in x, die Elemente als Tangentenvektoren. Nun ist die Mannigfaltigkeit nicht immer als Einbettung in den $\mathbb{R}^n$ gegeben. Für eine koordinatenunabhängige Definition bieten sich zwei leicht als äquivalent identifizierte Zugänge an.

Ein Tangentenvektor v im Punkt $x \in M$ wird als Richtungsableitung in diesem Punkt definiert, also als lineares, homogenes Funktional auf der Menge der (in einer Umgebung von x) glatten Funktionen, das der Produktregel der Differentiation genügt:

$$v(g \cdot h) = g(x) \cdot v(h) + v(g) \cdot h(x).$$

Anstelle einer Richtungsableitung kann man aber auch direkt die Richtung als Ableitung einer Kurve zur Definition von Tangentenvektoren heranziehen. Wir betrachten differenzierbare Kurven $\phi(t)$ auf M mit $\phi(0) = x_0$. Zwei Kurven $\phi(t), \psi(t)$ werden äquivalent genannt, wenn ihre ersten

Ableitungen in einer Karte übereinstimmen. Die Ableitungen stimmen dann in jeder Karte überein, da bei einer Koordinatentransformation $y = F(x)$

$$\left.\frac{d}{dt}\right|_{t=0} F(\phi(t)) = F_x|_{x=x_0}\,\phi'(0) = F_x|_{x=x_0}\,\psi'(0) = \left.\frac{d}{dt}\right|_{t=0} F(\psi(t)).$$

Definition 1.2.2. *Ein Tangentenvektor in x_0 ist eine Äquivalenzklasse von Kurven ϕ mit $\phi(0) = x_0$. Ist die Kurve in lokalen Koordinaten $x = (\phi_1(t), \ldots, \phi_m(t))$ gegeben, so schreiben wir*

$$v = \phi'(0) = \sum_{i=1}^{m} \phi_i'(0)\frac{\partial}{\partial x_i} = \sum_{i=1}^{m} v_i \frac{\partial}{\partial x_i}.$$

Die Ausdrücke $\frac{\partial}{\partial x_i}$ sind die kanonischen Basisvektoren bezüglich der Koordinaten x.

Diese Schreibweise gestattet es, die Tangentenvektoren in x im Sinne des ersten Zugangs direkt als Ableitungen auf dem Raum der glatten Funktionen aufzufassen. Definiert man für eine glatte Funktion $f : M \to \mathbb{R}$

$$v(f) = \left.\frac{d}{dt}\right|_{t=0} f(\phi(t)),$$

so ergibt sich in lokalen Koordinaten

$$v(f) = \sum_i v_i \frac{\partial f}{\partial x_i}.$$

Wir werden diese Schreibweise des öfteren verwenden.

Die Komponenten v_i hängen natürlich vom Koordinatensystem ab, bei einer Koordinatentransformation $y = F(x)$ ergibt sich für die Komponenten v_i^x bzw. v_i^y in den Koordinaten x bzw. y gerade

$$\begin{aligned} v(f) &= \sum_i v_i^x \frac{\partial f}{\partial x_i} = \sum_{i,j} v_i^x \frac{\partial f}{\partial y_j}\frac{\partial y_j}{\partial x_i} \\ \Rightarrow v_j^y &= \sum_i \frac{\partial y_j}{\partial x_i} v_i^x = v(y_j). \end{aligned}$$

Zur Identifikation eines Tangentenvektors v gehört neben den koordinatenabhängigen Komponenten v_i auch der Punkt, an dem der Tangentenvektor "angeheftet" ist (also der Punkt, in dem differenziert wird).

Definition 1.2.3. *Den Vektorraum aller Tangentenvektoren im Punkt $x \in M$ bezeichnen wir als Tangentialraum $TM|_x$. Die Vereinigung aller Tangentialräume $TM = \bigcup_{x\in M} TM|_x$ heißt Tangentenbündel. Die kanonische Projektion $\pi : TM \to M$ ordnet jedem Tangentenvektor $v \in TM|_x$ den Basispunkt x zu.*

Definieren wir Tangentenvektoren in jedem Punkt der Mannigfaltigkeit, so gelangen wir zum Begriff des Vektorfeldes.

Definition 1.2.4. *Ein Vektorfeld ist eine Abbildung $v : M \to TM$ mit $\pi \circ v = id$. Anstelle von $v(x)$ schreiben wir $v|_x$. Ein Vektorfeld heißt glatt, wenn die Darstellung in lokalen Koordinaten in jeder Karte glatt ist.*

1.2.3 Fluß und Integralkurve

Definition 1.2.5. *Eine Integralkurve eines Vektorfeldes v ist eine Kurve $\phi : \mathbb{R} \to M$ mit*

$$\phi'(t) = v|_{\phi(t)}\,.$$

Die Bestimmung einer Integralkurve durch einen gegebenen Punkt $x \in M$ entspricht der Lösung eines Anfangswertproblems für gewöhnliche Differentialgleichungen. Aufgrund der Glattheit des Vektorfeldes sind die Voraussetzungen des Existenz- und Eindeutigkeitssatzes von Picard-Lindelöff erfüllt. Eine Integralkurve ist daher durch Angabe eines Punktes eindeutig bestimmt.

Definition 1.2.6. *Wir bezeichnen die Integralkurve ϕ durch x (mit $\phi(0) = x$) mit*

$$\Psi(\varepsilon, x) = \phi(\varepsilon).$$

Fixiert man die Variable x und variiert ε, so durchläuft $\Psi(\varepsilon, x)$ gerade die Integralkurve. Umgekehrt erhält man für festes ε eine Abbildung der Mannigfaltigkeit in sich. Für glatte Vektorfelder ist diese Abbildung bijektiv, denn die Umkehrung ist durch die Integralkurven des Vektorfeldes $-v$ gegeben. Sie stellt daher eine Transformation der Mannigfaltigkeit dar.

Definition 1.2.7. *Die Abbildung $\Psi_\varepsilon(x) := \Psi(\varepsilon, x)$ heißt der Fluß des Vektorfeldes v.*

Der Fluß eines Vektorfeldes ist eine einparametrige Transformationsgruppe, es gilt

$$\Psi(\delta, \Psi(\varepsilon, x)) = \Psi(\delta + \varepsilon, x) \tag{1.10}$$

$$\Psi(0, x) = x \tag{1.11}$$

$$\frac{\partial \Psi(\varepsilon, x)}{\partial \varepsilon} = v|_{\Psi(\varepsilon, x)}\,. \tag{1.12}$$

Die Eigenschaften (1.10), (1.11) definieren gerade eine einparametrige Transformationsgruppe, und Eigenschaft (1.12) ordnet umgekehrt jeder einparametrigen Transformationsgruppe ein Vektorfeld zu. Dabei ist v wohldefiniert, denn aus (1.10) folgt

$$v|_{\Psi(\varepsilon,x)} = \frac{\partial \Psi(\varepsilon, x)}{\partial \varepsilon} = \left.\frac{\partial \Psi(\varepsilon + \delta, x)}{\partial \delta}\right|_{\delta=0} = \left.\frac{\partial \Psi(\delta, \Psi(\varepsilon, x)))}{\partial \delta}\right|_{\delta=0},$$

das heißt, $v|_{\Psi(\varepsilon,x)}$ hängt nur von $\Psi(\varepsilon, x)$ ab und nicht von der speziellen Wahl von x, ε.

Man bezeichnet das Vektorfeld v als infinitesimalen Generator der Transformationsgruppe. Ist eine solche Transformationsgruppe gegeben, so läßt sich ihr infinitesimaler Generator aus der Eigenschaft (1.12) des Flusses bestimmen. Umgekehrt bestimmt natürlich der infinitesimale Generator die Transformationsgruppe als seinen Fluß. Wir haben also eine 1-1 Korrespondenz zwischen den Vektorfeldern und den einparametrigen Transformationsgruppen.

Wir schreiben den von einem Vektorfeld v erzeugten Fluß als

$$\exp(\varepsilon v)x = \Psi_v(\varepsilon, x).$$

Dies ist möglich, da $\Psi_v(\varepsilon, x)$ für festes x nur vom Produkt εv abhängt, nicht aber von der speziellen Wahl von ε, x, genauer:

$$\Psi_v(\varepsilon, x) = \Psi_{\varepsilon v}(1, x).$$

Die Eigenschaften des Flusses schreiben sich somit sehr einprägsam als

$$\exp((\delta+\varepsilon)v) = \exp(\delta v)\exp(\varepsilon v) \tag{1.13}$$

$$\exp(0v) = \mathrm{id} \tag{1.14}$$

$$\frac{\partial}{\partial\varepsilon}\exp(\varepsilon v)x = v\circ\exp(\varepsilon v)x. \tag{1.15}$$

Betrachten wir zwei Beispiele.

Beispiel 1.2.3. *Sei $M=\mathbb{R}$. Zum Vektorfeld $v=\frac{\partial}{\partial x}$ gehört der Fluß $\exp(\varepsilon v)x = x+\varepsilon$. Für das Vektorfeld $v=x\frac{\partial}{\partial x}$ ergibt sich der Fluß $\exp(\varepsilon v)x = e^{\varepsilon}x$.*

Beispiel 1.2.4. *Im n-dimensionalen Vektorraum $M=\mathbb{R}^n$ erhalten wir analog zum konstanten Vektorfeld $v=\sum_i a_i\frac{\partial}{\partial x_i}$ den Fluß $\exp(\varepsilon v)x = x+\varepsilon a$, eine reine Translation. Der Fluß des Vektorfeldes $v=\sum_i\left(\sum_j a_{ij}x_j\right)\frac{\partial}{\partial x_i}$ ergibt die lineare Transformation $\exp(\varepsilon v)x = e^{A\varepsilon}x$ mit der Matrix $A=(a_{ij})$.*

Sei nun ein Vektorfeld v gegeben mit dem Fluß $\exp(\varepsilon v)x$. Uns interessiert die Änderung einer Funktion $f: M\to\mathbb{R}$ unter diesem Fluß. In lokalen Koordinaten ergibt sich:

$$\frac{\partial}{\partial\varepsilon}f(\exp(\varepsilon v)x) = \sum_i \frac{\partial f}{\partial x_i}(\exp(\varepsilon v)x)\ v_i|_{\exp(\varepsilon v)x} = v(f)|_{\exp(\varepsilon v)x} \tag{1.16}$$

Insbesondere ist

$$\left.\frac{\partial f(\exp(\varepsilon v)x)}{\partial\varepsilon}\right|_{\varepsilon=0} = v(f)(x).$$

Die Änderung der Funktion f durch den Fluß ist also gerade

$$f(\exp(\varepsilon v)x) = f(x)+\varepsilon v(f)(x)+\mathcal{O}\left(\varepsilon^2\right).$$

Weitere Differentiationen von (1.16) liefern

$$\frac{\partial^k f(\exp(\varepsilon v)x)}{\partial\varepsilon^k} = v(v(\ldots(f)\ldots)(x) =: v^k(f)(x).$$

Wir erhalten für f die Taylorentwicklung

$$f(\exp(\varepsilon v)x) = f(x)+\varepsilon v(f)(x)+\frac{\varepsilon^2}{2}v^2(f)(x)+\cdots+\frac{\varepsilon^k}{k!}v^k(f)(x)+\mathcal{O}\left(\varepsilon^{k+1}\right).$$

Die unendliche Reihe wird als Lie-Reihe bezeichnet. Ihr Erscheinungsbild läßt sich der Taylorentwicklung der klassischen Exponentialfunktion noch weiter annähern, wenn wir $\exp(\varepsilon v)$ als Abbildung von $C^\infty(M)\to C^\infty(M)$ vermöge $\exp(\varepsilon v)(f)(x) := f(\exp(\varepsilon v))(x)$ auffassen. Dann erhalten wir nämlich

$$\exp(\varepsilon v) := \sum_{k=0}^{\infty}\frac{\varepsilon^k v^k}{k!}.$$

Analog lassen sich vektorwertige Funktionen in Lie-Reihen entwickeln, hier wird die Entwicklung einfach komponentenweise vorgenommen. Betrachten wir die Koordinatenfunktionen x als vektorwertige Funktion, so erhalten wir

$$\exp(\varepsilon v)x = x+\varepsilon v+\frac{1}{2}\varepsilon^2 v(v)(x)+\ldots.$$

1.2.4 Das Differential

Sei $F : M \to N$ eine glatte Abbildung zwischen Mannigfaltigkeiten. Dann bildet F Kurven auf M in Kurven auf N ab. Dadurch wird eine Abbildung der Tangentenvektoren auf M in Tangentenvektoren auf N induziert.

Definition 1.2.8. *Die Abbildung*

$$dF : TM|_x \to TN|_{F(x)} \tag{1.17}$$

$$v \mapsto dF(v) \text{ mit } dF(v)(f) = v(f \circ F) \tag{1.18}$$

heißt Differential der Abbildung F in x.

Die folgenden Sätze ergeben sich unmittelbar aus der Definition.

Satz 1.2.1. *Das Differential ist eine lineare Abbildung.*

Satz 1.2.2. *Sind $F : M \to N$ und $H : N \to P$ glatte Abbildungen, so gilt*

$$d(H \circ F) = dH \circ dF.$$

Satz 1.2.3. *Seien x_i bzw. y_j lokale Koordinaten in M bzw. N in Umgebungen von x bzw. $y = F(x)$. Ist $v = \sum_i v_i \frac{\partial}{\partial x_i} \in TM|_x$ in lokalen Koordinaten gegeben, so ergibt sich das Differential $dF(v) \in TN|_y$ zu*

$$dF(v) = \sum_{i,j} \frac{\partial F_j}{\partial x_i} v_i \frac{\partial}{\partial y_j}.$$

Ist v als Ableitung $\phi'(0)$ einer parametrisierten Kurve $\phi(t)$ gegeben, so ergibt sich dF als Ableitung der Kurve $F \circ \phi$.

Stellt man v und w als Spaltenvektoren dar, so lautet die Definition einfach $dF(v) = w = F_x v$ – Multiplikation mit der Jacobi-Matrix. Satz 1.2.2 ist damit die bekannte Kettenregel für die Differentiation von Vektorfunktionen, d.h., die Jacobi-Matrix zweier verketteter Funktionen ist das Produkt der Jacobi-Matrizen.

Definition 1.2.9. *Zwei Vektorfelder $v \in TM$ und $w \in TN$ werden F-verträglich genannt, wenn $dF(v) = w$.*

1.2.5 Die Lie-Klammer

Definition 1.2.10. *Seien v, w zwei Vektorfelder auf einer Mannigfaltigkeit M. Ihre Lie-Klammer ist das Vektorfeld $[v, w]$, gegeben durch*

$$[v, w](f) = v(w(f)) - w(v(f)).$$

In lokalen Koordinaten erhalten wir

$$\begin{aligned}[v, w](f) &= \sum_i v_i \frac{\partial}{\partial x_i} \left(\sum_j w_j \frac{\partial f}{\partial x_j} \right) - \sum_i w_i \frac{\partial}{\partial x_i} \left(\sum_j v_j \frac{\partial f}{\partial x_j} \right) \\ &= \sum_j \left(\sum_i v_i \frac{\partial w_j}{\partial x_i} - w_i \frac{\partial v_j}{\partial x_i} \right) \frac{\partial f}{\partial x_j}.\end{aligned}$$

Die Lie-Klammer läßt sich damit also auch als $[v, w] = v(w) - w(v)$ schreiben, wobei die Ausdrücke $v(w)$ bzw. $w(v)$ so zu interpretieren sind, daß die Differentialoperatoren v bzw. w auf die Komponenten der vektorwertigen Funktionen w bzw. v anzuwenden sind.

Satz 1.2.4. *Die Lie-Klammer besitzt die folgenden Eigenschaften:*

1. *Bilinearität:* $[\alpha_1 v_1 + \alpha_2 v_2, w] = \alpha_1[v_1, w] + \alpha_2[v_2, w]$
2. *Schiefsymmetrie:* $[v, w] = -[w, v]$
3. *Jacobi-Identität:* $[u, [v, w]] + [v, [w, u]] + [w, [u, v]] = 0$

Beweis. : Die Eigenschaften 1 und 2 sind trivial. Zum Nachweis der Eigenschaft 3 betrachten wir

$$\begin{aligned}[u, [v, w]](f) =& u([v, w](f)) - [v, w](u(f)) \\ =& u(v(w(f))) - u(w(v(f))) - v(w(u(f))) + w(v(u(f))).\end{aligned}$$

Hier treten jeweils eine gerade bzw. eine ungerade Permutation mit einem positiven bzw. negativen Vorzeichen auf. Zyklisches Vertauschen von u, v, w zeigt, daß in der Jacobi-Identität jede gerade und jede ungerade Permutation einmal mit positivem und einmal mit negativem Vorzeichen auftritt. □

Die Lie-Klammer ist mit dem Differential verträglich:

Satz 1.2.5. *Seien v, w Vektorfelder auf M, und V, W Vektorfelder auf N. Möge F von M nach N abbilden, und gelte: $V = dF(v)$, $W = dF(w)$. Dann gilt*

$$dF([v, w]) = [V, W]$$

Beweis. Sei $f : N \to \mathbb{R}$. Anwenden der Definitionen von Lie-Klammer und Differential liefert

$$[V, W](f(y)) = V(W(f)) - W(V(f)) = v(W(f) \circ F) - w(V(f) \circ F).$$

Dabei sind $W(f), V(f)$ Funktionen von N nach $\mathbb{R}$. Nach Definition gilt $(W(f))(F(x)) = w(f \circ F)(x)$, es ergibt sich gerade

$$[V, W](f(y)) = v(w(f \circ F) - w(v(f \circ F)) = dF([v, w]).$$

□

Die so definierte Lie-Klammer begegnet uns an verschiedenen Stellen wieder. So beschreibt sie, inwieweit die Flüsse zweier Vektorfelder kommutieren:

Satz 1.2.6. *Seien v, w Vektorfelder auf M. Die Lie-Klammer $[v, w]$ ist der infinitesimale Generator der Transformationsgruppe*

$$\psi(\varepsilon, x) = \exp(-\sqrt{\varepsilon}w) \exp(-\sqrt{\varepsilon}v) \exp(\sqrt{\varepsilon}w) \exp(\sqrt{\varepsilon}v)x.$$

Zum Beweis siehe man [76].

Satz 1.2.7. *Es gilt $[v, w] = 0$ genau dann, wenn die von v und w erzeugten Flüsse kommutieren, d.h.,*

$$\exp(\varepsilon v) \exp(\theta w)x = \exp(\theta w) \exp(\varepsilon v)x \tag{1.19}$$

Eine interessante Frage ist, wann Systeme von Vektorfeldern invariante Mannigfaltigkeiten generieren. Ein einzelnes Vektorfeld kann als Basis eines eindimensionalen linearen Raumes von Vektorfeldern angesehen werden. Das Vektorfeld generiert Integralkurven, und diese sind eindimensionale Untermannigfaltigkeiten, die invariant bezüglich der durch das Vektorfeld generierten Transformationsgruppe sind. Ist dies stets auch bei mehrdimensionalen Systemen von Vektorfeldern der Fall? Generiert ein n-dimensionales System von Vektorfeldern stets ein System von n-dimensionalen invarianten Untermannigfaltigkeiten? Eine Antwort wird im folgenden gegeben:

Definition 1.2.11. *Ein System von Vektorfeldern $v_1, \ldots, v_n$ auf der Mannigfaltigkeit M ist in Involution, wenn glatte reelle Funktionen $c_{ijk}(x)$ existieren, so daß in jedem $x \in M$ gilt*

$$[v_i, v_j] = \sum_k c_{ijk}(x) v_k.$$

Mit zwei Vektorfeldern ist natürlich auch deren Lie-Klammer tangential zu einer Mannigfaltigkeit. Insofern kann ein System von Vektorfeldern nur dann eine Integralmannigfaltigkeit besitzen, wenn es in Involution ist. Es gilt sogar:

Satz 1.2.8. *(Frobenius): Seien $v_1, \ldots, v_r$ glatte Vektorfelder. Das System $v_1, \ldots, v_r$ generiert invariante r-dimensionale Untermannigfaltigkeiten genau dann, wenn es in Involution ist.*

1.3 Lie-Gruppen und Lie-Algebren

Für eine ausführliche Darstellung verweisen wir auf [108], [76], [105].

1.3.1 Lie-Gruppen

Der Begriff der Gruppe wird als bekannt vorausgesetzt. Von Interesse sind für uns insbesondere Matrizen-Gruppen, mit der Matrizenmultiplikation als Gruppenoperation. Erwähnt seien hier:

- $GL(n)$ – die Gruppe der regulären n-dimensionalen Matrizen über den reellen Zahlen. Bezeichnung: allgemeine lineare Gruppe (**g**eneral **l**inear group).
- $O(n)$ – die Gruppe der orthogonalen Matrizen der Dimension n. Sie zerfällt in 2 Zusammenhangskomponenten – Matrizen mit Determinante $+1$ und -1.
- $SO(n)$ – die orthogonalen Matrizen der Dimension n mit Determinante $+1$. Die Lie-Algebra ist $\mathfrak{so}(n)$ – die Menge der schiefsymmetrischen Matrizen.
- Sp(n) – die symplektische Gruppe. Sie ist als $\{X : X^T J X = J\}$ gegeben, wobei die feste Matrix J durch

  $$J = \begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix}. \tag{1.20}$$

  gegeben ist.
- Die symplektische Gruppe $Sp(n)$ und die Gruppe der orthogonalen Matrizen sind Spezialfälle sogenannter quadratischer Lie-Gruppen, die als $\{X : X^T F X = F\}$ für eine feste Matrix F gegeben sind. Die symplektische Gruppe ergibt sich für das oben erwähnte $F = J$, die

orthogonalen Matrizen ergeben sich mit $F = I$. Besonderheit quadratischer Lie-Gruppen ist, daß man sie mit Hilfe der eingangs der Arbeit erwähnten Cayley-Abbildung parametrisieren kann.

Eine Lie-Gruppe ist eine Gruppe, die mit der Struktur einer Mannigfaltigkeit ausgestattet ist. Einfachstes Beispiel ist die Gruppe $\mathbb{R}$. Auch $GL(n)$ wird mit der Struktur einer Mannigfaltigkeit ausgestattet, wenn man $GL(n)$ als Teilmenge des $\mathbb{R}^{n^2}$ betrachtet.

Definition 1.3.1. *Eine r-parametrige Lie-Gruppe ist eine Gruppe G, die mit der Struktur einer r-dimensionalen Mannigfaltigkeit ausgestattet ist. Dabei sind sowohl die Gruppenoperation als auch die Inversion glatte Abbildungen zwischen Mannigfaltigkeiten, also*

$$\begin{aligned} m :& G \times G \to G, m(g,h) = gh \\ i :& G \to G, i(g) = g^{-1} \end{aligned}$$

sind glatte Abbildungen.

Lie-Gruppen treten zumeist als Transformationsgruppen von Mannigfaltigkeiten auf. So ist $SO(2)$ die Gruppe der Drehungen der Ebene oder $GL(n)$ die Gruppe der invertierbaren linearen Transformationen in $\mathbb{R}^n$.

1.3.2 Lie-Algebren

Lie-Algebren lassen sich unabhängig vom Begriff der Lie-Gruppe definieren.

Definition 1.3.2. *Eine Lie-Algebra ist ein Vektorraum $\mathfrak{g}$ mit einem bilinearen, schiefsymmetrischen Produkt*

$$[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}, \tag{1.21}$$

das der Jacobi-Identität

$$[u,[v,w]] + [v,[w,u]] + [w,[u,v]] = 0 \tag{1.22}$$

genügt.

Im Zusammenhang mit Lie-Gruppen entstehen Lie-Algebren in natürlicher Weise aus Tangentenvektoren oder speziellen Vektorfeldern.

Sei G eine Lie-Gruppe. Für jedes Gruppenelement g liefert die Rechtsmultiplikation

$$R_g(h) = hg \tag{1.23}$$

einen Diffeomorphismus.

Definition 1.3.3. *Ein Vektorfeld v wird rechtsinvariant genannt, wenn*

$$dR_g(v|_h) = v|_{hg} \tag{1.24}$$

für alle $g, h \in G$.

Mit v und w ist natürlich auch jede Linearkombination von v und w rechtsinvariant. Die Menge der rechtsinvarianten Vektorfelder bildet also einen Vektorraum. Jedes rechtsinvariante Vektorfeld ist durch seinen Wert im Einselement bereits vollständig definiert, denn

$$v|_g = dR_g(v|_e). \tag{1.25}$$

Jeder Tangentenvektor in e liefert damit ein rechtsinvariantes Vektorfeld, da

$$dR_g(v|_h) = dR_g(dR_h(v|_e)) = dR_{hg}(v|_e) = v|_{hg}\,. \tag{1.26}$$

Definition 1.3.4. *Zu einer Lie-Gruppe G wird der Tangentialraum an die Identität e als ihre Lie-Algebra $\mathfrak{g} = TG|_e$ bezeichnet. Die Klammer der Lie-Algebra ist die Lie-Klammer für die zugehörigen rechtsinvarianten Vektorfelder.*

Wir überzeugen uns, daß die Eigenschaften einer Lie-Algebra gegeben sind.

Zum einen ist die Menge der rechtsinvarianten Vektorfelder abgeschlossen bezüglich der Lie-Klammer. $R_g : G \to G$ ist eine Abbildung von G auf sich. Somit ordnet dR_g jedem Vektorfeld v auf G ein Vektorfeld $dR_g(v)$ durch $R_g(v)|_{R_g(x)} = dR_g(v|_x)$ zu. Dabei sind rechtsinvariante Vektorfelder gerade Fixpunkte von dR_g für alle $g \in G$. Wenden wir Satz 1.2.5 auf die rechtsinvarianten Vektorfelder v, w an, so ergibt sich

$$dR_g([v, w]) = [dR_g(v), dR_g(w)]$$

Die rechte Seite ist nun nach Voraussetzung von g unabhängig, und somit ist auch $[v, w]$ rechtsinvariant.

Die Jacobi-Identität wurde bereits in (3) gezeigt.

Über die Interpretation als rechtsinvariante Vektorfelder wird also der Tangentialraum an e zur Lie-Algebra. Man spricht dabei auch von der rechtsinvarianten Version, analog existiert auch eine linksinvariante Version.

Die Lie-Algebra ist ein endlichdimensionaler Vektorraum von der selben Dimension wie die zugrundeliegende Lie-Gruppe.

Beispiel 1.3.1. *In $G = \mathbb{R}$ gibt es (bis auf Multiplikation mit einem konstanten Faktor) nur ein rechtsinvariantes Vektorfeld, dies ist $\frac{\partial}{\partial x}$.*

In $G = \mathbb{R}^+$ mit der Multiplikation als Gruppenoperation ist

$$v|_y = dR_y(v|_1) = y\, v|_1 \,.$$

In $SO(2)$ ist $v|_\theta = dR_\theta(v|_1) = v|_1$.

Beispiel 1.3.2. *Eine komplexere Aufgabenstellung ist die Bestimmung der Lie-Algebra der allgemeinen linearen Gruppe $GL(n)$, die mit $\mathfrak{gl}(n)$ bezeichnet wird.*

$GL(n)$ ist n^2-dimensional. Der Tangentialraum in I läßt sich auf natürliche Weise mit $GL(n)$ identifizieren. Ebenso identifiziert man die Lie-Algebra damit.

Wir wollen zunächst die Elemente der Lie-Algebra bestimmen. Sei $v|_I = \sum a_{ij} \frac{\partial}{\partial x_{ij}}$. Wir schreiben dies kurz als $v = A$, wobei A die Matrix mit den Einträgen a_{ij} ist.

$$\begin{aligned} v|_B =& dR_B(v|_I) \\ =& \sum a_{ik} b_{kj} \frac{\partial}{\partial x_{ij}} \end{aligned}$$

Schreibt man die Tangentenvektoren als Matrizen, so ergibt sich einfach

$$v_A|_B = v_A|_I \, B.$$

Die Lie-Klammer ergibt sich zu

$$\begin{aligned} [v_A, v_B] =& \sum_{i,j} (v_A(\sum_k b_{ik} x_{kj}) - v_B(\sum_k a_{ik} x_{kj})) \partial x_{ij} \\ =& \sum_{i,j} (\sum_{k,l} b_{ik} a_{kl} x_{lj} - \sum_{k,l} a_{ik} b_{kl} x_{lj}) \partial x_{ij} \\ =& v_{BA-AB} =: v_{-[A,B]} \end{aligned}$$

wobei man $[A, B] := AB - BA$ auch als Matrixkommutator bezeichnet. Die Lie-Algebra von $GL(n)$ läßt sich also mit GL(n) identifizieren, wobei die Lie-Klammer der negative Matrix-Kommutator ist.

Beispiel 1.3.3. *Betrachten wir anstelle dessen linksinvariante Vektorfelder, so wird die Lie-Klammer direkt zum Matrix-Kommutator.*

Wie wir später sehen werden, führen rechtsinvariante Vektorfelder auf die Differentialgleichung $X' = AX$ (siehe (1.30)), während linksinvariante zur Differentialgleichung $X' = XA$ führen.

Beispiel 1.3.4. *Wir betrachten den Torus T^2. Die rechtsinvarianten Vektorfelder sind von der Form $\mu\frac{\partial}{\partial\theta} + \nu\frac{\partial}{\partial\rho}$. Die Lie-Klammer $[\frac{\partial}{\partial\theta}, \frac{\partial}{\partial\rho}]$ verschwindet. Ist $\mu = 0$ oder ν/μ rational, so ist die erzeugte Untergruppe isomorph zu $SO(2)$. Ist aber ν/μ irrational, dann ist die erzeugte Untergruppe isomorph zu $\mathbb{R}$.*

1.3.3 Die Exponentialabbildung

Wir interpretieren die Lie-Algebra wieder als rechtsinvariante Vektorfelder. Jedes solche Vektorfeld $v \in \mathfrak{g}$ erzeugt einen Fluß Ψ_ε.

Definition 1.3.5. *Die Exponentialabbildung ist gegeben durch*

$$\begin{aligned} \exp : \mathfrak{g} &\to G \\ v &\mapsto \exp(v) = \Psi_1(e) \end{aligned} \tag{1.27}$$

Das Differential der exponentiellen Abbildung in 0 ist die Identität:

$$\begin{aligned} d\exp|_0 : T\mathfrak{g}|_0 \cong \mathfrak{g} &\to TG|_e \cong \mathfrak{g} \\ v &\mapsto d\exp|_0(v) = v \end{aligned} \tag{1.28}$$

Die Exponentialabbildung ist nach dem Satz über implizite Funktionen in einer Umgebung der 0 ein Diffeomorphismus. Global ist die Exponentialabbildung im allgemeinen weder injektiv noch surjektiv. Allerdings läßt sich jedes Element einer zusammenhängenden Lie-Gruppe als endliches Produkt

$$\exp(v_1)\exp(v_2)\ldots\exp(v_k)$$

schreiben.

1.3.4 Matrixformulierung

Einen leichter verständlichen Zugang zu Lie-Gruppen findet man, wenn man diese generell als Matrix-Gruppen betrachtet. Grundlage ist Ados Theorem [76, 105]

Satz 1.3.1. *Sei $\mathfrak{g}$ eine endlichdimensionale Lie-Algebra. Dann ist $\mathfrak{g}$ isomorph zu einer Unteralgebra von $\mathfrak{gl}(n)$ für ein n.*

Sei also G eine Lie-Gruppe mit Lie-Algebra $\mathfrak{g}$, die isomorph zu $\mathfrak{h} \subset \mathfrak{gl}(n)$ ist. Die Lie-Algebra $\mathfrak{h}$ generiert durch die exponentielle Abbildung eine Untermenge von $GL(n)$, deren Abschluß bezüglich der Multiplikation die Untergruppe $H \subset GL(n)$ mit eben der Lie-Algebra $\mathfrak{h}$ ergibt. Über das Diagramm

$$\begin{array}{ccc} \mathfrak{h} & \longrightarrow & \mathfrak{g} \\ {\scriptstyle \exp}\downarrow & & \downarrow \\ H & \longrightarrow & G \end{array} \tag{1.29}$$

ergibt sich dann ein eindeutig bestimmter Isomorphismus von H auf die Zusammenhangskomponente von G, die die Identität enthält. Wir können somit diese Komponente mit einer Matrix-Gruppe identifizieren.

Dieses Resultat besitzt im übrigen ein interessantes Analogon in der Gruppentheorie, nämlich daß jede endliche Gruppe isomorph zu einer Untergruppe einer Permutationsgruppe S_n ist. Permutationen sind ja gerade invertierbare Abbildungen von endlichen Mengen auf sich, während Matrix-Gruppen gerade aus den invertierbaren linearen Abbildungen von Vektorräumen auf sich bestehen.

In der zusammenhängenden Matrix-Gruppe $GL(n)$ mit Matrizen $X = (x_{ij})$ können wir $\frac{\partial}{\partial x_{ij}}$ als Basis des Tangentenvektorraumes verwenden. Die rechtsinvarianten und die linksinvarianten Vektorfelder hatten wir bereits in den Beispielen 1.3.2, 1.3.3 beschrieben. Die Lie-Algebra $\mathfrak{gl}(n)$ ist die Menge aller $(n \times n)$-Matrizen.

Der Fluß eines rechtsinvarianten Vektorfeldes $A|_X = AX$ ist die Lösung der linearen Differentialgleichung

$$X' = AX, \quad X(0) = I. \tag{1.30}$$

Die Exponentialabbildung $A \mapsto \exp(A) := X(1)$ ist damit die Matrix-Exponentialfunktion, gegeben durch die global konvergente Reihe

$$\exp(A) = \sum_{i=0}^{\infty} \frac{A^i}{i!} \tag{1.31}$$

Für linksinvariante Vektorfelder ergibt sich ebenso die Matrix-Exponentialfunktion als Exponentialabbildung.

Als Lie-Klammer wird der Matrix-Kommutator verwendet:

$$[A, B] = AB - BA. \tag{1.32}$$

Dies entspricht gerade der Lie-Klammer bei einer Interpretation von $\mathfrak{g}$ als Menge linksinvarianter Vektorfelder.

1.3.5 Die Baker-Campbell-Hausdorff-Formel

Die exponentielle Abbildung parametrisiert die Lie-Gruppe in einer Umgebung der Identität. Die Baker-Campbell-Hausdorff-Formel (BCH-Formel) gibt an, wie sich die Multiplikation in der Lie-Gruppe auf den Parameter auswirkt. Sie resultiert aus [50, 16, 5]. Sind $X, Y \in \mathfrak{g}$ hinreichend klein, so liegt das Produkt der Matrixexponentialfunktionen $\exp(X)\exp(Y)$ hinreichend nahe bei der Identität und damit in der Bildmenge der Exponentialabbildung, es kann daher wieder als Matrixexponentialfunktion geschrieben werden:

$$\exp(X)\exp(Y) = \exp(Z) \tag{1.33}$$

mit $Z \in \mathfrak{g}$. Dabei ergibt sich Z als Linearkombination von X, Y und den von X, Y generierten Kommutatoren, man spricht von der von X, Y erzeugten freien Lie-Algebra:

$$\begin{aligned} Z =& X + Y + \frac{1}{2}[X,Y] + \frac{1}{12}\left([X,X,Y] + [Y,Y,X]\right) \\ &- \frac{1}{24}[X,Y,X,Y] + \frac{1}{120}\left([X,Y,X,Y,X] + [Y,X,Y,X,Y]\right) \\ &- \frac{1}{720}\left([X,X,X,X,Y] + [Y,Y,Y,Y,X]\right) \\ &+ \frac{1}{360}\left([X,Y,Y,Y,X] + [Y,X,X,X,Y]\right) + - \dots \end{aligned} \tag{1.34}$$

Die Klammer-Ausdrücke bezeichnen iterierte Kommutatoren

$$[A_1, A_2, A_3] := [A_1, [A_2, A_3]]$$
$$[A_1, A_2, A_3, A_4] := [A_1, [A_2, [A_3, A_4]]] \text{ usw.}$$

wobei wir die einfache Klammer als Kommutator erster Stufe bezeichnen, und allgemein eine Klammer mit n Termen als Kommutator $(n-1)$-ter Stufe.

1.3.6 Transformationsgruppen

Infinitesimale Wirkungen

Wir betrachten eine Lie-Gruppe G von Transformationen einer Mannigfaltigkeit M. Es seien

$$\Psi : G \times M \to M$$

die Transformationen, wobei $\Psi(g, \cdot)$ einen Diffeomorphismus auf M darstellt.

Jedem Element v der Lie-Algebra $\mathfrak{g}$ von G läßt sich nun eine 1-parametrige Lie-Untergruppe $\exp(\varepsilon v)e$ von G zuordnen. Diese erzeugt einen Fluß auf M, der wiederum durch ein Vektorfeld $\psi(v)$ erzeugt wird. Es ist

$$\psi(v)|_x = \left.\frac{d}{d\varepsilon}\right|_0 \exp(\varepsilon v)(x).$$

Dies läßt sich auch anders schreiben: $\exp(\varepsilon v)(x)$ ist eine Kurve in M. Die Ableitung nach ε an der Stelle $\varepsilon = 0$ definiert den Tangentenvektor. Die Kurve selbst ergibt sich als Bild der Kurve $\exp(\varepsilon v)$ in G bei Anwendung von $\Psi_x(g) = \Psi(g, x)$. Der Tangentenvektor $\psi(v)$ ist daher

$$\psi(v) = d\Psi_x\left(\left.\frac{d}{d\varepsilon}\right|_0 \exp(\varepsilon v)\right) = d\Psi_x(v|_e).$$

Weiter ist

$$\Psi_x \circ R_g(h) = \Psi(hg, x) = \Psi(h, g \cdot x) = \Psi_{g\cdot x}(h)$$

und damit

$$d\Psi_x(v|_g) = d\Psi_x \circ dR_g(v|_e) = d\Psi_{g\cdot x}(v|_e) = \psi(v)|_{g\cdot x}$$

Satz 1.3.2. *ψ ist ein Lie-Algebra-Homomorphismus von der Lie-Algebra $\mathfrak{g}$ in die Lie-Algebra der Vektorfelder auf M.*

Beweis. Da $\psi(v)|_g = d\Psi_x(v|_g)$, ist die Linearität gesichert. Ebenso folgt aus den Eigenschaften des Differentials, daß $[\psi(v), \psi(w)] = \psi([v, w])$. □

Die Vektorfelder $\psi(v)$ bilden also eine Lie-Algebra, die isomorph zur Lie-Algebra $\mathfrak{g}$ ist. Sei umgekehrt eine endlich-dimensionale Lie-Algebra von Vektorfeldern auf M gegeben, so existiert eine lokale Transformationsgruppe, die diese Vektorfelder auf M erzeugt.

Satz 1.3.3. *Seien $w_1, \ldots, w_r$ Vektorfelder auf M in Involution mit den Strukturkonstanten c_{ij}:*

$$[v_i, v_j] = \sum_k c_{ij}^k v_k.$$

Dann gibt es eine Lie-Gruppe G lokaler Transformationen von M mit einer Lie-Algebra $\mathfrak{g}$, die die Strukturkonstanten c_{ij}^k bezüglich einer Basis $v_1, \ldots, v_k$ besitzt.

Definition 1.3.6. *Die Vektorfelder $v_1, \ldots, v_k$ werden infinitesimale Generatoren der entsprechenden Transformationsgruppe genannt.*

Man identifiziert die w_i (Vektorfelder auf M) mit den v_i (Lie-Algebra der Transformationsgruppe), da beide Lie-Algebren isomorph sind. Im allgemeinen steht man vor folgender Situation: Man hat auf der Mannigfaltigkeit M die Vektorfelder $v_1, \ldots, v_r$ gegeben. Jedes dieser Vektorfelder generiert eine Transformationsgruppe. Die Lie-Algebra generiert eine Lie-Gruppe von Transformationen. Man erhält eine Umgebung des Eins-Elementes dieser Lie-Gruppe, indem man jedem Element der Lie-Algebra durch die exponentielle Abbildung eine Transformation zuordnet.

Die Theorie der Transformationsgruppen von Sophus Lie

Wir wollen hier kurz auf die ursprünglichen Motive zur Entstehung der Lie-Gruppen eingehen. Sophus Lie hat in seinem Hauptwerk ([64], gemeinsam mit Friedrich Engel) Transformationsgruppen untersucht. Hierzu lohnt nun ein kurzer Blick auf die Galois-Theorie in der Algebra: Um die Lösungen algebraischer (polynomialer) Gleichungen zu beschreiben, betrachtet man Symmetriegruppen, also solche Transformationen, die die Lösungen invariant lassen. Die Struktur dieser Symmetriegruppen läßt nun Rückschlüsse auf die Struktur der Lösungen, ja auf die Lösung selbst zu. Motiviert von den Erfolgen dieser Theorie, hat Sophus Lie versucht, sie auf gewöhnliche und partielle Differentialgleichungen zu übertragen, indem er die Symmetriegruppen von Differentialgleichungen untersucht hat.

Leider war dieser Idee nur teilweise Erfolg beschieden. Für einige wenige Gleichungen kann man tatsächlich Lösungen ermitteln oder wesentliche Vereinfachungen durchführen. So kann man große Klassen gewöhnlicher Differentialgleichungen lösen, man vergleiche [76]. Auch läßt sich die Hopf-Cole Transformation für Burgers Gleichung herleiten.

Die Grundidee zur Lösung gewöhnlicher Differentialgleichungen besteht in der Bestimmung einer geeigneten Variablentransformation, die die Differentialgleichung $y' = f(t, y)$ in ein Quadraturproblem überführt. Die neuen Variablen τ, η werden dabei so bestimmt, daß die rechte Seite der transformierten Differentialgleichung $\eta' = \phi(\tau, \eta)$ nicht mehr von der Variablen η abhängt – also eine reine Integration darstellt.

Dazu bestimmt man eine Symmetriegruppe mit einem Ansatz durch infinitesimale Generatoren und betrachtet den Gruppenparameter als neue abhängige Variable η – in den transformierten Koordinaten ist dann die Verschiebung in η-Richtung eine Symmetriegruppe. Dies ist nur möglich, wenn die rechte Seite nicht von η abhängt – es liegt somit ein Quadraturproblem vor. Verbleibt die Bestimmung der Symmetriegruppe – dies ist mitunter sehr aufwendig und führt auf die Prolongation von Vektorfeldern in den Jet-Raum, man siehe [76].

1.3.7 Funktionen in speziellen Lie-Gruppen

Für Matrizen kleiner Dimension lassen sich Potenzreihen, zumeist unter Ausnutzung des Satzes von Cayley-Hamilton, explizit auswerten. Die Resultate dieses Abschnitts sind insofern klassisch, man findet sie beispielsweise in [58]. Wir verweisen außerdem noch kurz auf [21], wo weitere Approximationstechniken für die Exponentialabbildung untersucht werden, sowie auf den Klassiker [71].

Die Exponentialabbildung in $\mathfrak{so}(3)$

Für schiefsymmetrische Matrizen der Dimension 3 läßt sich die Matrixexponentialfunktion unter Anwendung des Satzes von Cayley-Hamilton explizit berechnen. Sei

$$A = \begin{pmatrix} 0 & a & -c \\ -a & 0 & b \\ c & -b & 0 \end{pmatrix} \tag{1.35}$$

eine schiefsymmetrische Matrix. Die charakteristische Gleichung ergibt sich zu

$$\lambda^3 + (a^2 + b^2 + c^2)\lambda = 0. \tag{1.36}$$

Da A selbst der charakteristischen Gleichung genügt, kann man für $k \geq 0$

$$A^{2k+1} = (-s^2)^k A, \qquad A^{2k+2} = (-s^2)^k A^2 \tag{1.37}$$

setzen, wobei

$$s(A) = \sqrt{a^2 + b^2 + c^2} = \|A\|_F / \sqrt{2}. \tag{1.38}$$

Die Matrixexponentialfunktion ergibt sich damit für $s \neq 0$ zu

$$\exp(A) = I + \sum_{k=0}^{\infty} \frac{A^{2k+1}}{(2k+1)!} + \sum_{k=0}^{\infty} \frac{A^{2k+2}}{(2k+2)!} \tag{1.39}$$

$$= I + \frac{1}{is} \sum_{k=0}^{\infty} \frac{(is)^{2k+1}}{(2k+1)!} A + \frac{1}{-s^2} \sum_{k=0}^{\infty} \frac{(is)^{2k+2}}{(2k+2)!} A^2 \tag{1.40}$$

$$= I + \frac{\sin(s)}{s} A - \frac{\cos(s) - 1}{s^2} A^2. \tag{1.41}$$

Die letzte Zeile ist als Formel von Rodriguez bekannt.

Der Logarithmus in $SO(3)$

Mit der Formel von Rodriguez läßt sich auch leicht die Umkehrung der Exponentialfunktion, der Logarithmus, berechnen. Sei im folgenden $B = \exp(A)$. Man folgert leicht, daß A^2 symmetrisch ist, wenn A schiefsymmetrisch ist:

$$(A^2)^T = A^T A^T = (-A)^2 = A^2. \tag{1.42}$$

Bis auf einen skalaren Faktor ergibt sich A aus dem schiefsymmetrischen Anteil von B, genauer

$$\frac{\sin(s(A))}{s(A)} A = 1/2(B - B^T) =: C \Rightarrow \tag{1.43}$$

$$\sin(s(A)) = s(C) \Rightarrow \tag{1.44}$$

$$A = \frac{\sin^{-1}(s(C))}{s(C)} C. \tag{1.45}$$

Die BCH-Formel in $\mathfrak{so}(3)$

Mit der Exponentialfunktion und der Logarithmusfunktion läßt sich auch die BCH-Formel explizit angeben. Nach [33] ergibt sich die BCH-Formel in der Form

$$\exp(Z) = \exp(X)\exp(Y) \tag{1.46}$$
$$Z = \alpha X + \beta Y + \gamma[X,Y]. \tag{1.47}$$

In [33] wird auch auf folgende Interpretation hingewiesen: $\exp(X), \exp(Y)$ stellen Drehungen im Raum dar, $\exp(Z)$ repräsentiert die Hintereinanderausführung zweier Drehungen als einzelne Drehung. Die skalaren Größen α, β, γ hängen gerade von den beiden Drehwinkeln $s(X), s(Y)$ und dem Winkel zwischen den beiden Drehachsen ab.

Bemerkung 1.3.1. *Auch die Abbildungen* $\exp$ *und* $\exp^{-1}$ *lassen sich in* $\mathfrak{so}(3)$ *effizient berechnen. Wir verweisen hier auf die Literatur, man siehe [58].*

Die Exponentialabbildung in $\mathfrak{gl}(2)$

Hier können wir ähnlich wie bei der Herleitung der Formel von Rodriguez vorgehen. Wir symmetrisieren das Spektrum der Matrix $A \in \mathbb{R}^{2\times 2}$ durch $A \mapsto B = A - rI$ mit $r = \operatorname{spur} A$. Die Exponentialabbildung ergibt sich damit zu

$$\exp(A) = e^r \exp(B). \tag{1.48}$$

Mit $s = \det B$ ergibt sich nach dem Satz von Cayley-Hamilton

$$B^2 = -sI \Rightarrow \tag{1.49}$$
$$B^{2k+1} = (-s)^k B \tag{1.50}$$
$$B^{2k} = (-s)^k I. \tag{1.51}$$

In die Reihenentwicklung der Exponentialabbildung eingesetzt ergibt sich

$$\begin{aligned}\exp(B) &= \sum_{k=0}^{\infty} \frac{(-s)^k}{(2k)!} I + \sum_{k=0}^{\infty} \frac{(-s)^{2k}}{(2k+1)!} B \\ &= \sum_{k=0}^{\infty} \frac{\sqrt{-s}^{2k}}{(2k)!} I + \frac{1}{\sqrt{-s}} \sum_{k=0}^{\infty} \frac{\sqrt{-s}^{2k+1}}{(2k+1)!} B \\ &= \cosh(\sqrt{-s}) I + \frac{\sinh(\sqrt{-s})}{\sqrt{-s}} B.\end{aligned} \tag{1.52}$$

Die Formel gilt unabhängig vom Vorzeichen der Determinante s, alternativ (für positive Determinanten) kann $\cosh(ix) = \cos x$ und $\sinh(ix) = \sin x$ eingesetzt werden.

1.4 Zusammenfassung

Wir haben in diesem Kapitel die mathematischen Grundlagen gelegt, um uns in den folgenden Kapiteln von Teil I näher mit der numerischen Lösung von Differentialgleichungen auf Lie-Gruppen zu beschäftigen. Zu diesen Grundlagen gehört die Analysis auf Mannigfaltigkeiten mit den Begriffsbildungen Tangentenvektor, Vektorfeld, Integralkurve und Fluß eines Vektorfelds, um so zur Abstraktion und

Verallgemeinerung gewöhnlicher Differentialgleichungen zu gelangen. Weiter folgen Lie-Gruppen als spezielle differenzierbare Mannigfaltigkeiten. Tangentenvektorräume und Vektorfelder führen in diesem Fall auf den Begriff der Lie-Algebra. Dabei sind dann insbesondere die Lie-Klammer, die Exponential-Abbildung sowie die BCH-Formel von Bedeutung.

Kapitel 2

Runge-Kutta-Verfahren in Lie-Gruppen – die RK-MK-Klasse

2.1 Übertragung des Grundverfahrens

2.1.1 Der Lie-Gruppen-Ansatz für Differentialgleichungen auf Mannigfaltigkeiten

Sei M eine m-dimensionale differenzierbare Mannigfaltigkeit. Durch

$$y'(t) = f(t, y(t)), \quad y(0) = y_0 \in M, \quad f : \mathbb{R} \times M \to TM, \quad f(t, y) \in TM|_y \tag{2.1}$$

ist eine Differentialgleichung auf dieser Mannigfaltigkeit gegeben, die Lösung ist für hinreichend glattes f eine glatte Kurve $y(t)$.

Einen allgemeinen Ansatz zur numerischen Lösung von Differentialgleichungen auf Mannigfaltigkeiten bieten die Crouch-Grossman-Methoden [22]. Man geht von einer endlichdimensionalen Basisdarstellung des Tangentialraums durch geeignete Vektorfelder v_i aus, deren Fluß bestimmt werden kann. Die rechte Seite wird in diesen Basis-Vektorfeldern dargestellt, also

$$y' = f(t, y) = \sum_{i=1}^{m} f_i(t, y)\ v_i|_y\,. \tag{2.2}$$

Ein aus dem expliziten Euler-Verfahren abgeleitetes Verfahren erster Ordnung ergäbe sich zum Beispiel durch

$$y_{n+1} = \exp(h f_1(t_n, y_n) v_1) \ldots \exp(h f_m(t_n, y_n) v_m) y_n. \tag{2.3}$$

Bemerkung 2.1.1. *Das abstrakte Konzept von Differentialgleichungen auf Mannigfaltigkeiten findet auch Anwendung bei der Untersuchung differential-algebraischer Systeme. Diese können eben als Differentialgleichungen auf Mannigfaltigkeiten interpretiert werden, man vergleiche [81].*

Der Lie-Gruppen-Ansatz nach [73], [74] setzt das Vorhandensein einer m-dimensionalen Lie-Gruppe G als Transformationsgruppe (vergleiche Abschnitt 1.3.6) der Mannigfaltigkeit M voraus. Ausgehend von einer Approximation y_n an die Lösung in $t = t_n$, berechnet man dann weitere Approximationen wie folgt: durch die Transformationsgruppe G stellen wir Elemente der Mannigfaltigkeit in einer Umgebung von y_n durch

$$y(t) = \Psi(g(t), y_n), \quad g(t) \in G \tag{2.4}$$

dar. Ist das Differential $d\Psi(\cdot, y_n)$ invertierbar, so erhalten wir eine Differentialgleichung für $g(t) \in G$

$$g'(t) = d\Psi(\cdot, y_n)^{-1}(f(t, \Psi(g(t), y_n))). \tag{2.5}$$

Für fixiertes zweites Argument bildet Ψ von G nach M ab, also bildet $d\Psi(\cdot, y_n)^{-1}$ von $TM|_{y(t)}$ nach $TG|_{g(t)}$ ab. Wir identifizieren den Tangentialraum $TG|_{g(t)}$ mit der Menge der rechtsinvarianten Vektorfelder – der Lie-Algebra – indem wir einem Tangentenvektor das rechtsinvariante Vektorfeld v zuordnen, das in $g(t)$ mit ihm übereinstimmt. Wir bilden also $\mathfrak{g} = TG|_e$ vermöge $dR_{g(t)}$ eineindeutig nach $TG|_{g(t)}$ ab.

Wir gelangen damit zu einer Differentialgleichung in der Lie-Gruppe G

$$g'(t) = v(t, g(t))|_{g(t)}\,. \tag{2.6}$$

mit einer rechten Seite der Form $v : \mathbb{R} \times G \to \mathfrak{g}$. Wir sind diesem Ansatz bereits im Abschnitt 1.1 über die Toda-Lattice-Gleichung begegnet.

2.1.2 Übertragung der Verfahrensvorschrift klassischer Runge-Kutta-Verfahren

Ausgangspunkt sind klassische[1] Runge-Kutta-Verfahren zur Lösung von Anfangswertproblemen gewöhnlicher Differentialgleichungen

$$y' = f(t, y), \quad y(t_0) = y_0, \quad t \in [t_0, t_e], \tag{2.7}$$

die auf Lie-Gruppen verallgemeinert werden sollen. Üblich sind Ansätze mit internen Stufen Y_{ni} oder mit internen Ableitungen k_{ni}. Wir formulieren die Runge-Kutta-Methode unter Einbeziehung sowohl von internen Stufen als auch internen Ableitungen:

$$\begin{aligned} k_{ni} &= f(t_n + c_i h, Y_{ni}) \\ Y_{ni} &= y_n + h \sum_{j=1}^{s} a_{ij} k_{nj}, \quad i = 1, \ldots, s \\ y_{n+1} &= y_n + h \sum_{i=1}^{s} b_i k_{ni}. \end{aligned} \tag{2.8}$$

Eine Übertragung in den Lie-Gruppen- bzw. Mannigfaltigkeiten-Kontext geschieht dann wie folgt. Anstelle der gewöhnlichen Differentialgleichung (2.7) betrachten wir ein Anfangswertproblem für eine Differentialgleichung auf einer Lie-Gruppe G

$$y' = v(t, y)|_y\,, \quad v : \mathbb{R} \times G \to \mathfrak{g}, \quad y(t_0) = y_0 \in G, \quad t \in [t_0, t_e]. \tag{2.9}$$

Gesucht ist die Kurve $y(t)$ auf der Lie-Gruppe (Mannigfaltigkeit) G. Die Ableitung liegt im Tangentialraum $TG|_y$ an G im Punkt y. Mit der Lie-Algebra $\mathfrak{g}$ ist der Tangentialraum an die Lie-Gruppe im neutralen Element gegeben. Wir interpretieren die Lie-Algebra hier als Menge rechtsinvarianter Vektorfelder, wenn also $v \in \mathfrak{g} = TG|_e$, dann setzen wir $v|_y = dR_y(v)$. Das Differential der Rechtsmultiplikation bildet dabei von TG_e nach TG_y ab, die rechte Seite in (2.9) liegt also im zugehörigen Tangentialraum.

[1] nicht zu verwechseln mit **dem** klassischen Runge-Kutta-Verfahren

Zur Übertragung der Verfahrensvorschrift interpretieren wir den $\mathbb{R}^n$ als affinen Raum. Wir unterscheiden zwischen festen Punkten (Elemente des affinen Raumes) und Translationen, die Punkte in Punkte abbilden.

Die festen Punkte betrachten wir als Lie-Gruppe mit der Addition als Gruppenoperation. Die Lie-Algebra dieser Gruppe ist der Tangentialraum an $x = 0$, und dieser ist isomorph zum $\mathbb{R}^n$. Die rechtsinvarianten Vektorfelder sind gerade die konstanten Vektorfelder, und der Fluß eines rechtsinvarianten Vektorfeldes ist eine Translation um den erzeugenden Vektor. Die Translationen ergeben sich somit gleichzeitig aus der Lie-Algebra durch Anwendung der Exponentialabbildung.

Wir interpretieren die Größen im Runge-Kutta-Verfahren also wie folgt: Die Approximationen y_n, y_{n+1} und die internen Werte Y_{ni} gehören der Lie-Gruppe an. Die internen Ableitungen $k_{ni} = f$ gehören zur Lie-Algebra, ebenso die Linearkombinationen $v_i = h\sum_j a_{ij}k_{nj}$. Die Aufdatierungen $y_n + v_i$ interpretieren wir als Translation von y_n, die durch das konstante Vektorfeld v_i erzeugt wird. In Koordinatendarstellung ist hier die Exponentialabbildung die Identität: Der Vektor $v \in \mathfrak{g}$ wird gleichzeitig als Element der Lie-Gruppe interpretiert, die zugehörige Translation um diesen Vektor ist die Rechtsmultiplikation R_v.

Ersetzen wir f durch v und fügen den Fluß des Vektorfeldes v_i ein, so ergibt sich das verallgemeinerte Runge-Kutta-Verfahren wie folgt:

$$\begin{aligned} k_{ni} &= v(t_n + c_i h, Y_{ni}) \in \mathfrak{g} \\ Y_{ni} &= \exp(h\sum_{j=1}^{s} a_{ij}k_{nj}) \cdot y_n, \quad i = 1, \ldots, s \\ y_{n+1} &= \exp(h\sum_{i=1}^{s} b_i k_{ni}) \cdot y_n. \end{aligned} \tag{2.10}$$

Dabei generiert $\exp : \mathfrak{g} \to G$ den Fluß des entsprechenden rechtsinvarianten Vektorfelds, da $\exp(v) \cdot y = \Psi_{v,1}(y)$. Die internen Stufen Y_{ni} sind explizit berechenbar, wenn $a_{ij} = 0$ für $i < j$, man spricht dann von expliziten Verfahren und macht dies für gewöhnlich in der Summation für Y_{ni} durch einen oberen Laufindex $i - 1$ kenntlich. Wir betrachten hier im wesentlichen auch nur explizite Verfahren, wobei die Resultate alle auch für implizite Verfahren gelten, allerdings ist in diesem Fall die Lösung der Verfahrensgleichungen nur iterativ möglich.

Im Vergleich zu den Methoden von Crouch und Großmann [22] stellt man fest, daß hier der Fluß für den kompletten Vektorraum bekannt (berechenbar) ist, während in [22] nur die Flüsse der Basis-Vektorfelder als bekannt vorausgesetzt werden. Daher wird dort eben auch keine Linearkombination $v_i = h\sum_j a_{ij}k_{nj}$ gebildet, sondern die Flüsse der Basis-Vektorfelder werden nacheinander angewandt.

2.1.3 Diskretisierungsfehler und Ordnungsbegriffe

Wir übertragen die klassischen Begriffe wie lokaler Fehler, globaler Fehler, Konsistenz- und Konvergenzordnung, siehe auch [116].

Sei der exakte Fluß der Differentialgleichung (2.9) durch

$$y(t + \tau) = \Psi_{t,\tau}(y(t)) \tag{2.11}$$

gegeben. Die Lösung werde schrittweise numerisch durch ein Einschrittverfahren

$$y_{n+1} = \exp(h\Phi(t_n, y_n, h))y_n \tag{2.12}$$

approximiert. Dabei heißt $\Phi : \mathbb{R} \times G \times \mathbb{R} \to \mathfrak{g}$ die Inkrementfunktion des Verfahrens.

Im Fall gewöhnlicher Differentialgleichungen (ODE-Fall) wird der lokale Fehler als Differenz aus numerischer und exakter Lösung nach einem Integrationsschritt definiert. Hier ist diese Definition nicht direkt übertragbar, da in der Lie-Gruppe keine Differenzen gebildet werden können. Wir benötigen eine Metrik in der Lie-Gruppe, wir müssen Differenzen messen können. Dies kann in lokalen Koordinaten getan werden, natürlicher erscheint es jedoch, die Lie-Algebra mit einzubeziehen und die exponentielle Abbildung zur Parametrisierung zu nutzen. Diese ist zwar nicht global umkehrbar, aber zumindest in einer Umgebung des Einselements. Dies genügt uns, da wir die auftretenden Fehler als hinreichend klein voraussetzen.

Definition 2.1.1. *Der lokale Fehler $le_h(t)$ ist gegeben durch*

$$y_{n+1} = \exp(le_h(t))y(t+h), \tag{2.13}$$

wobei y_{n+1} das Resultat eines Verfahrensschrittes (2.12) *mit Startwerten $t_n = t$, $y_n = y(t_n)$ auf der exakten Lösung ist.*

Definition 2.1.2. *Das Verfahren heißt konsistent von der Ordnung p (Konsistenzordnung p) für ein Anfangswertproblem, falls*

$$le_h(t) = \mathcal{O}\left(h^{p+1}\right). \tag{2.14}$$

Wir betrachten hier den globalen Fehler nur für konstante Schrittweiten. Eine Übertragung der Ergebnisse auf variable Schrittweiten ist ohne Schwierigkeiten möglich.

Definition 2.1.3. *Der globale Fehler $e_h(t)$ ist gegeben als*

$$y_n = \exp(e_h(t_n))y(t_n), \tag{2.15}$$

wobei y_n das Resultat von n Verfahrensschritten, beginnend im Anfangswert $y_0 = y(t_0)$, ist.

Definition 2.1.4. *Das Verfahren heißt konvergent von der Ordnung p (Konvergenzordnung p) für ein Anfangswertproblem, wenn für $t \in [t_0, t_e]$*

$$e_h(t) = \mathcal{O}\left(h^p\right) \tag{2.16}$$

gilt.

Definition 2.1.5. *Eine Einschrittmethode heißt stabil, wenn Konstanten $C, C', h > 0$ existieren, so daß für $h \leq h_0$ und Anfangswerte $y, \tilde{y}$ mit*

$$\tilde{y} = \exp(k)y, \quad k \in \mathfrak{g}, \|k\| \leq C'h \tag{2.17}$$

ein $k^* \in \mathfrak{g}$ existiert, so daß gilt

$$\begin{aligned} \exp(h\Phi(t,\tilde{y},h))\tilde{y} &= \exp(k^*)\exp(h\Phi(t,y,h))y \\ \|k^*\| &\leq (1+Ch)\|k\|. \end{aligned} \tag{2.18}$$

Stabilität ist für die meisten Einschrittverfahren gesichert. Sie folgt insbesondere aus der Konsistenz des Verfahrens, wenn die rechte Seite der Differentialgleichung Lipschitz-stetig ist.

Sei der Fluß der Differentialgleichung durch $y(t+h) = \exp(h\nu(t, y(t), h))y(t)$ und der führende Term im lokalen Fehler $d_{p+1}(t)h^{p+1}$, dann genügt die Inkrementfunktion

$$h\Phi(t, y(t), h) = h\nu(t, y(t), h) + d_{p+1}h^{p+1} + \mathcal{O}\left(h^{p+2}\right) \tag{2.19}$$

Die klassische Konvergenzaussage für Einschrittverfahren läßt sich nun übertragen:

Satz 2.1.1. *Sei Φ die Inkrementfunktion einer stabilen Einschrittmethode. Genügt der lokale Fehler zur exakten Lösung $y(t)$ der Bedingung*

$$le_h(t) = \mathcal{O}\left(h^{p+1}\right), \tag{2.20}$$

dann erfüllt der globale Fehler

$$e_h(t) = \mathcal{O}\left(h^p\right). \tag{2.21}$$

Beweis. Zum Beweis ziehen wir die klassische Abschätztechnik[2] heran.

Zu jedem Punkt $y(t_k)$ konstruieren wir eine Folge numerischer Näherungen $y_{k+i}^{(k)}$, beginnend mit $y_k^{(k)} = y(t_k)$, wobei sich die weiteren Werte durch Anwendung des Einschrittverfahrens mit der Inkrementfunktion Φ ergeben

$$y_{k+i+1}^{(k)} = \exp(h\Phi(t_{k+i}, y_{k+i}^{(k)}, h))y_{k+i}^{(k)}, \quad i \geq 0. \tag{2.22}$$

Die Konsistenzordnung p liefert

$$y_k^{(k-1)} = \exp(\mathcal{O}\left(h^{p+1}\right))y_k^{(k)}. \tag{2.23}$$

Mit der Stabilität folgt

$$\begin{aligned} y_n^{(k-1)} &= \exp(\mathcal{O}\left(h^{p+1}\right))y_n^{(k)} \\ y_n &= \left(\prod_{k=1}^{n} \exp(\mathcal{O}\left(h^{p+1}\right))\right) y(t_n) \end{aligned} \tag{2.24}$$

Eine Anwendung der Baker-Campbell-Hausdorff-Formel (1.34) liefert schließlich die Behauptung. □

2.2 Ordnung des Grundverfahrens

2.2.1 Ordnungsbedingungen im ODE-Fall

Wir geben hier einen kurzen Abriß der Vorgehensweise bei klassischen Runge-Kutta-Verfahren, wobei wir uns auf den autonomen Fall beschränken. Man entwickelt exakte und numerische Lösung in Taylorreihen. Die Terme in den Entwicklungen repräsentiert man durch sogenannte Wurzelbäume (Butcher-Bäume, [12]) und gibt Rekursionen zu ihrer Generierung an. Aus der Differentiation der Differentialgleichung $y' = f(y)$ an der Stelle $t = t_n$ sowie der Differentiation der Stufengleichungen $Y_{ni} = y_n + h\sum_j a_{ij} f(Y_{nj})$ nach h in $h = 0$ ergibt sich

$$\begin{aligned} y^{(k)} &= (f(y))^{(k-1)} \\ Y_{ni}^{(k)} &= k\sum_j a_{ij}(f(Y_{nj}))^{(k-1)}. \end{aligned} \tag{2.25}$$

mit den Initialisierungen $y^{(1)} = f(y_n)$, $Y_{ni}^{(1)} = \sum_j a_{ij} f(y_n)$.

[2] die typischerweise verwandte Skizze wird auch als „Lady Windermeres Fächer“ bezeichnet

Mit Faà di Brunos Formel [34] verfügt man über eine Kettenregel für Ableitungen höherer Ordnung und kann die Ausdrücke $(f(y))^{(k)}$ entwickeln. Die ersten Ableitungen ergeben sich (durch manuelle Differentiation) zu

$$\begin{aligned}(f(y))^{(1)} =& f_y y^{(1)} \\ (f(y))^{(2)} =& f_{yy}(y^{(1)}, y^{(1)}) + f_y y^{(2)} \\ (f(y))^{(3)} =& f_{yyy}(y^{(1)}, y^{(1)}, y^{(1)}) + 3 f_{yy}(y^{(1)}, y^{(2)}) + f_y y^{(3)}.\end{aligned} \tag{2.26}$$

Für den allgemeinen Fall stellt man die auftretenden Ausdrücke durch sogenannte einfache indizierte Wurzelbäume dar. Diese besitzen höchstens einen Verzweigungsknoten — die Wurzel. Gehen von der Wurzel k Äste der Längen $\rho_1, \ldots, \rho_k$ aus, so ordnet man diesem Baum das Symbol $[\rho_1, \ldots, \rho_k]$ zu und schreibt

$$F([\rho_1, \ldots, \rho_k]) = \frac{\partial^k f}{\partial y^k}(y^{(\rho_1)}, \ldots, y^{(\rho_k)}) \tag{2.27}$$

Differenziert man einen solchen Ausdruck, so muß zum einen $f_{y\ldots y}$ differenziert werden (und ein zusätzliches $y^{(1)}$ eingefügt werden), zum anderen muß jedes $y^{(\rho_i)}$ differenziert werden. Der erste Schritt entspricht gerade dem Anhängen eines Knotens an die Wurzel, der letzte Schritt gerade dem Anhängen eines Knotens an jeweils einen Ast.

Starten wir also die Rekursion mit dem Baum $[\,]$, der gerade f darstellt, und versehen den Knoten mit dem Index 1. In jedem weiteren Differentiationsschritt muß an jeden der im letzten Schritt generierten Bäume an jeweils alle Endknoten bzw. an die Wurzel ein neuer Knoten gehängt werden. Indizieren wir nun die anzuhängenden Knoten fortlaufend, so erkennen wir, daß auf diese Weise gerade alle einfachen Wurzelbäume generiert werden, die monoton indiziert sind, d.h., die Indizierung wächst auf jedem Ast mit wachsendem Abstand zur Wurzel. Jeder Baum mit solcher Indizierung kann offensichtlich durch sukzessives Anhängen indizierter Knoten generiert werden.

Auch wird jeder dieser Bäume genau einmal generiert. Dies folgt leicht induktiv: Treten in $(f(y))^{(k)}$ alle monoton indizierten Bäume mit $k+1$ Knoten genau einmal auf, so ergibt sich $y^{(k+1)}$ durch Anhängen des Knotens mit Index $k+2$ an jeden Endknoten bzw. an die Wurzel jedes dieser Bäume. Wären 2 der entstehenden Bäume gleich, so müßte zum einen der Knoten $k+2$ am gleichen Knoten angehängt sein, und zum anderen müßten die durch Streichung des Knotens mit Index $k+2$ entstehenden Restbäume identisch sein – und das heißt, bereits in $(f(y))^{(k)}$ waren zwei identische Bäume vorhanden – was aber der Induktionsvoraussetzung für $k = 1$ widerspricht.

Wir bemerken, daß Vertauschungen in der Anordnung der Äste nicht zu berücksichtigen sind, zur Identifikation eines monoton indizierten Baumes ist die „Topologie“ der Indizes wesentlich, d.h., welcher Knoten folgt welchem, von der Wurzel aus gesehen.

Obige Überlegungen führen dann auf Faá die Brunos Formel —

$$(f(y))^{(k)} = \sum_{\tau \in SLT_{k+1}} F(\tau). \tag{2.28}$$

Dabei steht SLT_l für die vereinfachten indizierten Bäume (**s**implified **l**abelled **t**rees) mit l Knoten (einschließlich der Wurzel). Da $F(\tau)$ nicht von der Indizierung von τ abhängt, kann natürlich auch über die nicht-indizierten einfachen Bäume summiert werden, wobei mit der entsprechenden Anzahl indizierter Bäume gewichtet werden muß. Für unsere Zwecke ist diese Anzahl aber nicht von Bedeutung.

Nun können wir uns der Darstellung der eigentlichen Entwicklung zuwenden. Mit Hilfe der Rekursionen (2.25) können wir die exakte und die numerische Lösung komplett darstellen, wenn wir Faá

di Brunos Formel einsetzen. In den Ausdrücken $\frac{\partial^k f}{\partial y^k}(y^{(\rho_1)}, \ldots, y^{(\rho_k)})$ sind dabei die $y^{(\rho_i)}$ durch bereits berechnete Ausdrücke zu ersetzen, und genau diese Ersetzungen nehmen wir auch in den Bäumen vor, d.h., die Äste ohne Verzweigungen werden nach und nach durch Bäume ersetzt. Dies führt auf eine Darstellung der Entwicklungen durch indizierte Wurzelbäume. Diese werden rekursiv definiert:

Definition 2.2.1. *Die Menge der indizierten Wurzelbäume ist rekursiv gegeben als Menge T durch*

- $[\,] \in T$
- *mit $t_1, \ldots, t_k$ ist auch $[t_1, \ldots, t_k] \in T$.*

Dabei sind $[t_1, \ldots, t_i, \ldots, t_j, \ldots, t_k]$ und $[t_1, \ldots, t_j, \ldots, t_i, \ldots, t_k]$ äquivalent.

Bei der rekursiven Generierung ist $[t_1, \ldots, t_k]$ gerade der Baum, der durch Anhängen von $t_1, \ldots, t_k$ an eine gemeinsame Wurzel entsteht. Der letzte Satz in Definition 2.2.1 bedeutet gerade, daß die Reihenfolge der Äste keine Rolle spielt.

Zur Darstellung der Entwicklungen werden Funktionen auf diesen Bäumen benötigt:

Definition 2.2.2. *Die Funktionen ρ, γ, Φ_i, $F(\cdot)(t)$ sind auf T gegeben durch die Initialisierung*

$$\begin{aligned} \rho([\,]) &= 1 \\ \gamma([\,]) &= 1 \\ \Phi_i([\,]) &= \sum_j a_{ij} \\ F([\,])(t) &= f(y(t)) \end{aligned} \tag{2.29}$$

sowie die Rekursionen für $\tau = [t_1, \ldots, t_k]$

$$\begin{aligned} \rho(\tau) &= 1 + \sum_{i=1}^{k} t_i \\ \gamma(\tau) &= \rho(\tau) \prod_{i=1}^{k} \gamma(t_k) \\ \Phi_i(\tau) &= \sum_j a_{ij} \Phi_j(t_1) \cdot \cdots \cdot \Phi_j(t_k) \\ F(\tau)(t) &= \frac{\partial^n f}{\partial y^n}(F(t_1)(t), \ldots, F(t_k)(t)). \end{aligned} \tag{2.30}$$

ρ heißt Ordnung des Wurzelbaums. Außerdem ist Φ_i auch für $i = s + 1$ definiert durch $a_{s+1,j} := b_j$.

Definition 2.2.3. *Ein indizierter Wurzelbaum ist ein Wurzelbaum τ mit Knoten, die mit den Ziffern $1, \ldots, \rho(\tau)$ indiziert sind. Beim monoton indizierten Wurzelbaum hat jeder Knoten ("Sohn") einen höheren Index als sein Vaterknoten, wobei der Vaterknoten der der Wurzel am nächsten liegende benachbarte Knoten ist (die Wurzel besitzt als einziger Knoten keinen Vaterknoten). Die Wurzel besitzt den Index 1.*

Definition 2.2.4. *Für einen Baum $\tau \in T$ bezeichne $\alpha(\tau)$ die Anzahl möglicher monotoner Indizierungen. Weiter sei T_ρ bzw. LT_ρ die Menge der Wurzelbäume bzw. der monoton indizierten Wurzelbäume der Ordnung ρ.*

Mit diesen Begriffsbildungen lassen sich die Entwicklung der analytischen und der numerischen Lösung einfach angeben:

Satz 2.2.1. *Die analytische Lösung $y(t_n + h)$ der Differentialgleichung $y' = f(y)$ genügt an der Stelle (t_n, y_n) für hinreichend glattes f der Entwicklung*

$$y(t_n + h) = y(t_n) + \sum_{\tau \in T_\rho} \frac{h^\rho}{\rho!} \alpha(\tau) F(\tau)(y_n). \tag{2.31}$$

Satz 2.2.2. *Die numerische Lösung der Differentialgleichung $y' = f(y)$ genügt der Entwicklung*

$$y_{n+1} = \sum_{\tau \in T_\rho} \frac{h^\rho}{\rho!} \alpha(\tau) \gamma(\tau) \Phi_{s+1}(\tau) F(\tau)(y_n). \tag{2.32}$$

Detaillierte Beweise bleiben wir an dieser Stelle schuldig, sie finden sich zum Beispiel in [44], wir wollen die Beweisführung lediglich skizzieren. Die rekursiven Definitionen legen einen induktiven Beweis nahe. Wesentlicher Gedanke beim Induktionsschritt ist, in den Sätzen 2.2.1 und 2.2.2 die Gewichte $\alpha(\tau)$ zu streichen und statt dessen über alle indizierten Bäume zu summieren. In die Entwicklungen für $(f(y))^{(k-1)}$ bzw. $(f(Y_{ni}))^{(k-1)}$ werden die Summendarstellungen für Ableitungen geringerer Ordnung eingesetzt, und alles wird ausmultipliziert. Man erhält eine mehrfache Summation über alle einfachen monoton indizierten Bäume der Ordnung k ($\sum_{\tau=[i_1,\ldots,i_l]}$) und über alle indizierten Bäume der Ordnungen $i_1, \ldots, i_l$. Dies ist äquivalent einer Summation über alle monoton indizierten Bäume t der Ordnung k, die Zuordnung erfolgt dabei durch Abschneiden der Äste von t. Diese Äste ergeben die monoton indizierten Bäume der Ordnungen $i_1, \ldots, i_l$. Durch Komprimieren der Äste zu einem Ast ohne Verzweigung ergibt sich wiederum der einfache Wurzelbaum. Die Indizierung wird für den einfachen monoton indizierten Wurzelbaum übernommen, die Indizierung der abgetrennten Bäume wird so vorgenommen, daß Ordnungsrelationen erhalten bleiben. Das Beispiel in Abbildung 2.1 verdeutlicht die Vorgehensweise.

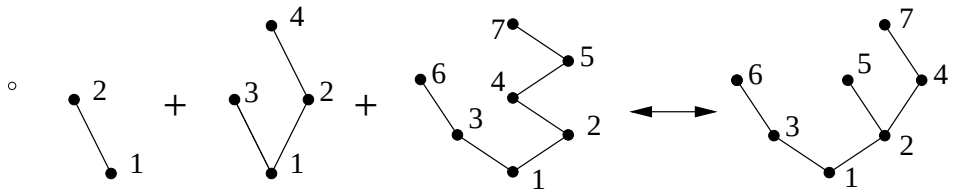


Abbildung 2.1: Zuordnung der Bäume im Beweis zu Satz 2.2.2.

2.2.2 Darstellung der exakten Lösung in der Lie-Gruppe

In analoger Weise kann man in Lie-Gruppen vorgehen, nur muß man an einigen Stellen Modifikationen vornehmen. In einer ersten Arbeit [72] wurden Runge-Kutta-Methoden auf kommutative Lie-Gruppen übertragen. Dies ist im wesentlichen ohne Änderungen möglich. Für nichtkommutative Lie-Gruppen führt die Erweiterung der Theorie gewöhnlicher Differentialgleichungen auf erhebliche Schwierigkeiten. In der Arbeit von Munthe-Kaas [73] wird ein erster Weg aufgezeigt, den wir hier in groben Zügen darstellen wollen. Ein eleganterer Ansatz findet sich in [74], worauf wir in Abschnitt 2.3.2 eingehen werden.

Wir betrachten jetzt also eine Differentialgleichung in einer Lie-Gruppe, die rechte Seite sei durch $v : G \mapsto \mathfrak{g}$ gegeben. Wir entwickeln dazu v in eine Reihe.

Zur Darstellung betrachten wir neben der rechten Seite $v \in \mathfrak{g}$ das zeitabhängige Vektorfeld $w(t) := v(y(t))$, wobei $y(t)$ die exakte Lösung sei. Das Vektorfeld w generiert einen Fluß $\Psi_{t,w}$ mit der Eigenschaft

$$\frac{d}{dt}\Psi_{t,w}(y) = w(t)|_y \,. \tag{2.33}$$

Wir interessieren uns nun für die Änderung einer zeitabhängigen Funktion $f(t,y), t \in \mathbb{R}, y \in G$, unter dem Fluß von w. Die Änderung unter einer reinen Bewegung des Argumentes $y \in G$ beschreiben wir durch die Operation $(\Psi^*_{t,w} f)(t,y) := f(t, \Psi_{t,w}(y))$. Damit ergibt sich

$$\begin{aligned}\frac{d}{dt}(\Psi^*_{t,w} f)(y) =& \frac{d}{dt} f(t, \Psi_{t,w}(y)) \\ =& f_t(t, \Psi_{t,w}(y)) + w(f)(t, \Psi_{t,w}(y)) \\ =& (\Psi^*_{t,w}(f_t + w(f))(y).\end{aligned} \tag{2.34}$$

Wenden wir dies mehrfach auf den Fall $f = v$ an, so ergibt sich

$$\begin{aligned}\frac{d}{dt}(\Psi^*_{t,w} v) =& \Psi^*_{t,w}(w(v)) \\ \frac{d^2}{dt^2}(\Psi^*_{t,w} v) =& \Psi^*_{t,w}(w_t(v) + w(w(v))) \\ \frac{d^3}{dt^3}(\Psi^*_{t,w} v) =& \Psi^*_{t,w}(w_{tt}(v) + w_t(w(v)) + 2w(w_t(v)) + w(w(w(v)))).\end{aligned} \tag{2.35}$$

Betrachtet man alle Ausdrücke für $t = t_n$, so ergibt sich

$$v^{(k)} := \frac{d^k}{dt^k}\Psi^*_{t,w} v = w_{t\ldots t}. \tag{2.36}$$

Somit ist (2.35) die Lie-Gruppen-Version von Faá di Brunos Formel [34]. In (2.35) wird (2.36) eingesetzt, und die erhaltenen Ausdrücke werden wie folgt in einfache Bäume übersetzt:

$$v^{(i_l)}(\ldots(v^{(i_1)}(v))\ldots) \tag{2.37}$$

entspricht einem einfachen Baum mit Ästen der Längen i_1,..., i_l. Dabei ist im Unterschied zum klassischen Fall die Anordnung der Äste wesentlich, wir setzen den Ast der Länge i_1 nach links. Indiziert man diese Bäume monoton, so ist nicht nur entlang eines jeden Astes monoton zu indizieren, sondern auch die mit der Wurzel direkt verbundenen Knoten müssen monoton wachsend von links nach rechts indiziert werden.

Nach und nach werden nun die vollständigen Ausdrücke für niedere Ableitungen (repräsentiert durch komplette Wurzelbäume) in die Darstellungen für höhere Ableitungen durch einfache Wurzelbäume eingesetzt. Die Zuordnung der Indizes erfolgt dabei so wie beim klassischen Fall. Da hier in den einfachen Bäumen die Indizes der Nachbarknoten der Wurzel monoton nach rechts wachsend angeordnet sind, überträgt sich diese Eigenschaft beim Einsetzen auf alle Knoten, daß heißt: Alle Knoten mit gemeinsamen Vaterknoten sind nach rechts monoton wachsend indiziert.

Die Ausdrücke $v(\ldots(v)\ldots)$ sind dabei wie folgt zu interpretieren: die Richtungsableitung $v(\ldots)$ bezieht sich nur auf das innerste v, alle anderen Terme sind als konstant anzusehen. Insofern entspricht $v((v(v))(v))$ gerade dem elementaren Differential $f_{yy}(f_y f, f)$ im Falle einer gewöhnlichen Differentialgleichung, wobei das innere v dem f_{yy} entspricht, und das äußere v dem 2.Argument f des Tensors f_{yy}, während der mittlere Operator $v(v)$ gerade dem ersten Argument $f_y f$ des Tensors f_{yy} entspricht.

Wir erhalten also im Prinzip die gleichen monoton indizierten Wurzelbäume wie beim klassischen Fall, ein (nach den Regeln des klassischen Falls) monoton indizierter Baum ist jetzt lediglich so darzustellen, daß die Indizierung aller Tochterknoten monoton von links nach rechts wachsend erfolgt. Wir schreiben also lediglich die Anordnung der Äste vor, ändern aber die Topologie des Baumes nicht.

2.2.3 Darstellung der numerischen Lösung

Betrachten wir nun die numerische Lösung. Sie ergibt sich aus

$$g_i = \sum_j a_{ij} k_j \tag{2.38}$$

$$Y_i = \exp(g_i) y_n \tag{2.39}$$

$$k_i = hv(Y_i). \tag{2.40}$$

Eine Entwicklung wie bei den gewöhnlichen Differentialgleichungen ist hier nicht so ohne weiteres möglich. Der Gleichung (2.39) entspricht im ODE-Fall $Y_i = y_n + g_i$, diese liefert sofort $Y_i^{(k)} = g_i^{(k)}$. Hier ergeben sich bereits bei einmaliger Differentiation nach der Schrittweite h wesentlich kompliziertere Ausdrücke:

$$Y_i' = \mathrm{dexp}_{g_i}(g_i') Y_i \tag{2.41}$$

$$= (g_i' - \frac{1}{2}[g_i, g_i'] + \mathcal{O}\left(h^4\right)) Y_i. \tag{2.42}$$

In [73] wird das Problem umgangen, indem man die Verfahrensvorschrift modifiziert. Die g_i werden durch korrigierte Werte $\tilde{g}_i$ ersetzt, so daß $Y_i := \exp(\tilde{g}_i) y_n$ wieder in den Ableitungen bis zur notwendigen Ordnung mit $\tilde{g}_i$ übereinstimmt: $Y_i^{(n)} = \tilde{g}_i^{(n)}$ für n=1, $\ldots$, $p-1$. In y_{n+1} wird ebenso verfahren, nur wird noch eine zusätzliche Ableitung abgeglichen.

Zur Bestimmung der notwendigen Korrekturen bis zur Ordnung 4 nutzen wir die Reihenentwicklung der dexp^{-1}-Abbildung. In [73] findet sich eine Herleitung für linksinvariante Vektorfelder unter Nutzung einer Entwicklung für dexp, wobei sich diese vom rechtsinvarianten Fall unterscheidet.

Wir lassen den Index i weg und erhalten:

$$\mathrm{dexp}_{\tilde{g}}(\tilde{g}') = g' \Rightarrow$$

$$\tilde{g}' = \mathrm{dexp}_{\tilde{g}}^{-1}(g') = g' - \frac{1}{2}[\tilde{g}, g'] + \frac{1}{12}[\tilde{g}, \tilde{g}, g'] + \mathcal{O}\left(h^5\right). \tag{2.43}$$

Man beachte, daß $\tilde{g} = \mathcal{O}(h)$ und $[\tilde{g}, g'] = \mathcal{O}\left(h^2\right)$. Wir bezeichnen die l-ten Ableitungen von $\tilde{g}, g$ in $h = 0$ mit $\tilde{g}^{(l)}, g^{(l)}$. Sie ergeben sich rekursiv durch Differentiation von (2.43)

$$\begin{aligned}
\tilde{g}^{(1)} &= g^{(1)} \\
\tilde{g}^{(2)} &= g^{(2)} - \frac{1}{2}[\tilde{g}^{(1)}, g^{(1)}] = g^{(2)} \\
\tilde{g}^{(3)} &= g^{(3)} - 2\frac{1}{2}[\tilde{g}^{(1)}, g^{(2)}] - \frac{1}{2}[\tilde{g}^{(2)}, g^{(1)}] + \frac{1}{12}[\tilde{g}^{(1)}, \tilde{g}^{(1)}, g^{(1)}] \\
&= g^{(3)} - \frac{1}{2}[g^{(1)}, g^{(2)}]
\end{aligned} \tag{2.44}$$

$$\begin{aligned}\tilde{g}^{(4)} =& g^{(4)} - \frac{1}{2}[\tilde{g}^{(3)}, g^{(1)}] + \frac{3}{2}[[\tilde{g}^{(2)}, g^{(2)}] - \frac{3}{2}[\tilde{g}^{(1)}, g^{(2)}] + \frac{1}{4}[\tilde{g}^{(2)}, \tilde{g}^{(1)}, g^{(2)}] \\ &+ \frac{1}{4}[\tilde{g}^{(1)}, \tilde{g}^{(2)}, g^{(1)}] + \frac{1}{2}[\tilde{g}^{(1)}, \tilde{g}^{(1)}, g^{(2)}] \\ =& g^{(4)} - [g^{(1)}, g^{(3)}].\end{aligned}$$

Eine Korrektur bis zur Ordnung 3 ergäbe sich somit zu

$$\tilde{g}_i = g_i + \frac{1}{12}h^3[g_i^{(1)}, g_i^{(2)}] + \mathcal{O}\left(h^4\right) \text{ oder} \tag{2.45}$$

$$\tilde{g}_i = g_i + \frac{h}{6}[g_i^{(1)}, g_i] + \mathcal{O}\left(h^4\right). \tag{2.46}$$

Die Ordnung 4 ergibt sich durch

$$\tilde{g}_i = g_i + \frac{h^3}{12}[g_i^{(1)}, g_i^{(2)}] - \frac{h^4}{24}[g_i^{(1)}, g_i^{(3)}] + \mathcal{O}\left(h^5\right) \text{ oder} \tag{2.47}$$

$$\tilde{g}_i = g_i - \frac{h}{4}[g_i^{(1)}, g_i] - \frac{1}{6}h^2[g_i^{(2)}, g_i]. \tag{2.48}$$

Dabei ist für g_i jeweils der von h abhängige Werte einzusetzen, während die Ableitungen $g_i^{(1)}, \ldots$ an der Stelle $h = 0$ geeignet zu approximieren sind. Dies kann durch geeignete Linearkombinationen der g_i geschehen, siehe auch [73]. Für Verfahren der Ordnung p genügt es, in den internen Stufen Korrekturen bis zur Ordnung $p-1$ vorzunehmen, und in der Aufdatierung y_{n+1} Korrekturen bis zur Ordnung p. Für konkrete Verfahren verweisen wir auf [73].

Bemerkung 2.2.1. *Die gesuchte exakte Korrektur $\tilde{g}_i$ ergibt sich als Lösung der Differentialgleichung $\operatorname{dexp}_{tg_i}(\tilde{g}_i') = g_i'$. Wir werden dieser Differentialgleichung in Kapitel 3 wiederbegegnen – sie kann zur Darstellung der Lösung linearer Differentialgleichungen genutzt werden. Hier läßt sich die Lösung des Anfangswertproblems $Y_i' = g_i'|_{Y_i}$, $Y_i|_{h=0} = y_n$ durch $Y_i = \exp(\tilde{g}_i)y_n$ darstellen, wobei $\tilde{g}_i$ der Differentialgleichung* (2.43) *mit dem Anfangswert $\tilde{g}_i|_{h=0} = 0$ genügt.*

Sind Korrekturfunktionen genügend hoher Ordnung vorhanden, ergibt sich die Entwicklung der numerischen Lösung genau wie im Falle gewöhnlicher Differentialgleichungen. Wir definieren h-abhängige Vektorfelder w_i mit

$$\frac{d}{dh}Y_i = w_i(h)|_{Y_i}. \tag{2.49}$$

Setzen wir eine exakte Korrektur durch

$$\begin{aligned}Y_i =& \exp(\tilde{g}_i)y_n, \tilde{g}_i = \zeta(g_i) \Rightarrow \\ Y_i^{(l)} =& g_i^{(l)}, \quad l = 1, \ldots\end{aligned} \tag{2.50}$$

voraus, so ergeben sich die Rekursionen

$$Y_i^{(l)} = l\sum_j a_{ij}(v(Y_j))^{(l-1)} \text{ wobei} \tag{2.51}$$

$$v(Y_j)^{(l)} = \left(\frac{d}{dh}\right)^l \Psi^*_{h,w_j}v \text{ und} \tag{2.52}$$

$$w_j^{(l)} = (v(Y_j))^{(l-1)}. \tag{2.53}$$

Man entwickelt zunächst (2.52) gemäß (2.35), und kann dann schrittweise die Ableitungen $v(Y_j)^{(1)}$, $v(Y_j)^{(2)}, \ldots$ bestimmen. Es ergeben sich dann wie bei der Butcher-Theorie im ODE-Fall für die Bäume rekursiv die Funktionen γ, Φ_i, F, mit denen die Ordnungsbedingungen dargestellt werden können. Dabei werden in γ die Faktoren l aus Gleichung (2.51) akkumuliert, und in Φ_i die Faktoren a_{ij} aus (2.51). Die Ordnungsbedingungen stimmen mit den im ODE-Fall erhaltenen überein, da die Rekursionen identisch sind.

Es zeigt sich, daß Verfahren ohne Korrekturfunktionen höchstens die Ordnung 2 besitzen können. Wir wollen uns kurz von dieser Tatsache überzeugen. Betrachten wir die Entwicklung von Y_i bis zur Ordnung 3, und interpretieren dann die $s+1$-te Stufe als Aufdatierung y_{n+1}.

$$\begin{aligned}
Y_i' =& \operatorname{dexp}_{g_i}(g_i') \\
=& g_i' + \frac{1}{2}[g_i, g_i'] + \frac{1}{6}[g_i, g_i, g_i'] + \mathcal{O}\left(h^4\right) \\
Y_i^{(1)} =& g_i^{(1)} = \sum_j a_{ij} v = c_i v \\
Y_i^{(2)} =& g_i(2) = \sum_j a_{ij} c_j v(v) \\
Y_i^{(3)} =& g_i^{(3)} + \frac{1}{2}[g_i^{(1)}, g_i^{(2)}] \\
=& \sum_j a_{ij} c_j^2 v(v(v)) + \sum_{j,k} a_{ij} a_{jk} (v(v))(v) + c_i \sum_j a_{ij} c_j [v, v(v)]
\end{aligned} \tag{2.54}$$

Die erste und zweite Ableitung sind analog zum ODE-Fall, aber in der dritten Ableitung taucht der Term $[v, v(v)]$ auf. Dieser ist nicht im Sinne von $v(v(v)) - (v(v))(v)$ zu verstehen, sondern so: zuerst sind die rechtsinvarianten Vektorfelder v und $v(v)$ zu bilden, und dann die Lie-Klammer als Operation in der Lie-Algebra. Ist die Lie-Gruppe eine Matrizengruppe, so ergibt sich die Klammer zu $[v, v(v)] = v \cdot v(v) - v(v) \cdot v$, wobei $\cdot$ die Matrizenmultiplikation bezeichnet.

2.3 Andere Zugänge

2.3.1 Matrixdarstellung

Nach Ado's Theorem sind Lie-Gruppen lokal äquivalent zu Matrix-Gruppen. Die vorhergehenden Ausführungen werden leichter verständlich, wenn wir zur Matrix-Darstellung übergehen. Die Lie-Gruppe G sei die Menge $GL(m)$ der regulären Matrizen der Dimension m, die Lie-Algebra als Tangentialraum an die Einheitsmatrix ergibt sich zu $\mathbb{R}^{m\times m}$. Die Rechtsmultiplikation ist durch die Matrixmultiplikation $R_A(B) = B \cdot A$ gegeben, die Ableitung ist eine lineare Abbildung der Tangentialräume

$$dR_A : TG|_B \to TG|_{B\cdot A}\,. \tag{2.55}$$

Sie ist für den Tangentenvektor $V = \frac{d}{dt}\big|_{t=0}\Gamma(t)$ definiert durch

$$\begin{aligned} dR_A(V) &= \left.\frac{d}{dt}\right|_{t=0} R_A(\Gamma(t)) \\ &= V \cdot A \end{aligned} \tag{2.56}$$

Wir haben hier bewußt nicht $R_A(V)$ geschrieben, da R_A als Operation in der Lie-Gruppe und nicht der Lie-Algebra definiert ist.

Die Tangentialräume $TG|_B$ sind gerade $\mathbb{R}^{m\times m}$, sie werden alle mit dem Tangentialraum an die Einheitsmatrix vermöge der Rechtsmultiplikation identifiziert. Für die Gruppe $GL(m)$ scheint dies nicht naheliegend zu sein, da man alle Tangentenräume auch direkt identifizieren könnte. Für die Gruppe der orthogonalen Matrizen wird der Vorteil der Vorgehensweise offensichtlich – über das Differential der Rechtsmultiplikation wird jeder Tangentialraum mit der Lie-Algebra $\mathfrak{so}(m)$ identifiziert, die allgemeine Gestalt eines Tangentenvektors V an eine orthogonale Matrix Q ist gerade

$$\begin{aligned} (Q+tV)(Q+tV)^T =& I + \mathcal{O}\left(t^2\right) \Rightarrow \\ (VQ^T)^T + VQ^T =& 0 \Rightarrow \\ V =& SQ \text{ mit } S^T + S = 0. \end{aligned} \tag{2.57}$$

Somit sind alle Tangentialräume mit dem Vektorraum der schiefsymmetrischen Matrizen identifiziert.

Die rechte Seite einer Differentialgleichung ist durch ein (im nichtautonomen Fall zeitabhängiges) Vektorfeld gegeben. Dieses Vektorfeld stellen wir mit Hilfe der Rechtsmultiplikation durch die Lie-Algebra dar – die autonome Differentialgleichung ergibt sich damit einfach zu

$$Y' = A(Y)Y \tag{2.58}$$

wobei im allgemeinen Fall $Y, A(\cdot) \in GL(m)$. Bei nichtautonomen Systemen kann A zusätzlich von t abhängen, doch diese stellen keine echte Erweiterung dar, man kann wie bei gewöhnlichen Differentialgleichungen das System autonomisieren und erhält eine Differentialgleichung in $G \otimes \mathbb{R}$. Die internen Stufen im Runge-Kutta-Ansatz werden zu

$$Y_{ni} = \exp(h\sum_j a_{ij}A(Y_{nj}))y_n, \tag{2.59}$$

die Aufdatierungsvorschrift ergibt sich analog. Der Vorteil einer solchen Formulierung besteht darin, daß alle internen Stufen bei Rechnung in exakter Arithmetik in der Matrixgruppe verbleiben. Dies setzt natürlich eine geeignete Auswertung der Exponentialabbildung voraus.

Wir leiten zunächst eine äquivalente Faá di Bruno-Formel her. Wir berechnen dabei die Zeitableitungen von $A(y(t))$ nicht explizit, sondern fügen sie als $A^{(k)}$ ein. Lediglich y wird differenziert und y' durch Ay ersetzt:

$$\begin{aligned} y' =& Ay \\ y'' =& A^{(1)}y + AAy \\ y''' =& A^{(2)}y + 2A^{(1)}Ay + AA^{(1)}y + AAAy \\ y^{(4)} =& A^{(3)}y + 3A^{(2)}Ay + 3A^{(1)}A^{(1)}y + +3A^{(1)}AAy + AA^{(2)}y \\ &+ 2AA^{(1)}Ay + AAA^{(1)}y + AAAAy \end{aligned} \tag{2.60}$$

Die Rekursion ist der bei ODEs analog, es entsprechen einander die Ausdrücke

$$\frac{\partial^k f}{\partial y^k}\left(y^{(i_1)}, \ldots, y^{(i_k)}\right) \leftrightarrow A^{(i_1-1)} \cdot \ldots \cdot A^{(i_k-1)}y. \tag{2.61}$$

Dabei übernimmt hier der jeweils letzte Faktor in jedem Term (y) die Rolle des Tensors $\frac{\partial^k f}{\partial y^k}$, und die Terme $A^{(i-1)}$ entsprechen $y^{(i)}$. Ein wesentlicher Unterschied besteht aber: Während die symmetrische Multilinearform unabhängig von der Reihenfolge der Argumente ist, hängen die Matrizenprodukte sehr wohl von der Reihenfolge der Faktoren ab, denn im allgemeinen werden die Ableitungen verschiedener Ordnung von $A(y(t))$ natürlich nicht vertauschbar sein.

Die totalen Zeitableitungen von A ergeben sich aus der klassischen Faá di Bruno-Formel:

$$\begin{aligned} A^{(1)} =& A_y[y'] \\ A^{(2)} =& A_{yy}[y', y'] + A_y[y''] \\ A^{(3)} =& A_{yyy}[y', y', y'] + 3A_{yy}[y', y''] + A_y[y'''] \end{aligned} \tag{2.62}$$

Die hierbei auftretenden Multilinearformen sind symmetrisch in den Argumenten. Beide Entwicklungen können nun abwechselnd ineinander eingesetzt werden, um nacheinander y', A', y'', A'', ... zu generieren. Zur Vermeidung von Verwechslungen haben wir eckige Klammern $A_{y\ldots y}[\ldots]$ zur Darstellung der Multilinearformen für die partiellen Ableitungen von A verwendet.

Zur Darstellung von y, A sowie der Ableitungen greifen wir auf die bewährten Butcher-Bäume [13] zurück, man siehe auch die einleitenden Ausführungen in Abschnitt 2.2. Wir verwenden zwei Arten von Knoten, dünne schwarze und fette weiße.

Die Menge der Bäume, die Ableitungen von y bzw. A repräsentieren, bezeichnen wir mit T_y bzw. T_A. Sie unterscheiden sich in der Wurzel; wir verwenden einen dünnen schwarzen Knoten als Wurzel für Ableitungen von y und einen fetten weißen Knoten als Wurzel für Ableitungen von A.

Für einen Baum mit dünner schwarzer Wurzel, an der die Äste $\tau_1, \ldots, \tau_k$ hängen, schreiben wir $[\tau_1, \ldots, \tau_k]$. Für einen Baum mit fetter weißer Wurzel, an der die Äste $\tau_1, \ldots, \tau_k$ hängen, schreiben wir $\{\tau_1, \ldots, \tau_k\}$. Folgerichtig bezeichnen wir mit $\{\}$ einen einzelnen weißen Knoten, der A repräsentiert.

Es gilt weiterhin: schwarze und weiße Knoten alternieren. Jeder Ast endet mit einem weißen Knoten. Die Ordnung eines Baumes ergibt sich aus der Anzahl der weißen Knoten. Für Bäume mit schwarzer Wurzel entspricht dies der repräsentierten Ableitungsordnung, für Bäume mit weißer Wurzel ist die Ableitungsordnung um eins größer.

Rekursiv sind die Bäume gegeben durch

$$\begin{aligned} \{\,\} &\in T_A \\ \tau_1, \ldots, \tau_k \in T_A &\Rightarrow [\tau_1, \ldots, \tau_k] \in T_y \\ \tau_1, \ldots, \tau_k \in T_y &\Rightarrow \{\tau_1, \ldots, \tau_k\} \in T_A \\ \text{Äquivalenz von } \{\tau_1, \ldots, \tau_k\} &= \{\tau_{\pi(1)}, \ldots, \tau_{\pi(k)}\} \\ \text{für } \pi &\in S_k. \end{aligned} \tag{2.63}$$

Wir haben bewußt den Baum $[\,]$ nicht zugelassen, der y repräsentieren könnte und die Ordnung 0 hätte, damit die Rekursionen möglichst einfache Gestalt besitzen. Wesentlich ist, daß bei Bäumen aus T_y die Anordnung der Äste an der Wurzel berücksichtigt wird (also bei allen schwarzen Knoten), bei Bäumen aus T_A (also allen weißen Knoten) hingegen nicht.

Die rekursive Definition ermöglicht eine einfache Formalisierung der repräsentierten elementaren Differentiale

$$\begin{aligned} F(\{\,\}) &= A \\ F([\tau_1, \ldots, \tau_k]) &= F(\tau_1) \cdot \cdots \cdot F(\tau_k) \cdot y \\ F(\{\tau_1, \ldots, \tau_k\}) &= \frac{\partial^n A}{\partial y^n}[F(\tau_1, \ldots, F(\tau_k))] \end{aligned} \tag{2.64}$$

sowie der Ordnung ρ:

$$\begin{aligned} \rho(\{\,\}) &= 1 \\ \rho([\tau_1, \ldots, \tau_k]) &= \sum_{i=1}^{k} \rho(\tau_i) \\ \rho(\{\tau_1, \ldots, \tau_k\}) &= 1 + \sum_{i=1}^{k} \rho(\tau_i). \end{aligned} \tag{2.65}$$

Wichtig ist, daß die elementaren Differentiale bei schwarzer Wurzel von der Anordnung der Äste abhängen. Wir tragen diesem Fakt Rechnung, indem wir die Klammer $\{\ldots\}$ als kommutativ definiert haben, die Klammer $[\ldots]$ jedoch nicht.

Die Bäume zur Darstellung von y bis zur Ordnung 4 finden sich in Abbildung 2.2 und 2.3. In Tabelle 2.1 haben wir die zugehörigen Differentiale für die Bäume aus Abbildung 2.2 angegeben. Dabei entsprechen schwarze Knoten den eckigen Klammern und weiße Knoten den geschweiften Klammern.

Mit obiger Konstruktion werden gerade alle elementaren Differentiale erzeugt, die in den Ableitungen von y und A auftreten, wie ein Vergleich mit den beiden Entwicklungen (2.60) und (2.62) zeigt. Auch wird $y^{(k)}$ durch Bäume mit Ordnung k (k weiße Knoten), und $A^{(k)}$ durch Bäume der Ordnung $k+1$ repräsentiert.

Die Koeffizienten in der Taylorentwicklung von y ergeben sich durch schrittweise Differentiation der Bäume. Dabei gehen wir zu monoton indizierten Wurzelbäumen über. Die weißen beziehungsweise schwarzen Knoten entsprechen gerade $A_{y\ldots y}$ bzw. y, jeder Baum stellt ein Produkt dar. Bei der Differentiation muß also „jeder Knoten“ einmal differenziert werden. Da $y' = Ay$, muß an jeden schwarzen Knoten ein weißer ganz rechts angehängt werden. Wird $A_{y\ldots y}$ differenziert, so wird eine zusätzliche partielle Ableitung angefügt mit der inneren Ableitung $y' = Ay$, die durch den Baum $[\{\}]$ repräsentiert wird, siehe Abbildung 2.4. Die Indizierung der Knoten geschieht wie folgt: Die weißen

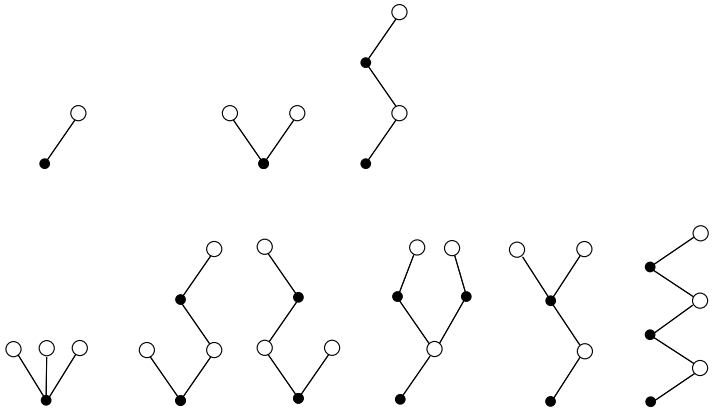


Abbildung 2.2: Die Bäume der Ordnung 1 bis 3.

Baum	Differential
$[\{\ \}]$	$A \cdot y$
$[\{\ \}, \{\ \}]$	$A \cdot A \cdot y$
$[\{[\{\ \}]\}]$	$A_y[A \cdot y] \cdot y$
$[\{\ \}, \{\ \}, \{\ \}]$	$A \cdot A \cdot A \cdot y$
$[\{\ \}, \{[\{\ \}]\}]$	$A \cdot A_y[A \cdot y] \cdot y$
$[\{[\{\ \}]\}, \{\ \}]$	$A_y[A \cdot y] \cdot A \cdot y$
$[\{[\{\ \}], [\{\ \}]\}]$	$A_{yy}[A \cdot y, A \cdot y] \cdot y$
$[\{[\{\ \}, \{\ \}]\}]$	$A_y[A \cdot A \cdot y] \cdot y$
$[\{[\{[\{\ \}]\}]\}]$	$A_y[A_y[A \cdot y] \cdot y] \cdot y$

Tabelle 2.1: Die elementaren Differentiale für die Matrix-Darstellung.

Knoten werden in der Reihenfolge ihrer Generierung indiziert. Wir erhalten damit Indizierungen, die (von der Wurzel aus) aufsteigend sind. Zusätzlich gilt: die mit dem gleichen schwarzen Vater-Knoten verbundenen weißen Knoten sind von links nach rechts monoton wachsend indiziert. Offensichtlich wird somit jeder monoton indizierte Wurzelbaum aus T_y genau einmal generiert, wenn wir, beginnend mit $y' = Ay = [\{\}]$, schrittweise differenzieren.

Bezeichnen wir die derart monoton indizierten Wurzelbäume mit LT_y, so ergibt sich für die exakte Lösung

$$y(t+h) = y(t) + \sum_{k=1}^{n} \frac{h^k}{k!} \sum_{\substack{\tau \in LT_y \\ \rho(\tau)=k}} F(\tau)(y(t)). \tag{2.66}$$

Die Darstellung der numerischen Lösung mit diesen Bäumen wird sehr technisch, Probleme macht die Exponentialabbildung. Einen besser geeigneten Ansatz stellen wir im folgenden vor.

2.3.2 Parametrisierung durch die Lie-Algebra

In [74] findet sich die entscheidende Idee, um die Darstellung zu vereinfachen. Man parametrisiert y in jedem Runge-Kutta-Schritt durch die Exponentialabbildung vermöge

$$y = \exp(w) y_n, \tag{2.67}$$

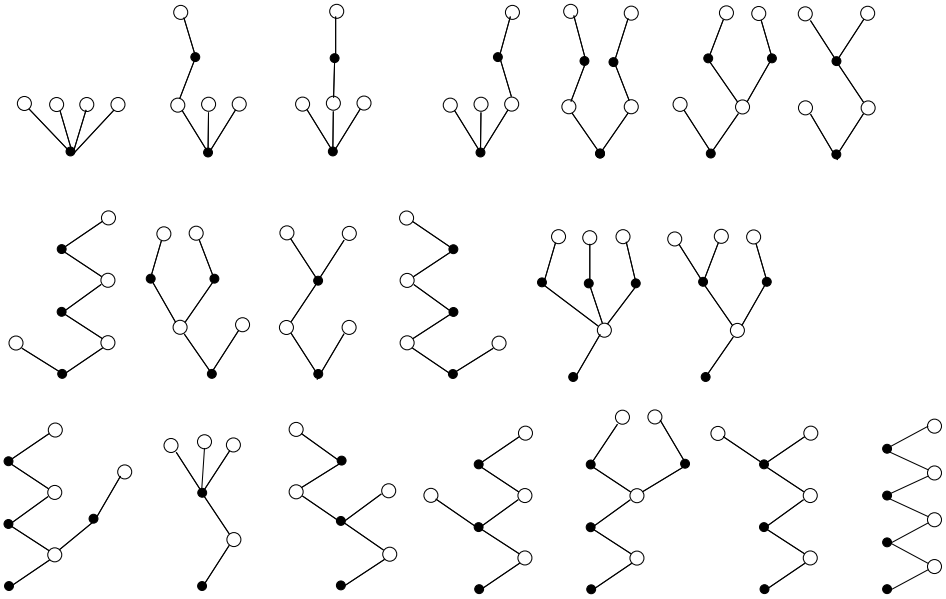


Abbildung 2.3: Die Bäume der Ordnung 4.

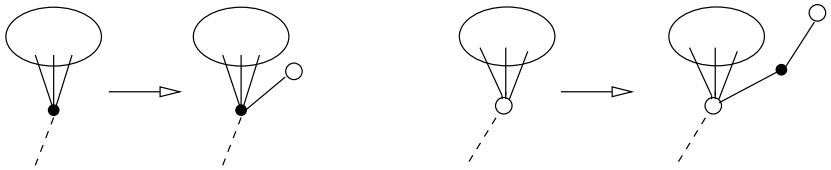


Abbildung 2.4: Generierung der Bäume durch Differentiation.

betrachtet w als neue Variable, und stellt eine Differentialgleichung für w auf. Analog wurde im Abschnitt 1.1 beim Toda-Lattice vorgegangen, wo allerdings die Abbildung cay zur Parametrisierung genutzt wurde. Hier gelangt man zu

$$\mathrm{dexp}_w(w') = v. \tag{2.68}$$

Die rechte Seite liegt damit als Reihe für die Abbildung dexp^{-1} vor:

$$w' = \mathrm{dexp}_w^{-1}(v) = \sum_{k=0}^{\infty} \frac{B_k}{k!} \mathrm{ad}_w^k(v) \tag{2.69}$$

$$= v - \frac{1}{2}[w,v] - \frac{1}{12}[w,w,v] + \dots. \tag{2.70}$$

Diese gewöhnliche Differentialgleichung in einem Vektorraum kann nun mit Standard-Verfahren integriert werden, wobei für jede Auswertung der rechten Seite eine Approximation der Reihe für dexp^{-1} notwendig ist. Es ergibt sich

$$Y_i = \exp(W_i) y_n \tag{2.71}$$

$$W_i = h \sum_j a_{ij} \, \mathrm{dexp}_{W_j}^{-1}(v(Y_j)) \tag{2.72}$$

$$y_{n+1} = \exp(h \sum_i b_i \, \mathrm{dexp}_{W_i}^{-1}(v(Y_i)). \tag{2.73}$$

Die Abbildung dexp^{-1} entspricht den Korrekturfunktionen im Original-Ansatz. Sie muß wieder mit hinreichender Genauigkeit approximiert werden.

Die Approximation der dexp^{-1}-Abbildung geschieht effizient mit dem Konzept der freien bewerteten Lie-Algebra, siehe [75]. Die Koeffizienten des Verfahrens werden von Standard-Verfahren übernommen, in [75] findet sich auch eine Version von DOPRI5 [28].

Dieser Ansatz erlaubt die Übertragung aller Verfahrensklassen für ODEs auf Lie-Gruppen, die Lie-Gruppen-Differentialgleichung wird auf eine ODE in einem Vektorraum zurückgeführt.

2.3.3 Parametrisierung mit der Cayley-Abbildung

Für spezielle Matrix-Gruppen steht neben der exponentiellen Abbildung eine weitere Abbildung zur Verfügung. Quadratische Lie-Gruppen sind dadurch charakterisiert, daß für eine feste Matrix S

$$X^T S X = S \quad \forall X \in G \tag{2.74}$$

gilt. Für die zugehörige Lie-Algebra bedeutet dies, daß für alle $A \in \mathfrak{g}$

$$A^T S + S A = 0 \tag{2.75}$$

gilt, wie man leicht durch Differentiation von (2.74) und einsetzen von $X' = AX$ erhält. Quadratische Lie-Gruppen können in einer Umgebung der Identität durch die Abbildung

$$\mathrm{cay}(A) = (I - A)^{-1}(I + A) \tag{2.76}$$

mit der Lie-Algebra parametrisiert werden. Eine solche quadratische Lie-Gruppe stellen die orthogonalen Matrizen dar. Man überzeugt sich leicht davon, daß $\mathrm{cay}(A)$ für schiefsymmetrisches A orthogonal ist:

$$\begin{aligned} \mathrm{cay}(A)^T \mathrm{cay}(A) =& (I + A^T)(I - A^T)^{-1}(I - A)^{-1}(I + A) \\ =& (I - A)(I + A)^{-1}(I + A)(I - A)^{-1} = I. \end{aligned} \tag{2.77}$$

Löst man (2.76) nach A auf, so ergibt sich umgekehrt, daß A schiefsymmetrisch ist, wenn $U := \mathrm{cay}(A)$ orthogonal ist:

$$\begin{aligned} A =& (U - I)(U + I)^{-1} \Rightarrow \\ A + A^T =& (U - I)(U + I)^{-1} + (U^{-1} + I)^{-1}(U^{-1} - I) \\ =& (U - I)(U + I)^{-1} + (I + U)^{-1}(I - U) = 0. \end{aligned} \tag{2.78}$$

Mit der Darstellung $Y = \mathrm{cay}(A) y_n$ ergibt sich durch Differentiation $Y' = \mathrm{dcay}_A(A')Y$, wobei wir wie schon bei der Exponentialabbildung

$$\mathrm{dcay}_A(B) := \left.\frac{d}{dt}\right|_{t=0} cay(A + tB)\,\mathrm{cay}(A)^{-1} \tag{2.79}$$

setzen, damit dcay in die Lie-Algebra abbildet. Es ergibt sich

$$\begin{aligned} \mathrm{dcay}_A(B)\,\mathrm{cay}(A) =& -(I - A)^{-1}(-B)(I - A)^{-1}(I + A) + (I - A)^{-1}B \\ \mathrm{dcay}_A(B) =& 2(I - A)^{-1}B(I + A)^{-1}. \end{aligned} \tag{2.80}$$

Die Cayley-Abbildung hat gegenüber der Exponentialabbildung verschiedene Vorteile

- Die numerische Berechnung der Abbildung cay ist deutlich weniger aufwendig als die Berechnung der Exponentialabbildung.
- Die Ableitung dcay läßt sich direkt berechnen, es ist keine Approximation wie bei dexp notwendig.

Als Nachteil ist nicht nur zu nennen, daß cay nur für quadratische Lie-Gruppen angewandt werden kann. Im Falle rechtsinvarianter Vektorfelder liefert die Parametrisierung mit der Exponentialabbildung die exakte Lösung, für cay ist dies nicht der Fall.

Methoden auf dieser Basis sind in [26, 27], [70] untersucht worden.

2.3.4 Andere Verfahren im Überblick

Der Übertragung von Extrapolationsmethoden ist das nun folgende Kapitel gewidmet. Zuvor wollen wir noch einen kurzen Abriß weiterer Verfahren und Techniken im Zusammenhang mit Differentialgleichungen auf Lie-Gruppen geben.

Weitere Beiträge zur RK-MK-Klasse findet man in [32]. Mit der Übertragung linearer Mehrschritt-Verfahren auf Lie-Gruppen beschäftigt sich [35]. Die Kollokationsidee läßt sich ebenfalls übertragen, man siehe [125]. Hierzu bedarf es aber einer Prozedur zur Quadratur in Lie-Gruppen (also lineare Differentialgleichungen mit zeitabhängigen Koeffizienten bei Verwendung der Matrixdarstellung). Diese steht mit der Magnus-Methode zur Verfügung, die wir in Kapitel 4 einsetzen werden.

Bei der Untersuchung impliziter Methoden stößt man schnell auf weitere numerische Problemstellungen, so die Lösung nichtlinearer Gleichungen mit dem Newton-Verfahren [78] oder Interpolation [69].

Wir wollen eine letzte Bemerkung zur zweiten großen Klasse geometrischer Integratoren machen — symplektische Verfahren. Hamiltonsysteme generieren einen symplektischen Fluß – symplektische Verfahren erhalten diese Eigenschaft, der von ihnen generierte Fluß ist ebenfalls eine symplektische Transformation. Wir erhalten somit keine Invariante der Lösung, vielmehr erhalten wir eine Eigenschaft, die nur an einer Schar von Lösungen überprüft werden kann.

Es zeigt sich, daß für symplektische Verfahren eine Rückwärtsfehleranalyse in dem Sinne möglich ist, daß die numerische Lösung gleichzeitig die exakte Lösung eines (benachbarten) Hamiltonsystems ist. Das qualitative Verhalten der Lösung gleicht daher dem eines Hamiltonsystems, und insbesondere für lange Integrationsintervalle zeigt sich ein verbessertes Fehlerverhalten, [45].

Einen Überblick über das Gebiet bieten die Monographien [87] und [47]. Interessante Ansätze unter der Einbeziehung von Extrapolation liefern unter anderem [124], [17].

2.4 Zusammenfassung

Wir haben hier den grundlegenden Ansatz zur Lösung von Differentialgleichungen auf Lie-Gruppen durch Einschrittverfahren vorgestellt. Er führt zum einen zu den RK-MK-Methoden, zum anderen läßt sich, wie im folgenden Kapitel 3, darauf eine Theorie der Extrapolationsverfahren aufbauen. Um beide Vorgehensweisen vergleichen zu können, haben wir hier die von Munthe-Kaas entwickelte RK-MK-Klasse beschrieben, und die Schritte der Entwicklung – beginnend vom Fall kommutativer Lie-Gruppen, der dem ODE-Fall äquivalent ist, über die Herleitung von RK-MK-Verfahren mit Korrektur-Funktionen, bis zur Parametrisierung durch die Lie-Algebra – dargestellt. Als Bindeglied zum klassischen ODE-Fall stellen wir die Interpretation als Matrix-Differentialgleichung dar, geben eine Klasse von Wurzelbäumen zur Darstellung der analytischen Lösung an, wo aufgrund der Nicht-Kommutativität die Anordnung der Äste (im Gegensatz zum ODE-Fall) berücksichtigt wird.

Kapitel 3

Extrapolationsverfahren in Lie-Gruppen

3.1 Einfache Extrapolation

3.1.1 Asymptotische Entwicklung des globalen Fehlers

Wir betrachten Einschrittverfahren der Form

$$y_{n+1} = \exp(h\Phi(t_n, y_n, h))y_n \tag{3.1}$$

mit der Inkrementfunktion Φ zur Lösung der Lie-Gruppen-Differentialgleichung

$$y'(t) = f(y(t))|_{y(t)}\,, \quad f : G \to \mathfrak{g}, \tag{3.2}$$

$$y(0) = y_0, \tag{3.3}$$

wobei die Elemente der Lie-Algebra als rechtsinvariante Vektorfelder interpretiert werden.

Satz 3.1.1. *Die durch eine Einschrittmethode von der Ordnung p mit Inkrementfunktion Φ berechneten numerischen Näherungen besitzen eine asymptotische Entwicklung nach Potenzen der Schrittweite h der Form*

$$y_n = \exp(h^p e_p(t_n)) \exp(h^{p+1} e_{p+1}(t_n)) \dots \exp(h^N e_N(t_n)) \exp(\mathcal{O}\left(h^{N+1}\right))y(t_n), \tag{3.4}$$

wobei $y(t_n)$ die exakte Lösung an der Stelle t_n ist und die e_i glatte Abbildungen vom Integrationsintervall $[t_0, t_e]$ in die Lie-Algebra $\mathfrak{g}$ darstellen.

Beweis. Wir nutzen die Technik aus [43] im Lie-Gruppen-Kontext. Wir konstruieren für $i = 0, \dots, N - p + 1$ Approximationen $y_n^{(i)}$ der Ordnung $p + i$ durch

$$y_n^{(0)} = y_n \tag{3.5}$$

$$y_n^{(i+1)} = \exp(-h^{p+i} e_{p+i}(t_n))y_n^{(i)}. \tag{3.6}$$

Möge $y_n^{(i)}$ die Ordnung $p + i$ besitzen. Für $i = 0$ gilt dies sicher. Um die Ordnung $p + i + 1$ für $y_n^{(i+1)}$ sicherzustellen, konstruieren wir Inkrementfunktionen $\Phi^{(i+1)}$, die gerade die $y_n^{(i+1)}$ erzeugen:

$$\exp(h\Phi^{(i+1)}(t_n, y_n^{(i+1)}, h))y_n^{(i+1)} = y_{n+1}^{(i+1)} \tag{3.7}$$

Einsetzen der Rekursion (3.6) liefert

$$\begin{aligned}\exp(h\Phi^{(i+1)}(t_n, y_n^{(i+1)}, h))y_n^{(i+1)} &= \exp(-h^{p+i}e_{p+i}(t_n+h))y_{n+1}^{(i)} \\ &= \exp(-h^{p+i}e_{p+i}(t_n+h))\exp(h\Phi^{(i)}(t_n, y_n^{(i)}, h))y_n^{(i)} \\ &= \exp(-h^{p+i}e_{p+i}(t_n+h)) \\ &\quad \cdot \exp(h\Phi^{(i)}(t_n, \exp(h^{p+i}e_{p+i}(t_n))y_n^{(i+1)}, h))) \\ &\quad \cdot \exp(h^{p+i}e_{p+i}(t_n))y_n^{(i+1)}.\end{aligned}$$

Wir gelangen zu Rekursionen für $\Phi^{(i)}$:

$$\Phi^{(0)} = \Phi \tag{3.8}$$

$$\begin{aligned}\exp(h\Phi^{(i+1)}(t,y,h)) = &\exp(-h^{p+i}e_{p+i}(t+h))\cdot \\ &\exp(h\Phi^{(i)}(t, \exp(h^{p+i}e_{p+i}(t))y, h))\exp(h^{p+i}e_{p+i}(t)).\end{aligned} \tag{3.9}$$

Wir zeigen, daß e_{p+i} so gewählt werden kann, daß $\Phi^{(i+1)}$ die Ordnung $p+i+1$ besitzt. Exemplarisch führen wir dies für $i=0$ durch: Sei der lokale Fehler gegeben durch

$$le_h(t) = d_{p+1}h^{p+1} + \mathcal{O}\left(h^{p+2}\right).$$

Wir entwickeln Φ in eine Taylorreihe, wobei die BCH-Formel Anwendung findet:

$$\begin{aligned}h\Phi^{(1)}(t,y(t),h) = &- h^p e_p(t+h) + h\Phi(t, \exp(h^p e_p(t))y(t), h) + h^p e_p(t) \\ &+ \frac{1}{2}([-h^p e_p(t+h), h\Phi(\dots)] + [-h^p e_p(t+h), h^p e_p(t)] \\ &+ [h\Phi(\dots), h^p e_p(t)]) + \mathcal{O}\left(h^{p+2}\right) \\ = &- h^{p+1}e_p'(t) + h\Phi(t, \exp(h^p e_p(t))y(t), h) + [hf(y(t)), h^p e_p(t)] \\ &+ \mathcal{O}\left(h^{p+2}\right)\end{aligned}$$

Da die Methode Φ wenigstens die Ordnung 1 besitzt, gilt $\Phi(t,y,h) = f(y) + \mathcal{O}(h)$. Wir betrachten Φ als eine Funktion von $y \in G$ unter dem Fluß des rechtsinvarianten Vektorfeldes $h^p e_p(t)$ und entwickeln es in eine Lie-Reihe:

$$\begin{aligned}\Phi(t, \exp(h^p e_p(t))y(t), h) = &\Phi(t, y(t), h) + h^p e_p(t)[\Phi(t, \cdot, h)]|_{y(t)} + \mathcal{O}\left(h^{2p}\right) \\ = &\Phi(t, y(t), h) + h^p e_p(t)[f]|_{y(t)} + \mathcal{O}\left(h^{p+1}\right)\end{aligned}$$

Unter Verwendung von (2.19) erhalten wir:

$$h\Phi^{(1)}(t,y(t),h) = h\nu(t,y(t),h) + d_{p+1}(t)h^{p+1} - e_p'(t)h^{p+1} \tag{3.10}$$

$$+ e_p(t)[f]|_{y(t)}\, h^{p+1} + [f(y(t)), e_p(t)]h^{p+1} + \mathcal{O}\left(h^{p+2}\right) \tag{3.11}$$

Die Methode $\Phi^{(1)}$ ist von der Ordnung $p+1$, wenn e_p eine Lösung des Anfangswertproblems

$$e_p'(t) = e_p(t)[f]|_{y(t)} + [f(y(t)), e_p(t)] + d_{p+1}(t) \tag{3.12}$$

$$e_p(0) = 0 \tag{3.13}$$

ist.

Diese Schlußweise kann nun wiederholt werden, man bestimmt den führenden Fehlerkoeffizienten des erzeugten Verfahrens höherer Ordnung, und kann damit den nächsten Term in der Entwicklung des globalen Fehlers bestimmen. Die so konstruierten Verfahren $\Phi^{(i)}$ haben die Ordnung $p+i$, allerdings eben nur für das betrachtete spezielle Anfangswertproblem. Auch sind alle diese Verfahren stabil, wenn die Original-Inkrementfunktion Lipschitzstetig ist. Dies genügt, um auch im globalen Fehler (für dieses spezielle Anfangswertproblem) die Ordnung $p+i$ zu erhalten. □

3.1.2 Das Extrapolationsprinzip

Extrapolationsverfahren sind Einschrittverfahren, sie erweitern ein Grundverfahren (zumeist) niedriger Ordnung zu einer Klasse von Verfahren höherer Ordnung. Die zugrundeliegende Idee findet man bereits im Romberg-Verfahren zur numerischen Quadratur. Im komplizierteren Fall gewöhnlicher Differentialgleichungen basiert Extrapolation auf den Arbeiten von Gragg [40], Bulirsch und Stoer [10] sowie Stetter [94]. Mit dem Gragg-Bulirsch-Stoer-Verfahren steht ein effizienter Extrapolations-Algorithmus im $\mathbb{R}^n$ zur Verfügung. Bei der Erweiterung auf Lie-Gruppen sorgt die Nicht-Kommutativität für weitere Komplikationen.

Gegeben sei als Grundverfahren eine Einschrittmethode (3.1) mit Inkrementfunktion Φ für die Differentialgleichung (2.9). Um einen Schritt mit der Basisschrittweite H von y_n in t_n nach y_{n+1} in $t_{n+1} = t_n + H$ auszuführen, berechnen wir für $i = 1, \ldots, l$ Näherungen $y_{n+1,i}$ in t_{n+1} mit jeweils n_i Schritten mit der konstanten Schrittweite $h_i = H/n_i$. Dies führt für jedes $i \in \{1, \ldots, l\}$ zu einem Gesamtinkrement $H\Phi_i$ in der Form

$$\begin{aligned} y_{n+1,i} &= \exp(h_i\Phi_{i,n_i}) \cdot \cdots \cdot \exp(h_i\Phi_{i,2}) \cdot \exp(h_i\Phi_{i,1}) y_n && (3.14)\\ &=: \exp(H\Phi_i) y_n. && (3.15) \end{aligned}$$

Die asymptotische Entwicklung des globalen Fehlers ergibt zusammen mit der BCH-Formel

$$\begin{aligned} \exp(H\Phi_i) &= \exp(h_i^p e_p(t_{n+1})) \ldots \exp(h_i^N e_N(t_{n+1})) && (3.16)\\ &\quad \cdot \exp(\mathcal{O}\left(H^{N+1}\right)) \exp(H\nu(t, y_n, H)) && (3.17) \end{aligned}$$

$$H\Phi_i = H\nu(t, y_n, H) + h_i^p e_p(t_{n+1}) + \sum_{k=p+1}^{N} h_i^k \tilde{e}_k(t_{n+1}) + \mathcal{O}\left(h^{N+1}\right). \quad (3.18)$$

Man beachte, daß sich nur der Koeffizient vom $\mathcal{O}(h^p)$-Term direkt aus (3.16) ergibt, in die Terme höherer Ordnung gehen noch Kommutatoren aus der BCH-Formel ein.

Wenn die Φ_i mit hinreichender Ordnung berechnet sind, kann wie im Fall gewöhnlicher Differentialgleichungen auf höhere Ordnungen extrapoliert werden. Die Berechnung kann mit Hilfe der BCH-Formel durchgeführt werden, wenn genügend viele Kommutatoren verwendet werden. Wir führen dies im folgenden für Basis-Methoden Φ der Ordnung 1 und 2 aus.

3.1.3 Extrapolation für Grundverfahren der Ordnung 1

Wir betrachten hier exemplarisch die explizite Euler-Methode. Um auf die Ordnung p zu extrapolieren, müssen wir die Φ_i bis zur Ordnung p berechnen, d.h., ein Fehler von $\mathcal{O}\left(h^{p+1}\right)$ ist zulässig. Es gilt mit $h = h_i$

$$\begin{aligned} \exp(H\Phi_i) &= \exp(h\Phi_{i,n_i}) \cdot \ldots \cdot \exp(h\Phi_{i,1}) \text{ mit} && (3.19)\\ \Phi_{i,j} &= f(y_n) + \mathcal{O}(H). && (3.20) \end{aligned}$$

Für den einfachen Kommutator gilt somit

$$[h\Phi_{i,j_1}, h\Phi_{i,j_2}] = [h\Phi_{i,j_1}, h\Phi_{i,j_2} - h\Phi_{i,j_1}] = \mathcal{O}\left(h^3\right). \tag{3.21}$$

Für einen iterierten Kommutator mit k Termen (wir sagen: $(k-1)$-facher iterierter Kommutator oder Kommutator $(k-1)$-ter Stufe) erhalten wir

$$[h\Phi_{i,j_1}, \ldots, h\Phi_{i,j_k}] = \mathcal{O}\left(h^{k+1}\right).$$

Somit haben wir die Kommutatoren bis zur $p-2$-ten Stufe zu berechnen.

Betrachten wir die Kommutatoren als Korrekturfunktionen, so stimmt die Struktur mit den Korrekturfunktionen bei den RK-MK-Methoden [73, 74] überein. Keine Korrekturen sind für Ordnung 2 notwendig, für Ordnung 3 sind Kommutatoren erster Stufe notwendig, für Ordnung 4 benötigt man Kommutatoren zweiter Stufe.

Zur Implementierung nutzen wir die harmonische Folge $n_i = i$. Für $h\Phi_1$ sind keine Korrekturfunktionen notwendig. Für $n_2 = 2$ ergibt sich $h\Phi_2$ direkt aus der BCH-Formel, angewandt auf $\Phi_{2,2}$ und $\Phi_{2,1}$. Dabei werden nur Kommutatoren bis zur Ordnung $p-2$ berücksichtigt. Zur Bestimmung von Φ_i, $i > 2$, kann die BCH-Formel mehrfach angewendet werden. Diese Vorgehensweise ist allerdings nicht sehr effektiv, sie führt zu einer großen Zahl von Kommutatoren für höhere Ordnungen. Wir berechnen daher direkt geeignete Approximationen.

Vor uns steht nun die Aufgabe, Versionen der BCH-Formel der Approximationsordnungen $p = 3, 4$ für jeweils $n = 2, 3, 4$ Exponentialfunktionen zu finden, also Z_n mit

$$\exp(Z_2) = \exp(X_1)\exp(X_1 + X_2) + \mathcal{O}\left(h^{p+1}\right) \tag{3.22}$$

$$\exp(Z_3) = \exp(X_1)\exp(X_1 + X_2)\exp(X_1 + X_3) + \mathcal{O}\left(h^{p+1}\right) \tag{3.23}$$

$$\exp(Z_4) = \exp(X_1)\exp(X_1 + X_2)\exp(X_1 + X_3)\exp(X_1 + X_4) + \mathcal{O}\left(h^{p+1}\right) \tag{3.24}$$

$$\tag{3.25}$$

wobei $X_1 = \mathcal{O}(h)$ und $X_2, X_3, X_4 = \mathcal{O}\left(h^2\right)$.

Gesamtordnung 2

Der Kommutator niedrigster Ordnung $[X_1, X_2]$ ist bereits dritter Ordnung in h, so daß keinerlei Kommutatoren berücksichtigt werden müssen.

Gesamtordnung 3

Hier müssen nur einfache Kommutatoren berücksichtigt werden, Kommutatoren zweiter Stufe sind bereits von vierter Ordnung in h. Wir erhalten

$$Z_2 = 2X_1 + X_2 + [X_1, X_2] \tag{3.26}$$

$$Z_3 = 3X_1 + X_2 + X_3 + [X_1, X_3]. \tag{3.27}$$

Diese Varianten sind offensichtlich optimal, da sie nur jeweils einen Kommutator enthalten.

Gesamtordnung 4

Wir wollen hier ins Detail gehen und die Berechnung optimaler Formeln mit Mathematica beschreiben.

2 Exponentialfunktionen Die freie Lie-Algebra mit Termen bis zur Ordnung 4 besteht aus 4 Elementen:

$$\{X_1, X_2, [X_1, X_2], [X_1, X_1, X_2]\} \tag{3.28}$$

mit jeweils Ordnung $1, 2, 3, 4$ in h. Die Lösung ergibt sich identisch zur Ordnung 3, da der einzige Term 4. Ordnung $[X_1, X_1, X_2]$ in Z_2 nicht auftritt:

$$Z_2 = 2X_1 + X_2 + [X_1, X_2] \tag{3.29}$$

3 Exponentialfunktionen Die freie Lie-Algebra mit Termen bis zur Ordnung 4 enthält die 8 Elemente

$$\{X_1, X_2, X_3, [X_1, X_2], [X_1, X_3], [X_2, X_3], [X_1, X_1, X_2], [X_1, X_1, X_3]\} \tag{3.30}$$

mit jeweils Ordnung $1, 2, 2, 3, 3, 4, 4, 4$ in h. Man erhält

$$Z_3 = 3X_1 + X_2 + X_3 + [X_1, X_2] + \frac{1}{2}[X_1, X_3] - \frac{1}{3}[X_1, X_1, X_2] + \frac{1}{6}[X_1, X_1, X_3] \tag{3.31}$$

Dieser Ausdruck läßt sich offensichtlich mit 4 Kommutatoren auswerten. Durch geschicktes Zusammenfassen ergibt sich folgende Variante mit 3 Kommutatoren:

$$C_1 = [X_1, X_2], C_2 = [X_1, X_3], C_3 = \frac{1}{6}[X_1, -2C_1 + C_2].$$

Naturgemäß stellt sich die Frage, ob diese Variante optimal ist. Wir versuchen, den Ausdruck Z_3 mit 2 Kommutatoren zu berechnen und setzen an:

$$C_1 = [X_1 + a_1X_2 + a_2X_3, a_3X_1 - 2X_2 + X_3] \tag{3.32}$$
$$C_2 = [X_1 + a_4X_2 + a_5X_3, a_6X_1 + a_7X_2 + a_8X_3 + C_1] \tag{3.33}$$
$$Z_3 = a_9X_1 + a_{10}X_2 + a_{11}X_3 + a_{12}C_1 + a_{13}C_2. \tag{3.34}$$

Ein Koeffizientenvergleich mit (3.31) liefert ein System von 8 polynomialen Gleichungen, der maximale Polynomgrad ist dabei 3. Mathematica gibt 4 verschiedene parametrisierte Lösungsscharen an, wir wählen davon eine aus und setzen so viele Koeffizienten wie möglich gleich null. Es ergibt sich die Lösung

$$C_1 = [X_1, -2X_2 + X_3] \tag{3.35}$$
$$C_2 = [X_1 - \frac{1}{4}X_3, \frac{1}{12}X_2 + C_1] \tag{3.36}$$
$$Z_3 = 3X_1 + X_2 + X_3 + C_1 + \frac{1}{6}C_2. \tag{3.37}$$

Diese Variante ist offensichtlich optimal hinsichtlich der Zahl der Kommutatoren, denn zur Generierung von $[X_1, X_1, X_2]$ werden zwei Kommutatoren benötigt.

4 Exponentialfunktionen Die freie Lie-Algebra mit Termen bis zur Ordnung 4 enthält die 13 Elemente

$$\begin{aligned}&\{X_1, X_2, X_3, X_4, [X_1, X_2], [X_1, X_3], [X_1, X_4], [X_2, X_3], [X_2, X_4], [X_3, X_4],\\ &[X_1, X_1, X_2], [X_1, X_1, X_3], [X_1, X_1, X_4]\}\end{aligned} \tag{3.38}$$

mit jeweils Ordnung $1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4$ in h. Hier besteht die Möglichkeit, die Formel für 2 Exponentialfunktionen sukzessiv anzuwenden, jeweils für den 1. und 2. sowie für den 3. und 4. Exponentialterm, und dann ein drittes Mal auf diese beiden Ergebnisse. Dies führt auf 6 Kommutatoren, was nicht befriedigend erscheint. Betrachten wir zunächst die Darstellung des gesuchten Z_4 in der freien Lie-Algebra.

$$\begin{aligned}Z_4 =&4X_1 + X_2 + X_3 + X_4 - \frac{1}{2}[X_1, X_2] + \frac{1}{2}[X_1, X_3] + \frac{3}{2}[X_1, X_4]\\ &+ \frac{1}{2}[X_2, X_3] + \frac{1}{2}[X_2, X_4] + \frac{1}{2}[X_3, X_4] - \frac{1}{2}[X_1, X_1, X_2]\\ &- \frac{1}{2}[X_1, X_1, X_3] + \frac{1}{2}[X_1, X_1, X_4]\end{aligned} \tag{3.39}$$

Wir versuchen, das System für einen Ansatz mit 2 Kommutatoren zu lösen, und bauen dabei gleich die Kommutatoren höchster Ordnung in den Ansatz ein:

$$C_1 = [X_1 + a_1X_2 + a_2X_3 + a_3X_4, a_4X_1 - \frac{1}{2}X_2 - \frac{1}{2}X_3 + \frac{1}{2}X_4] \tag{3.40}$$

$$C_2 = [X_1 + a_5X_2 + a_6X_3 + a_7X_4, a_8X_1 + a_9X_2 + a_{10}X_3 + a_{11}X_4 + C_1] \tag{3.41}$$

$$Z_4 = 4X_1 + X_2 + X_3 + X_4 + a_{12}C_1 + C_2 \tag{3.42}$$

Man erhält 9 Gleichungen mit 12 Variablen, Mathematica ermittelt 38 parametrisierte Lösungsscharen, wir wählen folgende Lösung aus:

$$C_1 = [X_1 + \frac{1}{3}X_4, -\frac{1}{2}X_2 - \frac{1}{2}X_3 + \frac{1}{2}X_4] \tag{3.43}$$

$$C_2 = [X_1 - \frac{1}{2}X_3, 4X_1 + 1X_2 + C1] \tag{3.44}$$

$$Z_4 = 4X_1 + X_2 + X_3 + X_4 + 3C_1 + C_2 \tag{3.45}$$

Übersicht

Wir fassen zunächst kurz den benötigten Aufwand für Ordnung $p = 1, 2, 3, 4$ zusammen:

	EXP			RK-MK		
p	$\#f$	$\#\exp$	$\#[\,]$	$\#f$	$\#\exp$	$\#[\,]$
1	1	1	0	1	1	0
2	3	2	0	2	2	0
3	6	4	2	3	3	1
4	10	7	6	4	4	2

(3.46)

Im Vergleich zu den RK-MK-Methoden benötigen wir hier, insbesondere für höhere Ordnungen, deutlich mehr Funktionsauswertungen, Exponentialfunktionen und Kommutatoren.

3.2 Quadratische Extrapolation

3.2.1 Symmetrische Methoden

Obwohl im allgemeinen die Schrittweite $h > 0$ vorausgesetzt wird, kann man fast alle Einschrittverfahren ohne weiteres mit negativer Schrittweite ausführen, in diesem Falle gilt eben $t_{n+1} < t_n$. Unter diesen Voraussetzungen kann man zu einem Einschrittverfahren ein adjungiertes oder auch gespiegeltes Verfahren definieren.

Definition 3.2.1. *Das zum Einschrittverfahren* (3.1) *adjungierte Verfahren*

$$y_{n+1} = \exp(h\Phi^*(t_n, y_n, h))y_n \tag{3.47}$$

ist implizit gegeben durch

$$y_n = \exp(-h\Phi(t_{n+1}, y_{n+1}, -h))y_{n+1}. \tag{3.48}$$

Bemerkung 3.2.1. *Wir betrachten hier ausschließlich autonome Differentialgleichungen, so daß für die üblichen Verfahren die Inkrementfunktion von der Zeit unabhängig ist. Wir belassen sie hier trotzdem als formales Argument der Inkrementfunktion.*

Interpretiert man das Grundverfahren mit Schrittweite h als Abbildung, so ist das adjungierte Verfahren mit der Schrittweite $-h$ gerade die inverse Abbildung, gewissermaßen macht das adjungierte Verfahren das Originalverfahren rückgängig. So ist das adjungierte Verfahren zur expliziten Euler-Methode

$$y_{n+1} = \exp(hf(y_n))y_n \tag{3.49}$$

die implizite Euler-Methode

$$y_{n+1} = \exp(hf(y_{n+1}))y_n, \tag{3.50}$$

das adjungierte Verfahren zur Trapezregel

$$y_{n+1} = \exp(h/2f(y_{n+1}))\exp(h/2f(y_n))y_n \tag{3.51}$$

ist die Trapezregel selbst. Im Rahmen der Extrapolationsverfahren sind Verfahren mit dieser Eigenschaft von besonderer Bedeutung.

Definition 3.2.2. *Ein Einschrittverfahren heißt symmetrisch, wenn es mit seinem adjungierten Verfahren übereinstimmt.*

An der Trapezregel erkennt man auch ein wichtiges Konstruktionsprinzip für symmetrische Methoden – sie ergeben sich durch Hintereinanderausführung eines Verfahrens und des adjungierten Verfahrens.

Bereits im ODE-Fall zeigt sich allerdings, daß symmetrische Verfahren stets implizit sind. Als Ausweg bietet sich, wie von Stetter [94] vorgeschlagen, die explizite Mittelpunktregel – ein Zweischrittverfahren – an.

Definition 3.2.3. *Die explizite Mittelpunktregel ist gegeben durch*

$$y_1 = \exp(hf(y_0))y_0 \tag{3.52}$$

$$y_{n+1} = \exp(2hf(y_n))y_{n-1} \tag{3.53}$$

Wir werden zeigen, daß die explizite Mittelpunktegel äquivalent zu einer symmetrischen Einschrittmethode ist. Unser Beweis lehnt sich an Stetters Beweis der Symmetrie der expliziten Mittelpunktregel im klassischen Fall an, wir weisen die Symmetrie aber nicht direkt wie in [94], [116] nach, sondern schreiben die Mittelpunktregel als zusammengesetzte Methode.

Wir betrachten

$$\begin{aligned} u_k &= \begin{cases} y_k & k \text{ gerade} \\ \exp(-hf(y_k))y_{k+1} & k \text{ ungerade} \end{cases} \\ v_k &= \begin{cases} \exp(-hf(y_k))y_{k+1} & k \text{ gerade} \\ y_k & k \text{ ungerade} \end{cases} \end{aligned} \tag{3.54}$$

sowie das verdoppelte System

$$\begin{aligned} u' &= f(v)|_u\,, \quad u(t_0) = y_0 \\ v' &= f(u)|_v\,, \quad v(t_0) = y_0. \end{aligned} \tag{3.55}$$

Für dieses System ist die symplektische Euler-Methode [47], die einen expliziten Euler-Schritt für v mit einem impliziten Euler-Schritt für u kombiniert, gegeben durch

$$\begin{aligned} u_{n+1} &= \exp(hf(v_{n+1}))u_n \\ v_{n+1} &= \exp(hf(u_n))v_n. \end{aligned} \tag{3.56}$$

Lemma 3.2.1. *Die adjungierte Methode zur symplektischen Euler-Methode ergibt sich durch Vertauschung von u und v, d.h.,*

$$\begin{aligned} u_{n+1} &= \exp(hf(v_n))u_n \\ v_{n+1} &= \exp(hf(u_{n+1}))v_n. \end{aligned} \tag{3.57}$$

Beweis. Durch wechselseitiges Vertauschen von u_n und u_{n+1}, wechselseitiges Vertauschen von v_n und v_{n+1} sowie Ersetzen von h durch $-h$ ergibt sich die Behauptung. □

Der entscheidende Schritt ist nun zu erkennen, daß gilt:

Lemma 3.2.2. *Die Lösungen u_k, v_k in* (3.54) *ergeben sich durch alternierende Anwendung der adjungierten symplektischen Euler-Methode* (3.57) *und der symplektischen Euler-Methode* (3.56) *auf die Startwerte $u_0 = v_0 = y_0$.*

Beweis. Durch Einsetzen überprüft man leicht die Richtigkeit der Behauptung. Zur Illustration stellen wir die ersten zwei Schritte im folgenden Tableau dar:

$$\begin{aligned} u_0 &= y_0 && \xrightarrow{\text{implizit}} && u_1 = \exp(hf(y_1))y_0 \\ v_0 &= \exp(-hf(y_0))y_1 && \xrightarrow{\text{explizit}} && v_1 = y_1 = \exp(hf(y_0))y_0 \end{aligned} \tag{3.58}$$

$$\begin{aligned} u_1 &= \exp(hf(y_1))y_0 && \xrightarrow{\text{explizit}} && u_2 = y_2 = \exp(2hf(y_1))y_0 \\ v_1 &= y_1 = \exp(hf(y_0))y_0 && \xrightarrow{\text{implizit}} && v_2 = \exp(-hf(y_2))y_3 = \exp(hf(y_2))y_1 \end{aligned} \tag{3.59}$$

□

Faßt man jeweils 2 Integrationsschritte zusammen, so ergeben diese nach dem oben Bemerkten eine symmetrische Einschrittmethode, da ein Verfahren und das adjungierte Verfahren hintereinander ausgeführt werden. Die geraden Werte u_{2k}, v_{2k} ergeben sich also aus einem symmetrischen Einschrittverfahren für das System (3.55). Von diesen approximieren die u_{2k} die gesuchte Lösung.

3.2.2 Quadratische Entwicklung des globalen Fehlers

Wir werden zeigen, daß symmetrische Methoden eine für die Extrapolation besonders günstige asymptotische Entwicklung des globalen Fehlers besitzen. Wir betrachten hier eine Einschrittmethode (3.1) mit Inkrementfunktion Φ der Ordnung p. Der Fehler $le_h(t)$ nach (2.13) besitze eine Entwicklung

$$le_h(t) = d_{p+1}h^{p+1} + \mathcal{O}\left(h^{p+2}\right). \tag{3.60}$$

Lemma 3.2.3. *Der lokale Fehler le^* der adjungierten Methode besitzt eine Entwicklung der Form*

$$le^*(t,h) = (-1)^p d_{p+1}(t)h^{p+1} + \mathcal{O}\left(h^{p+2}\right). \tag{3.61}$$

Beweis. Möge $\nu(t, y(t), h) \in \mathfrak{g}$ den exakten Fluß der Differentialgleichung vermöge

$$y(t+h) = \exp(h\nu(t, y(t), h))y(t) \tag{3.62}$$

generieren. Dann gilt insbesondere auch

$$y(t+h) = \exp(h\nu(t+h, y(t+h), -h))y(t). \tag{3.63}$$

Wir können aus der Darstellung des lokalen Fehlers, indem wir einen Schritt mit der Schrittweite $-h$ von $t+h$ nach t betrachten, folgern:

$$-h\Phi(t+h, y(t+h), -h) = -h\nu(t+h, y(t+h), -h) + d_{p+1}(-h)^{p+1} + \mathcal{O}\left(h^{p+2}\right). \tag{3.64}$$

Sei nun $y_n = y(t)$ und sei y^*_{n+1} der mit der adjungierten Methode berechnete Wert. Dieser ist durch

$$y_n = \exp(-h\Phi(t+h, y^*_{n+1}, -h))y^*_{n+1} \tag{3.65}$$

gegeben. Wir schreiben dies als Fixpunktgleichung für die Inkrementfunktion der adjungierten Methode – die Gleichung

$$w = h\Phi(t+h, \exp(w)y(t), -h) \tag{3.66}$$

hat gerade $\bar{w} = h\Phi^*(t, y(t), h)$ als Fixpunkt. Dieser ist auch eindeutig für $h \to 0$, da die rechte Seite eine Lipschitzkonstante $L = \mathcal{O}(h)$ für $w = \mathcal{O}(h)$ besitzt. Wir setzen in die rechte Seite der Fixpunktgleichung (3.66) $w_0 = h\nu(t+h, y(t+h), -h)$ als Startwert einer Fixpunktiteration ein:

$$\begin{aligned} h\Phi(t+h, \exp(w_0)y(t), -h) &= h\Phi(t+h, y(t+h), -h) && (3.67)\\ &= w_0 - d_{p+1}(t+h)(-h)^{p+1} + \mathcal{O}\left(h^{p+2}\right), && (3.68) \end{aligned}$$

wobei wir zunächst (3.63) und in der zweiten Zeile (3.64) verwendet haben. Nach Banachschem Fixpunktsatz haben wir

$$\|(\bar{w} - w_0) - (w_1 - w_0)\| \leq \frac{L}{1-L}\|w_1 - w_0\| = \mathcal{O}\left(h^{p+2}\right),$$

da $L = \mathcal{O}(h)$. □

Aus Gleichung (3.12) folgt nun, daß der führende Term in der asymptotischen Entwicklung des globalen Fehlers der adjungierten Methode durch $(-1)^p e_p(t)h^p$ gegeben ist. Dies können wir verallgemeinern:

Satz 3.2.4. *Sei die asymptotische Entwicklung einer Einschrittmethode* (3.1) *durch* (3.4) *nach Satz 3.1.1 gegeben. Dann besitzt die adjungierte Methode die asymptotische Entwicklung*

$$y_n = \exp(h^p e_p^*(t_n)) \exp(h^{p+1} e_{p+1}^*(t_n)) \dots \exp(h^N e_N^*(t_n)) \exp(\mathcal{O}\left(h^{N+1}\right)) y(t_n), \tag{3.69}$$

mit $e_k^*(t) = (-1)^k e_k(t)$ *für* $k = p, \dots, N$.

Beweis. Rekapitulieren wir noch einmal die Bezeichnungen aus Satz 3.1.1 über die asymptotische Entwicklung. Zur Methode Φ von der Ordnung p wurde eine Methode $\Phi^{(1)}$ der Ordnung $p+1$ mittels

$$\begin{aligned} \exp[h\Phi^{(1)}(t,y,h)] = &\exp[-h^p e_p(t+h)] \cdot \\ &\cdot \exp\left[h\Phi(t, \exp[h^p e_p(t)]y, h)\right] \cdot \exp[h^p e_p(t)] \end{aligned} \tag{3.70}$$

konstruiert. Die Exponentialfunktionen haben wir der Übersichtlichkeit halber komplett mit eckigen Klammern versehen. Wir zeigen zunächst: die Adjungierte $(\Phi^{(1)})^*$ von $\Phi^{(1)}$ ist durch $(\Phi^*)^{(1)}$ gegeben. Wir können $(\Phi^*)^{(1)}$ einfach angeben, wobei wir beachten, daß e_p in $(-1)^p e_p$ übergeht beim Übergang von Φ zu Φ^*:

$$\begin{aligned} y_1 := \exp[h(\Phi^*)^{(1)}(t,y,h)]y = &\exp[-(-h)^p e_p(t+h)] \cdot \\ &\underbrace{\cdot \exp\left[h\Phi^*(t, \exp[(-h)^p e_p(t)]y, h)\right] \exp[(-h)^p e_p(t)]y}_{=:\,\eta} \end{aligned} \tag{3.71}$$

Den Ausdruck η identifizieren wir als die Anwendung der Adjungierten auf das Argument $\exp[(-h)^p e_p(t)]y$ – also gilt

$$\exp[-h\phi(t+h, \eta, -h)]\eta = \exp[(-h)^p e_p(t)]y. \tag{3.72}$$

Betrachten wir nun andererseits $(\Phi^{(1)})^*$. Mit $y_2 := \exp[h(\Phi^{(1)})^*(t,y,h)]y$ gilt

$$\begin{aligned} y &= \exp[-h\Phi^{(1)}(t+h, y_2, -h)]y_2 \\ &= \exp[-(-h)^p e_p(t)] \cdot \exp\left[-h\Phi(t+h, \exp[(-h)^p e_p(t+h)]y_2, -h)\right] \cdot \\ &\quad \cdot \exp[(-h)^p e_p(t+h)]y_2 =: RHS \end{aligned} \tag{3.73}$$

Für hinreichend kleines h ist y_2 eindeutig bestimmt, es genügt also, wenn wir $y_2 = y_1$ in RHS einsetzen und Gleichheit nachweisen. Durch zweimaliges Einsetzen von (3.71) ergibt sich zunächst

$$RHS = \exp[-(-h)^p e_p(t)] \cdot \exp\left[-h\Phi(t+h, \eta, -h)\right] \cdot \eta \tag{3.74}$$

und hieraus folgt mit (3.72) die Behauptung $y = RHS$ und damit die Kommutativität des Diagramms

$$\begin{array}{ccc} \Phi & \xrightarrow{(3.70)} & \Phi^{(1)} \\ \text{adj}\downarrow & & \downarrow\text{adj} \\ \Phi^* & \xrightarrow{(3.70)} & (\Phi^{(1)})^* = (\Phi^*)^{(1)}. \end{array}$$

Ersetzt man also Φ durch Φ^*, so geht d_{p+1} in $(-1)^p d_{p+1}$, e_p in $(-1)^p e_p$, und die nach (3.9), (3.70) konstruierte Inkrementfunktion $\Phi^{(1)}$ in die Adjungierte $(\Phi^{(1)})^*$ über.

Iterieren wir den Prozeß $\Phi^{(i)} \to \Phi^{(i+1)}$, so ergibt sich eine Folge von kommutativen Diagrammen

$$\begin{array}{ccccccccc} \Phi & \xrightarrow{(3.9)} & \Phi^{(1)} & \xrightarrow{(3.9)} & \Phi^{(2)} & \xrightarrow{(3.9)} & \Phi^{(3)} & \xrightarrow{(3.9)} & \dots \\ \text{adj}\downarrow & & \text{adj}\downarrow & & \text{adj}\downarrow & & \text{adj}\downarrow & & \\ \Phi^* & \xrightarrow{(3.9)} & (\Phi^*)^{(1)} & \xrightarrow{(3.9)} & (\Phi^*)^{(2)} & \xrightarrow{(3.9)} & (\Phi^*)^{(3)} & \xrightarrow{(3.9)} & \dots . \end{array} \tag{3.75}$$

Wir folgern, daß beim Übergang zur adjungierten Methode e_{p+i} in $(-1)^p e_{p+i}$ übergeht. □

Wir können dieses Ergebnis auch so formulieren, daß sich die asymptotische Entwicklung der adjungierten Methode ergibt, indem h durch $-h$ in der asymptotischen Entwicklung der Ausgangsmethode ersetzt wird.

Als wichtigstes Ergebnis dieses Abschnitts erhalten wir damit

Satz 3.2.5. *Symmetrische Methoden besitzen eine asymptotische Entwicklung des globalen Fehlers in geraden Potenzen von h.*

Beweis. Eine symmetrische Methode stimmt mit ihrer adjungierten Methode übereinstimmt, daher fallen die ungeraden Potenzen in der asymptotischen Entwicklung des globalen Fehlers weg. □

3.2.3 Extrapolation mit der expliziten Mittelpunktregel

Die explizite Mittelpunktregel kann als symmetrische Einschrittmethode für die Approximationen y_{2k} interpretiert werden. Die geraden Werte besitzen daher eine asymptotische Entwicklung des globalen Fehlers in geraden Potenzen von h. Wir nutzen diese Approximationen y_{2k} zur Extrapolation.

Bemerkung 3.2.2. *Auch für die Werte an ungeraden Knoten läßt sich eine quadratische Entwicklung nachweisen, man nutzt dazu eine symmetrische Darstellung mittels der Lösung am vorhergehenden und am folgenden Knoten.*

Ein Schritt mit der Grundschrittweite H ergibt sich genauso wie im Abschnitt 3.1.2 beschrieben, nur sind jetzt alle Schrittzahlen n_i gerade. Der einfachste Extrapolationsschritt ergibt sich für die Schrittweitensequenz $(2, 4)$ - wir erhalten bei entsprechender Approximation der BCH-Formel eine Methode der Ordnung 4.

Der erste Teilschritt mit $n_1 = 2$ liefert

$$Y_{1,1} = \exp(H/2 f(y_n))y_n \tag{3.76}$$
$$h\Phi_1 = hf(Y_{1,1}). \tag{3.77}$$

Die eigentliche Lösung $Y_{1,2}$ an der Stelle $t+H$ wird nicht berechnet, da wir nur $h\Phi_1$ zur Extrapolation benötigen. Ebenso verfahren wir im 2. Teilschritt mit $n_2 = 4$

$$\begin{aligned} Y_{2,1} &= \exp(H/4 f(y_n))y_n \\ Y_{2,2} &= \exp(H/2 f(Y_{2,1}))y_n \\ Y_{2,3} &= \exp(H/2 f(Y_{2,2}))Y_{2,1} \\ H\Phi_{2,1} &= f(Y_{2,1}) \\ H\Phi_{2,2} &= f(Y_{2,3}) \\ H\Phi_2 &= H\Phi_{2,2} + H\Phi_{2,1} + \frac{1}{2}[H\Phi_{2,2}, H\Phi_{2,1}]. \end{aligned} \tag{3.78}$$

Aus Abschnitt 3.1.3 wissen wir bereits, daß die Approximation in (3.78) genau bis einschließlich der Ordnung 4 ist. Mit $H\Phi_1$ und $H\Phi_2$ wird dann der eigentliche Extrapolationschritt ausgeführt:

$$H\Phi = (4H\Phi_2 - H\Phi_1)/3 \tag{3.79}$$
$$y_{n+1} = \exp(H\Phi) y_n \tag{3.80}$$

Im Vergleich zu RK-MK-Methoden der Ordnung 4 benötigen wir nur einen Kommutator. Im Gegensatz zu gewöhnlichen Differentialgleichungen, wo die Extrapolationsverfahren eine Unterklasse der Runge-Kutta-Verfahren darstellen, sind die hier entwickelten Extrapolationsverfahren nicht in der RK-MK-Klasse enthalten.

3.2.4 Effiziente Approximationen an die BCH-Formel

Vor uns steht wie in Abschnitt 3.1.3 die Aufgabe, Approximationen an die BCH-Formel zu konstruieren, diesmal für die Approximationsordnungen $p = 2, 4, 6, 8$ für jeweils $n = 2, 3, 4$ Exponentialfunktionen. Davon wurden die Fälle $p = 2, 4$ bereits im oben erwähnten Abschnitt 3.1.3 gelöst.

Wir suchen also Z_n mit

$$\exp(Z_2) = \exp(X_1)\exp(X_1 + X_2) + \mathcal{O}\left(h^{p+1}\right) \tag{3.81}$$
$$\exp(Z_3) = \exp(X_1)\exp(X_1 + X_2)\exp(X_1 + X_3) + \mathcal{O}\left(h^{p+1}\right) \tag{3.82}$$
$$\exp(Z_4) = \exp(X_1)\exp(X_1 + X_2)\exp(X_1 + X_3)\exp(X_1 + X_4) + \mathcal{O}\left(h^{p+1}\right) \tag{3.83}$$
$$\tag{3.84}$$

wobei $X_1 = \mathcal{O}(h)$ und $X_2, X_3, X_4 = \mathcal{O}\left(h^2\right)$.

Darstellung für zwei Exponentialfunktionen

Für 2 Exponentialfunktionen läßt sich Z_2 noch mit vertretbarem Aufwand manuell aus der BCH-Formel ableiten. Betrachten wir die BCH-Formel in der Form

$$\exp(Z) = \exp(X)\exp(Y) \text{ mit} \tag{3.85}$$
$$Z = X + Y + \sum_{i \geq 2} C_i(X, Y). \tag{3.86}$$

Die Funktionen C_i sind Linearkombinationen von iterierten Kommutatoren $(i-1)$-ter Stufe der Form

$$[U_1, [\ldots, [U_{i-1}, U_i]\ldots]] \text{ mit } U_j \in \{X, Y\}, \tag{3.87}$$

man siehe [105]. Für den konkreten Fall $X = X_1$, $Y = X_1 + X_2$ gilt $[X, Y] = [X, Y - X] = \mathcal{O}\left(h^3\right)$, und somit

$$C_i(X_1, X_1 + X_2) = \mathcal{O}\left(h^{i+1}\right). \tag{3.88}$$

Für ungerades $i = 2k + 1$ können wir aber noch mehr zeigen.

Zum einen gilt offensichtlich

$$C_i(-X, -Y) = -C_i(X, Y), \tag{3.89}$$

zum anderen folgt aus $\exp(-Z) = \exp(-Y)\exp(-X)$ gerade

$$C_i(-Y,-X) = -C_i(X,Y). \tag{3.90}$$

Somit führt eine Vertauschung der Argumente in C_i gerade auf $C_i(Y,X) = C_i(X,Y)$ oder auch

$$C_i(Y,X) = \frac{1}{2}(C_i(X,Y) + C_i(Y,X)). \tag{3.91}$$

Obiger Ausdruck setzt sich aus Summanden der Form

$$[U_1,[U_2,\ldots,[U_{i-2},[X,Y]]\ldots]] + [V_1,[V_2,\ldots,[V_{i-2},[Y,X]]\ldots]] \tag{3.92}$$

mit $\{U_j,V_j\} = \{X,Y\}$ zusammen. In der innersten Klammer haben wir dabei o.B.d.A. $U_i = Y$ gesetzt.

Für $X = X_1$ und $Y = X_1 + X_2$ entwickeln wir diesen Ausdruck nach Potenzen von h, und bestimmen den Term niedrigster Ordnung ($\mathcal{O}\left(h^{i+1}\right)$). Die innerste Klammer ist bereits $\mathcal{O}\left(h^3\right)$, daher ergibt sich der gesuchte Term, indem $U_j = V_j = X_1$ für $j = 1,\ldots,i-2$ eingesetzt wird. Dieser Term

$$\begin{aligned} &[X_1,[\ldots,[X_1,X_1+X_2]\ldots]] + [X_1,[\ldots,[X_1+X_2,X_1]\ldots]] \\ =&[X_1,[\ldots,[X_1,X_1+X_2]\ldots]] + [X_1,[\ldots,-[X_1,X_1+X_2]\ldots]] \\ =&0 \end{aligned} \tag{3.93}$$

verschwindet aber, und somit gilt

$$C_i(X_1,X_1+X_2) = \mathcal{O}\left(h^{i+2}\right). \tag{3.94}$$

Als Konsequenz ergibt sich, daß wir für Ordnung $p = 4,6,8$ für 2 Exponentialfunktionen in der BCH-Formel für die gesuchte Approximation $Z_2^{(p)}$ Terme mit bis zu $i = 2,4,6$ Matrizen berücksichtigen müssen. Für $Z_2^{(p)}$ ergibt sich somit direkt aus der BCH-Formel

$$Z_2^{(4)} = 2X_1 + X_2 + \frac{1}{2}[X_1,X_2] \tag{3.95}$$

$$Z_2^{(6)} = 2X_1 + X_2 + \frac{1}{2}[X_1,X_2] - \frac{1}{12}[X_2,X_1,X_2] - \frac{1}{24}[X_1+X_2,X_1,X_1,X_2] \tag{3.96}$$

$$\begin{aligned} Z_2^{(8)} =& 2X_1 + X_2 + \frac{1}{2}[X_1,X_2] - \frac{1}{12}[X_2,X_1,X_2] - \frac{1}{24}[X_1+X_2,X_1,X_1,X_2] \\ &+ \frac{1}{240}[X_1,[X_1,[X_1,[X_1,[X_1,X_2]]]]] + \frac{1}{480}[X_1,[X_1,[X_1,[X_2,[X_1,X_2]]]]] \\ &- \frac{1}{360}[X_1,[X_1,[X_2,[X_1,X_2]]]]/360 + \frac{1}{160}[X_1,[X_2,[X_1,[X_1,[X_1,X_2]]]]] \\ &- \frac{1}{720}[X_1,[X_2,[X_2,[X_1,X_2]]]]/720 + \frac{1}{80}[X_2,[X_1,[X_1,[X_1,X_2]]]]/80 \\ &+ \frac{1}{90}[X_2,[X_1,[X_2,[X_1,X_2]]]] \end{aligned} \tag{3.97}$$

Mit der effektiven Bestimmung dieser Ausdrücke sowie der Approximationen für 3 und 4 Exponentialfunktionen werden wir uns in den folgenden Unterabschnitten beschäftigen.

Ordnung 6 für 2 Exponentialfunktionen In diesem Fall gelangt man noch mit einem geeigneten Ansatz manuell ans Ziel. Offensichtlich benötigt man mindestens 3 Kommutatoren, um $C := [X_1 + X_2, X_1, X_1, X_2]$ zu bestimmen. Wir setzen daher

$$\begin{aligned} C_1 =& [X_1, X_2] \\ C_2 =& [X_1, C_1] \\ C_3 =& [X_1 + X_2, \alpha C_1 + C_2] = C + \alpha[X_1, X_1, X_2] + \alpha[X_2, X_1, X_2]. \end{aligned} \tag{3.98}$$

Damit ist der Kommutator C generiert, und man erkennt leicht, daß für $\alpha = 2$

$$Z_2^{(6)} = 2X_1 + X_2 + \frac{1}{2}C_1 + \frac{1}{12}C_2 - \frac{1}{24}C_3 \tag{3.99}$$

die geforderten Eigenschaften besitzt.

Ordnung 8 für 2 Exponentialfunktionen Die Ausdrücke werden nun so unübersichtlich, daß wir uns Mathematica bedienen, um die Gleichungen aufzustellen und zu lösen. Mit dem Paket GLIEALG (siehe Abschnitt 3.3) generieren wir eine Basis der Lie-Algebra.

Die Basis der Lie-Algebra besteht jetzt aus 17 Elementen. Die Ausdrücke in der BCH-Formel werden übersichtlicher, wenn wir eine symmetrische Version entwickeln. Mathematica liefert dies durch $Z = \log(\exp(Y_1 - Y_2)\exp(Y_1 + Y_2))$. Wir erhalten

$$\begin{aligned} Z =& 2Y_1 + [Y_1, Y_2] - \frac{1}{3}[Y_2, Y_1, Y_2] - \frac{1}{12}([Y_1, Y_1, Y_1, Y_2] + [Y_2, Y_2, Y_1, Y_2]) \\ & - \frac{1}{90}[Y_1, Y_1, Y_2, Y_1, Y_2] + \frac{1}{20}[Y_2, Y_1, Y_1, Y_1, Y_2] \\ & + \frac{1}{120}[Y_1, Y_1, Y_1, Y_1, Y_1, Y_2]. \end{aligned} \tag{3.100}$$

Ohne tiefergehende Überlegungen kann man zunächst nur feststellen, daß mindestens 5 Kommutatoren notwendig sind, da ja $[Y_1, Y_1, Y_1, Y_1, Y_1, Y_2]$ generiert werden muß. In [117] wird eine Lösung mit 6 Kommutatoren angegeben. Wir versuchen hier einen Ansatz mit 5 Kommutatoren. Der komplette Ansatz beinhaltet die 35 Variablen $a_0, \ldots, a_{34}$ und generiert, beginnend mit $X_1 = Y_1 - Y_2$ und $X_2 = Y_1 + Y_2$, die Kommutatoren

$$\begin{aligned} C_1 =& [X_1, X_2] \\ C_2 =& [a_0 X_1 + a_1 X_2, C_1] \\ C_3 =& [a_2 X_1 + a_3 X_2 + a_4 C_1, a_5 X_1 + a_6 X_2 + a_7 C_1 + C_2] \\ C_4 =& [a_8 X_1 + a_8 X_2 + a_{10} C_1 + a_{11} C_2, \\ & \quad a_{12} X_1 + a_{13} X_2 + a_{14} C_1 + a_{15} C_2 + C_3] \\ C_5 =& [a_{16} X_1 + a_{17} X_2 + a_{18} C_1 + a_{19} C_2 + a_{20} C_3, \\ & \quad a_{21} X_1 + a_{22} X_2 + a_{23} C_1 + a_{24} C_2 + a_{25} C_3 + C_4] \\ Z_2^{(8)} =& a_{26} X_1 + a_{27} X_2 + a_{28} C_1 + a_{29} C_2 + a_{30} C_3 + a_{31} C_4 + a_{32} C_5 \end{aligned} \tag{3.101}$$

Der entstehende Ausdruck für $Z_2^{(8)}$ besteht aus 6488 atomaren Termen und setzt sich aus 1013 Summanden zusammen. Allein dies verdeutlicht, daß eine manuelle Auswertung dieser Ausdrücke nicht möglich ist. Mit Mathematica fassen wir die Koeffizienten vor den 17 Basiselementen zu 17 Gleichungen zusammen, die von unterschiedlicher Komplexität sind. Die einfachste Gleichung besteht aus 6

atomaren Ausdrücken ($-2 + a_{26} + a_{27} = 0$), die komplexeste aus 1095 atomaren Ausdrücken. Eine Gleichung ist identisch erfüllt, so daß 16 Gleichungen für 32 Variable verbleiben.

Symbolische Lösungsalgorithmen lösen polynomiale Gleichungen durch eine Transformation in Gröbner-Basen des erzeugten Ideals im Polynomring, wobei man als Ergebnis eine parametrisierte Darstellung der Lösung erhält. Dabei werden im Verlaufe des Algorithmus neue Elemente des Ideals generiert und andere dafür eliminiert, wobei die Menge der betrachteten Polynome im allgemeinen deutlich größer als die Eingabemenge oder die Lösungsmenge ist. Haben wir also deutlich mehr Variable als Gleichungen, so ist kaum damit zu rechnen, daß eine Lösung in vertretbarer Zeit gefunden wird. Günstiger ist es, wenn nur wenig mehr Variable als Gleichungen vorhanden sind.

Wir gehen daher wie folgt vor: Wir setzen so lange freie Variablen zu Null, bis das System keine Lösung mehr besitzt, und verwenden dann die zuletzt zu Null gesetzten Variablen wieder als freie Variable. Konkret setzen wir

$$a_0 = a_3 = a_6 = a_8 = a_{14} = a_{15} = a_{17} = a_{19} = a_{20} = a_{21} = a_{25} = 0. \tag{3.102}$$

Die obige Auswahl kam nach einer ganzen Reihe von Versuchen zustande, zu erwähnen ist, daß die Lösbarkeit des Systems noch nicht ausreicht, denn die berechnete Lösung muß auch noch reell sein! Eine weitere, allerdings nicht notwendige Forderung ist, daß die Lösung rational ist.

Damit verbleiben noch 21 Variablen für 16 Gleichungen, die mit Mathematica dann in ca. 1 Sekunde gelöst werden. Der maximale Polynomgrad ist dabei 5. Aus den 4 parametrisierten Lösungsscharen wählen wir eine aus. In dieser sind 6 freie Variable enthalten, also waren die Gleichungen nicht unabhängig.

Wir setzen noch $a_1 = a_9 = 1$ und $a_{13} = a_{16} = a_{22} = a_{32} = 0$ und erhalten die Lösung

$$\begin{aligned}
C_1 =& [X_1, X_2] \\
C_2 =& [X_2, C_1] \\
C_3 =& \frac{1}{240}[X_1, -\frac{7}{3}C_1 + C_2] \\
C_4 =& [\frac{3}{49}C_1 + X_2, C_3] \\
C_5 =& [X_1 + \frac{3}{14}C_1, \frac{13}{240}C_1 - \frac{7}{1080}C_2 + C_4] \\
Z_2^{(8)} =& X_1 + X_2 + \frac{1}{2}C_1 - \frac{1}{12}C_2 - 3C_3 + \frac{7}{3}C_4 + C_5.
\end{aligned} \tag{3.103}$$

Approximationen für drei und vier Exponentialfunktionen

Ordnung 6 für 3 Exponentialfunktionen Ohne Ausnutzung spezieller Eigenschaften der X_i konnte in [117] ein geeignetes $Z_3^{(6)}$ mit 6 Kommutatoren konstruiert werden. Wir stellen die erwähnte Lösung hier kurz dar:

Es bezeichne $Y_1 = X_1$, $Y_2 = X_1 + X_2$, $Y_3 = X_1 + X_3$.

Im Ergebnis erhalten wir

$$Z_3^{(6)} = Y_1 + Y_2 + Y_3 + d_3 - d_4 + O(h^7) \tag{3.104}$$

mit

$$\begin{aligned}
d_1 &= \left[Y_1 - \frac{13}{12}Y_2\,,\ \frac{11}{13}Y_2 - \frac{12}{13}Y_3\right] \\
d_2 &= \left[Y_1 - \frac{13}{11}Y_3 - \frac{1339}{704}d_1\,,\ \frac{11}{824}Y_2 + \frac{7}{6592}Y_3 - \frac{1053}{8192}d_1\right] \\
d_3 &= \left[Y_1 - Y_3 - \frac{3965}{1236}d_1 - \frac{8}{3}d_2\,,\ Y_2 + Y_3 - \frac{164957}{9888}d_1 + \frac{5}{3}d_2\right] \\
&\quad M_1 = Y_1 - Y_3 - \frac{2561}{309}d_1 + \frac{752}{3}d_2 - 2d_3 \\
&\quad M_2 = \frac{1}{2}Y_2 + \frac{1}{2}Y_3 - \frac{160745}{9888}d_1 - \frac{179}{3}d_2 + \frac{3}{8}d_3 \\
d_4 &= [M_1, M_2]
\end{aligned}$$

wobei wir mit 6 Kommutatoren auskommen. Dies erscheint für das so gestellte Problem optimal. Neben $Y_i - Y_j = \mathcal{O}\left(h^2\right)$ existieren jedoch noch weitere Abhängigkeiten unter den Y_i. Wir können das Resultat verbessern, indem wir folgendes Lemma ausnutzen, welches einfach aus der asymptotischen Entwicklung der numerischen Lösung folgt:

Lemma 3.2.6. *Es gilt*

$$\Phi_{i,j} - 2\Phi_{i,j+1} + \Phi_{i,j+2} = \mathcal{O}\left(h^2\right) \tag{3.105}$$

$$\Phi_{i,j} - 3\Phi_{i,j+1} + \Phi_{i,j+2} - \Phi_{i,j+3} = \mathcal{O}\left(h^3\right). \tag{3.106}$$

Mit Lemma 3.2.6 können wir unseren Ansatz wie folgt modifizieren:

$$\exp(Z_2^{(6)}) = \exp(X_1 - X_2)\exp(X_1)\exp(X_1 + X_2 + X_3) \tag{3.107}$$

$$\text{mit } X_1 = \mathcal{O}(h)\,, X_2 = \mathcal{O}\left(h^2\right), X_3 = \mathcal{O}\left(h^3\right). \tag{3.108}$$

Wir generieren mit Mathematica die entsprechende bewertete Lie-Algebra der Dimension 15. Für $Z_2^{(6)}$ ergibt sich durch Logarithmieren

$$\begin{aligned}
Z_2^{(6)} =& 3X_1 + X_3 + 2[X_1, X_2] + [X_1, X_3] - \frac{1}{2}[X_2, X_3] + \frac{1}{6}[X_1, X_1, X_3] - \frac{5}{6}[X_2, X_1, X_2] \\
&- \frac{5}{6}[X_2, X_1, X_3] + \frac{5}{12}[X_1, X_2, X_3] - \frac{1}{3}[X_1, X_1, X_1, X_2] - \frac{1}{6}[X_1, X_1, X_1, X_3].
\end{aligned} \tag{3.109}$$

Wir vermuten, diesen Ausdruck mit 3 Kommutatoren berechnen zu können. Der verwendete Ansatz hat die Gestalt

$$\begin{aligned}
C_1 =& 2[X_1, X_2] + [X_1, X_3] + a_0[X_2, X_3] = [X_1 + a_0 X_2, 2X_2 + X_3] \\
C_2 =& [a_1 X_1 + a_2 X_2 + a_3 X_3, a_4 X_1 + a_5 X_2 + a_6 X_3 + C_1] \\
C_3 =& [a_7 X_1 + a_8 X_2 + a_9 X_3 + a_{10} C_1, a_{11} X_1 + a_{12} X_2 + a_{13} a_{14} X_3 + a_{15} C_1 + C_2] \\
Z_2^{(6)} =& a_{16} X_1 + a_{17} X_2 + a_{18} X_3 + a_{19} C_1 + a_{20} C_2 + a_{21} C_3.
\end{aligned} \tag{3.110}$$

Ein Koeffizientenabgleich ergibt 14 nichttriviale Gleichungen, denn die Gleichung für das Basiselement $[X_1, X_1, X_1, X_1, X_2]$ ist identisch erfüllt, es wird nicht durch 3 Kommutatoren generiert. Wir setzen (nach einigem Probieren) $a_3 = a_{10} = a_{15} = 0$ und erhalten 2 Lösungsscharen. Von diesen

wählen wir die erste, setzen freie Variable zu Null, und erhalten folgende Formel, wobei wir eine Darstellung mit den Argumenten $Y_1 = X_1 - X_2$, $Y_2 = X_1$, $Y_3 = X_1 + Y_2 + Y_3$ der Exponentialfunktionen wählen:

$$\begin{aligned} C_1 =& [5/2Y_1 - 7/2Y_2, Y_1 - Y_3] \\ C_2 =& [Y_1 - 7/5Y_2, C_1 + 13/10Y_2 - 3/2Y_3] \\ C_3 =& [25/24Y_1 - 5/8Y_2, -88/125Y_2 + 36/25Y_3 + C_2] \\ Z_2^{(6)} =& Y_1 + Y_2 + Y_3 + 11/40C_1 + 5/24C_2 + C_3 \end{aligned} \tag{3.111}$$

Ordnung 8 für 3 Kommutatoren Wir greifen wieder auf Lemma 3.2.6 zurück. Der Ansatz $Y_1 = X_1 - X_2$, $Y_2 = X_1$, $Y_3 = X_1 + X_2 + X_3$ führt auf eine bewertete freie Lie-Algebra mit 3 Atomen der Ordnung $1, 2, 3$. Die Produktbasis besteht hier aus 176 Elementen, die Lie-Algebra besitzt Dimension 40. Die Approximation Z_3^8 ergibt sich zu

$$\begin{aligned} Z_3^8 =& 3X_1 + X_3 + 2[X_1, X_2] + [X_1, X_3] - 1/2[X_2, X_3] \\ &+ 1/6[X_1, X_1, X_3] + 5/12[X_1, X_2, X_3] - 5/6[X_2, X_1, X_2] - 5/6[X_2, X_1, X_3] \\ &+ 1/6[X_2, X_2, X_3] - 1/6[X_3, X_1, X_3] + 1/12[X_3, X_2, X_3] \\ &- 1/3[X_1, X_1, X_1, X_2] - 1/6[X_1, X_1, X_1, X_3] - 7/24[X_1, X_2, X_1, X_3] \\ &+ 1/8[X_1, X_2, X_2, X_3] - 1/6[X_1, X_3, X_1, X_3] + 1/3[X_1, X_1, X_2, X_3] \\ &- 1/24[X_2, X_1, X_1, X_3] - 3/8[X_2, X_1, X_2, X_3] + 1/4[X_2, X_2, X_1, X_2] \\ &+ 3/8[X_2, X_2, X_1, X_3] - 1/24[X_1, X_1, X_1, X_1, X_3] - 31/80[X_1, X_2, X_1, X_1, X_3] \\ &- 23/240[X_1, X_1, X_1, X_2, X_3] - 1/15[X_1, X_1, X_2, X_1, X_2] \\ &+ 77/240[X_1, X_1, X_2, X_1, X_3] + 31/120[X_2, X_1, X_1, X_1, X_2] \\ &+ 31/120[X_2, X_1, X_1, X_1, X_3] + 13/180[X_1, X_1, X_1, X_1, X_1, X_2] \\ &+ 13/360[X_1, X_1, X_1, X_1, X_1, X_3]. \end{aligned} \tag{3.112}$$

Man erkennt, daß mindestens 5 Kommutatoren notwendig sind. Die Struktur der Kommutatoren mit 6 Termen fügen wir in den Ansatz ein (Koeffizienten vor X_1 in $C_1, \ldots, C_5$, sowie $a_{40} = 13/360$), außerdem verwenden wir einen sechsten Kommutator. Unser Ansatz lautet somit

$$\begin{aligned} C_1 =& 2[X_1, X_2] + [X_1, X_3] + a_0[X_2, X_3] = [X_1 + a_0 X_2, 2X_2 + X_3] \\ C_2 =& [X_1 + a_1 X_2 + a_2 X_3, a_3 X_1 + a_4 X_2 + a_5 X_3 + C_1] \\ C_3 =& [X_1 + a_6 X_2 + a_7 X_3 + a_8 C_1, a_9 X_1 + a_{10} X_2 + a_{11} X_3 + a_{12} C_1 + C_2] \\ C_4 =& [X_1 + a_{13} X_2 + a_{14} X_3 + a_{15} C_1 + a_{16} C_2, \\ & a_{17} X_1 + a_{18} X_2 + a_{19} X_3 + a_{20} C_1 + a_{21} C_2 + C_3] \\ C_5 =& [X_1 + a_{22} X_2 + a_{23} + X_3 + a_{24} C_1 + a_{25} C_2 + a_{26} C_3, \\ & a_{27} X_1 + a_{28} X_2 + a_{29} X_3 + a_{30} C_1 + a_{31} C_2 + a_{32} C_3 + C_4] \\ \tilde{C}_3 =& [a_{41} X_1 + a_{42} X_2 + a_{43} X_3 + a_{44} C_1 + a_{45} C_2, \\ & a_{46} X_1 + a_{47} X_2 + a_{48} X_3 + a_{49} C_1 + a_{50} C_2 + a_{51} C_3] \\ Z_3^{(8)} =& a_{33} X_1 + a_{34} X_2 + a_{35} X_3 + a_{36} C_1 + a_{37} C_2 + a_{38} C_3 + a_{39} C_4 + a_{40} C_5 + \tilde{C}_3. \end{aligned} \tag{3.113}$$

Dies sind zunächst 40 Gleichungen für 52 Variable, wobei die 3 Gleichungen für X_1, X_2, X_3 sofort a_{33}, a_{34}, a_{35} ergeben. Die Gleichung für den Kommutator mit 7 Termen ist identisch erfüllt, und durch

die Wahl $a_{40} = 13/360$ fallen zwei weitere Gleichungen weg. Es verbleiben 34 Gleichungen für 48 Variable.

Der Lösungsprozeß gestaltet sich sehr schwierig, nach längerer Suche erhält man schließlich folgende Parameter

$$
\begin{aligned}
a_0 =& -\left(\frac{1373}{2912}\right), \quad a_1 = \frac{\sqrt{\frac{410591}{14}}}{130}, \quad a_2 = \frac{6865 + 8 \quad \sqrt{5748274}}{29120},\\
a_5 =& -\left(\frac{15}{13}\right), \quad a_6 = \frac{-\sqrt{\frac{410591}{14}}}{130}, \quad a_7 = \frac{-731937 - 104 \quad \sqrt{5748274}}{378560},\\
a_{10} =& \frac{167700}{410591}, \quad a_{11} = \frac{150 \quad \left(7267 - 6 \quad \sqrt{5748274}\right)}{5337683}, \quad a_{12} = 60 \quad \sqrt{\frac{14}{410591}},\\
a_{14} =& \frac{42633}{209248}, \quad a_{19} = \frac{4980}{42029}, \quad a_{20} = -\left(\frac{4922910}{5337683}\right),\\
a_{21} =& -60 \quad \sqrt{\frac{14}{410591}}, \quad a_{23} = \frac{329}{416}, \quad a_{28} = \frac{5783680982160}{563493352279},\\
a_{29} =& \frac{360 \quad \left(8032890253 - 3200400 \quad \sqrt{5748274}\right)}{563493352279}, \quad a_{30} = \frac{76809600 \quad \sqrt{\frac{1778}{3233}}}{13407251},\\
a_{31} =& -\left(\frac{3200400}{1031327}\right), \quad a_{33} = 3, \quad a_{35} = 1, \quad a_{36} = \frac{4516957761960}{5504896595341},\\
a_{37} =& \frac{-79380 \quad \sqrt{\frac{14}{410591}}}{3233}, \quad a_{38} = -\left(\frac{29575}{1031327}\right), \quad a_{42} = \frac{31}{240},\\
a_{43} =& \frac{31}{480}, \quad a_{49} = -\left(\frac{109278141590}{52784347187}\right), \quad a_{50} = \frac{1342140 \quad \sqrt{\frac{14}{410591}}}{9889}, \quad a_{51} = 1
\end{aligned}
\tag{3.114}
$$

Ordnung 8 für 4 Kommutatoren Da das vierte Atom bereits den Grad 4 besitzt, erhöht sich die Dimension der Produktbasis (223) und auch der Lie-Algebra (55) nur moderat. Trotzdem war es bisher nicht möglich, eine Lösung mit 6 Kommutatoren zu finden. Eine Erweiterung des Ansatz auf eine höhere Anzahl von Kommutatoren ist zwar möglich, aber das System besitzt dann sehr viele Freiheitsgrade, die erst wieder durch geschickte Heuristiken reduziert werden müssen. Wir wählen daher pragmatisch folgende Lösung:

$$
Z_4^{(8)}(Y_1, Y_2, Y_3, Y_4) := Z_2^{(8)}(Z_2^{(8)}(Y_1, Y_2), Z_2^{(8)}(Y_3, Y_4)). \tag{3.115}
$$

Zur Auswertung sind 15 Kommutatoren notwendig, während man typischerweise mit 6 bis 7 Kommutatoren auskommen würde. Der Gesamtaufwand an Kommutatoren erhöht sich damit moderat von $5 + 6 + 7 = 18$ auf $5 + 6 + 15 = 26$, was bei 14 Matrixexponentialfunktionen nicht ins Gewicht fällt.

Übersicht

Wir stellen den benötigten Aufwand an Funktionsaufrufen, Matrixexponentialfunktionen und Kommutatoren für Ordnung $p = 2, 4, 6, 8$ dar. Am effizientesten erscheint die doppelte harmonische Folge als Schrittweitensequenz. Zum Vergleich führen wir die RK-MK-Methoden der Ordnung $2, 4$ an.

Beim Zählen der Funktionsauswertungen berücksichtigen wir, daß der erste Funktionsaufruf $f(y_n)$ mehrfach verwendet werden kann, somit verbleiben jeweils $1, 3, 5, 7$ Funktionsaufrufe für die Berechnungen mit $2, 4, 6, 8$ Teilschritten. In der Summe ergibt sich dann gerade $p^2 + 1$ für Ordnung $2p$.

Bei der Berechnung der Exponentialfunktion können die Auswertungen $\exp(h/n_i f(y_n))$ auf eine Exponentialfunktion $\exp(h/k f(y_n))$ mit $k = \text{k.g.V.}(n_1, n_2, \dots)$ zurückgeführt werden. Es verbleiben dann $0, 2, 4, 6$ Exponentialfunktionen für die Schritte mit $n_i = 2, 4, 6, 8$ Teilschritten, womit sich für Ordnung $2p$ gerade $p(p-1) + 2$ Exponentialfunktionen ergeben.

An Kommutatoren benötigt man für Ordnung 4 genau einen, für Ordnung 6 zu jeweils 3 Kommutatoren für 2 bzw. 3 Exponentialterme, und für Ordnung 8 benötigt man $5, 6, 15$ für jeweils $2, 3, 4$ Exponentialterme.

	EXP			RK-MK		
p	$\#f$	$\#\exp$	$\#[\,]$	$\#f$	$\#\exp$	$\#[\,]$
2	2	2	0	2	2	0
4	5	4	1	4	4	2
6	10	8	6			
8	17	14	26			

(3.116)

Für niedere Ordnung ergibt sich ein ähnlicher Aufwand wie bei den RK-MK-Methoden. Für höhere Ordnung fehlen Vergleichsmöglichkeiten, man verzeichnet einen annähernd quadratischen Anstieg des Aufwands. Gleiches gilt allerdings für den ODE-Fall, wo Extrapolationsverfahren ihre Effizienz bereits unter Beweis gestellt haben.

3.3 Generierung der bewerteten freien Lie-Algebra mit Mathematica und GLIEALG

3.3.1 Einleitung

Die Bestimmung der Gleichungen wird mit wachsender Genauigkeitsordnung in der BCH-Formel immer aufwendiger. Natürlich ist ebenso die Lösung der Gleichungen ab ca. 10 Variablen kaum noch manuell möglich. In einem Formelmanipulationsprogramm läßt sich sowohl die Generierung der Gleichungen als auch die Lösung derselben noch für eine größere Anzahl von Gleichungen durchführen.

Natürlich sind auch hier der Größe der Systeme Grenzen gesetzt. Bei einer Größenordnung von 20 bis 40 Gleichungen ist auch die Leistungsfähigkeit schnellster PCs erschöpft. Im allgemeinen wächst der Aufwand auch exponentiell mit der Zahl der Gleichungen und Variablen, so daß auch ein Ausweichen auf Hochleistungsrechner die Grenze nur wenig verschiebt. Aus Effizienzgründen empfiehlt es sich, kaum länger als 10 Sekunden, in Ausnahmefällen vielleicht mehrere Minuten, auf die Lösung des Systems zu warten. Wird in dieser Zeit keine Lösung berechnet, so ist die Chance sehr groß, daß auch nach Wochen keine Lösung berechnet ist – man formuliert besser das Problem um, reduziert zum Beispiel die Anzahl der Variablen.

3.3.2 Das Paket GLIEALG

Zur automatischen Generierung der Gleichungen wurde im Rahmen dieser Arbeit das Paket GLIEALG für Mathematica [123] entwickelt. Es kann auf der Mathematica-Homepage unter [119] abgerufen werden.

Das Paket stellt die Routine $GFLA$ zur Verfügung, die ihrerseits Objekte aus der Lie-Algebra-Basis und der universellen einhüllenden Algebra erzeugt, die der gesuchten Arithmetik genügen. Durch den Aufruf $GFLA[n, \{n1, n2, ...\}]$ wird eine Basis für die Elemente bis zum Grad n einer bewerteten freien Lie-Algebra erzeugt, die von Basiselementen (Atomen) der Grade $n1, n2, \ldots$ generiert wird. Dies geschieht in mehreren Schritten, zunächst werden alle möglichen Produkte der Basiselemente bis zum Grad n (Produktbasis) generiert, für diese wird dann eine Operationstafel für das Produkt aufgestellt. Dann werden aus den Atomen alle möglichen Klammern bis zum Grad n generiert, wobei diese in der Produktbasis dargestellt werden. Diese Darstellung ermöglicht einen einfachen Test auf lineare Abhängigkeit. Es wird eine Operationstafel für die Klammer aufgestellt, und dann werden die Operationen Produkt bzw. Klammer für die Elemente $X[1], X[2], \ldots$ der Produktbasis bzw. $Y[1], Y[2], \ldots$ der Lie-Algebra-Basis definiert. Eine Sonderrolle spielt dabei die Identität $X[0]$. Zum Schluß werden die Funktionen EXP und LOG definiert, wobei LOG nicht direkt definiert ist, sondern nur über $\log(1 - x)$ unter dem Namen $LOGOMX$ (log **O**ne **M**inus **X**) verfügbar ist. Dies verhindert das versehentliche Logarithmieren von Elementen vom Grad 0, also solchen, die die Identität $X[0]$ enthalten.

Anstelle der Symbole P, B für Produkt und Klammer (Bracket) wäre auch die Verwendung des eingebauten Matrizenprodukts möglich gewesen, doch da dies die Modifikation der Attribute einer eingebauten Funktion erfordert, haben wir davon Abstand genommen.

Die Generierung der Produktbasis

Funktionales Programmieren erlaubt die kompakte Schreibweise komplexer Operationen. Die Produktbasis wird im wesentlichen mit dem Befehl FixedPointList generiert, wobei die Atome und Produkte als Listen positiver ganzer Zahlen dargestellt werden. Iteriert wird im wesentlichen die Funktion

$Outer[Join, alist, \#, 1]\&$ mit dem Startwert alist=Liste der Atome. Elemente von zu hohem Grad werden dabei in jedem Iterationsschritt gelöscht.

```
alist := Transpose[{Range[NATOMS]}];
PBASIS = FixedPointList[
        Select[Flatten[Outer[Join, alist , #, 1],
              1], (pgrade[#] <= MAXDEG) &] &,
        alist] // Flatten[#, 1] & // DeleteCases[#, 1] &;
NPBASIS = Length[PBASIS];
```

Ziel ist eine vektorielle Darstellung der Elemente der Produktalgebra in der Produktbasis. Mit einer Operationstafel für das Produkt in der Algebra der Produkte kann dann das Produkt beliebiger Vektoren effizient implementiert werden. Zuerst wird die symbolische Operationstafel $PTABLE$ generiert, aus dieser dann die vektorielle Version $vecPTABLE$ erzeugt.

Für Algebren mit relativ großer Produktbasis ist aber auch dies noch ineffektiv. Ein Produkt verschwindet ja gerade, wenn die Summe der Grade höher als der Maximalgrad ist, für Faktoren vom Maximalgrad verschwindet zum Beispiel jedes Produkt. Daher verschwindet der überwiegende Teil der Produkte, womit sich die Implementierung als Liste anbietet. Dies geschieht durch PLIST, und mit PLIST wird die Vektorform des Produkts $vecP$ definiert.

```
PTABLE = Table[
  Module[{a}, a = Join[PBASIS[[i]], PBASIS[[j]]];
    If[pgrade[a] > MAXDEG, 0, Position[PBASIS, a][[1, 1]]]
  ],{i, NPBASIS}, {j, NPBASIS}
];
vecPTABLE = Map[unitvector[#, NPBASIS] &, PTABLE, {2}];
PLIST = {};
Do[If[PTABLE[[i, j]] != 0,
     PLIST = Append[PLIST, {PTABLE[[i, j]], i, j}]]
   , {i, NPBASIS}, {j, NPBASIS}];
vecP[vv_List, ww_List] :=
  Plus @@ Map[(unitvector[#[[1]], NPBASIS]*vv[[#[[2]]]]*ww[[
       #[[3]]]]) &, PLIST];
```

Die Generierung der Klammer-Basis

Zur Initialisierung werden die Atome der Basis hinzugefügt. Gesucht sind nun weitere Elemente der Lie-Algebra, die sich durch iterative Anwendung der Klammer ergeben. Dazu wird neben der symbolischen Darstellung der Klammern als iterierte Listen eine vektorielle Darstellung in der Produktbasis erzeugt ($vecBBASIS$).

```
BBASIS = Range[NATOMS];
vecBBASIS = Map[unitvector[#, NPBASIS] &, Range[NATOMS]];
vecB[a_, b_] :=
  Plus @@ Plus @@ ((Outer[Times, a, b] - Outer[Times, b, a])*
         vecPTABLE);
```

Auf der sich dynamisch erweiternden Liste $BBASIS$ wird mit zwei Zeigern operiert. Die Klammer der an den Zeigerpositionen befindlichen Elemente wird als Vektor bezüglich der Produktbasis dargestellt. Ist das Ergebnis linear unabhängig von den bisher ausgewerteten Klammern, wird es der Liste hinzugefügt.

```
slow = 1;
  While[slow < Length[BBASIS],
    slow = slow + 1;
    Do[v = vecB[vecBBASIS[[fast]], vecBBASIS[[slow]]];
       If[NullSpace[Transpose[Append[vecBBASIS, v]]] == {},
          vecBBASIS = Append[vecBBASIS, v];
          BBASIS = Append[BBASIS,{BBASIS[[fast]],BBASIS[[slow]]}];
          DEGLIST = Append[DEGLIST, DEGLIST[[slow]] +
             DEGLIST[[fast]]]; Print["New Element:", Last[BBASIS]]
       ,
          Print["failed:", fast, "+", slow, " of ", Length[BBASIS]];
       ],{fast, slow - 1}
    ]
  ];
NBBASIS = Length[vecBBASIS];
```

Nach erfolgter Generierung der Basis der Lie-Algebra wird nun eine Operationstafel für die Klammer erzeugt. Das Ergebnis wird dabei nicht mehr in der Produktbasis, sondern in der Klammerbasis dargestellt.

```
vecBTABLE =
  Table[LinearSolve[Transpose[vecBBASIS],
        vecB[vecBBASIS[[i]], vecBBASIS[[j]]]]
    , {i, NBBASIS}, {j, NBBASIS}
    ];
```

Die Definition der Arithmetik

In den folgenden Programmzeilen werden zunächst die Produkte für die Elemente $X[1]$, $X[2], \ldots,$ $X[NPBASIS]$ der Produktbasis definiert. Außerdem werden die Eigenschaften der Identität $X[0]$ festgelegt.

```
P[X[i_], X[j_]] := 0 /; i > 0 && j > 0 ;
Do[If[PTABLE[[i, j]] != 0, P[X[i], X[j]] = X[PTABLE[[i, j]]]],
  {i, NPBASIS}, {j, NPBASIS}
  ];
P[y___, X[0], x__] := P[y, x];
P[x__, X[0], y___] := P[x, y];
P[X[0]] := X[0];
```

Um Ausdrücke wie $P[X[1] + 2X[2], X[3] - 4X[4]]$ vereinfachen zu können, wird zusätzlich die Linearität und Homogenität der Produktoperation in allen Argumenten definiert.

```
P[x___, Times[a_?NumberQ, y__], z___] := a P[x, Times[y], z];
P[x___, 0, z___] := 0;
P[x___, y_ + z_, w___ ] := P[x, y, w] + P[x, z, w];
P[x___, y_ - z_, w___ ] := P[x, y, w] - P[x, z, w];
```

Für die Klammer $B[\ ,\ ]$ werden identische Festlegungen getroffen.

Zusätzlich kann mit der Funktion $getAtomForm$ eine Darstellung von Elementen der Lie-Algebra als Formel abgerufen werden. Die Basiselemente werden dabei als iterierte Klammern über die Atome geschrieben.

```
getAtomForm[x_+y_]:=getAtomForm[x]+getAtomForm[y];
getAtomForm[x_-y_]:=getAtomForm[x]-getAtomForm[y];
getAtomForm[x___,a_?NumberQ y_, z___]:=a getAtomForm[x,y,z];
Do[getAtomForm[Y[i]]=StringReplace[ToString[StringForm[
           "``",BBASIS[[i]]]], {"{"->"[","}"->"]"}];
   DEGREE[Y[i]]=DEGLIST[[i]]
     ,{i,NBBASIS}
];
```

Die Funktionen EXP und LOG

Die Funktionen EXP und $LOGOMX$ werden einfach über die Potenzreihen implementiert. Für EXP wird das Argument (aus der Lie-Algebra) in die Produktform transformiert, und dann mit der bereits definierten Produktoperation die Potenzreihe ausgewertet.

```
EXP[x_] := Module[{y},
    y = Y2X[x];
    X[0] + Plus @@
        FoldList[Simplify[P[y, #1]]/#2 &, y, Range[2, MAXDEG]]
    ];
```

Die Konvertierung $Y2X$ erfolgt, indem man $Y2X$ für die Basis definiert und dann wieder Linearität und Homogenität fordert.

Die Funktion $LOGOMX(X)$ liefert $\log(X[0] - X)$. Sie bildet von der Produkt-Algebra in die Lie-Algebra ab. Die Potenzreihe wird in der Produktalgebra ausgewertet.

```
LOGOMX[x_] :=
  X2Y[-Plus @@ (NestList[(Print[Length[#]];P[x, #]) &, x,
          MAXDEG - 1]/Range[MAXDEG])];
```

Das Ergebnis der Potenzreihe liegt in der Produktalgebra, es wird mit $X2Y$ in der Lie-Algebra dargestellt. Dies erfordert die Auflösung eines linearen Gleichungssystems mit der Koeffizientenmatrix $vecBBASIS$.

3.3.3 Generierung der Gleichungen mit GLIEALG

Wir illustrieren dies kurz am Beispiel der Berechnungen für 3 Exponentialterme mit Ordnung 6. Nach dem Laden von `GLIEALG` steht mit dem Befehl `GFLA(6,1,2,3)` eine Basis der Lie-Algebra mit Elementen bis zur Ordnung 6 für drei Atome der Ordnung 1, 2 bzw. 3 zur Verfügung (15 Basiselemente).

Die BCH-Formel wird durch

```
LOGOMX[X[0]-P[EXP[Y[1]-Y[2]], EXP[Y[1]], EXP[Y[1]+Y[2]+Y[3]]]]
```

generiert. Der Ansatz (3.110) wird dann expandiert und ein Koeffizientenvergleich bezüglich der Basiselemente `Y[1]` ,..., `Y[15]` durchgeführt. Zu beachten ist, daß den Parametern im Ansatz das Attribut $Number$ zuzuordnen ist, damit sie aus den Kommutatoren ausgeklammert werden können und somit alle Kommutatoren auf die 15 Basiselemente zurückgeführt werden können.

3.4 Numerische Tests

3.4.1 Ordnungstest

Trotz theoretischen Nachweises der klassischen Konvergenzordnung sind erfolgreiche Ordnungstests Voraussetzung für die Anwendbarkeit eines Verfahrens. Für ein praktisch verwendbares Verfahren muß die prognostizierte Ordnung im relevanten Schrittweitenbereich nachweisbar sein – als warnendes Beispiel dient die Ordnungsreduktion bei steifen Differentialgleichungen.

Zum Test wählen wir eine einfache Differentialgleichung in $GL(5)$, siehe [14]. In matlab-Notation lautet sie

$$X' = A(X)X = \frac{1}{2}(triu(X,1) - tril(X,-1))X. \tag{3.117}$$

Mit $triu, tril$ extrahiert man obere und untere (Teil-) Dreiecksmatrizen, hier extrahiert $triu(X,1)$ alle Elemente oberhalb der Hauptdiagonalen, und $tril(X,-1)$ alle Elemente unterhalb der Hauptdiagonalen.

Wir betrachten ein Anfangswertproblem im Intervall $t \in [0,1]$. Der Anfangswert ist in matlab-Notation durch

$$X(0) = \texttt{-2*eye(5)+diag(ones(4,1),1)+diag(ones(4,1),-1)} \tag{3.118}$$

gegeben. Mit DOPRI5 und kleinstmöglichen Toleranzen wird eine Vergleichslösung im Endpunkt $t_e = 1$ berechnet.

Zum Ordnungstest lösen wir das Problem mit konstanten Schrittweiten $h_i = 1/N_i$ für verschiedene Schrittzahlen N_i für die Extrapolationsmethoden der Ordnung $2, 4, 6, 8$. Zu den Schrittweiten h_i, h_{i+1} ergeben sich globale Fehler e_i im Endpunkt (gemessen in $\|\cdot\|_1$). Damit kann die Ordnung des Verfahrens durch

$$p \approx \frac{\log e_{i+1} - \log e_i}{\log h_{i+1} - \log h_i} \tag{3.119}$$

geschätzt werden. Für alle 3 Verfahren zeigt sich in Abbildung 3.1 eine sehr gute Übereinstimmung mit den theoretischen Vorhersagen. Hier ist insbesondere von Interesse, ob die relativ teuren Matrixexponentialfunktionen eine Verwendung von Methoden höherer Ordnung zulassen. In Abbildung 3.2 ist die Genauigkeit im Endpunkt gegenüber der Anzahl von Exponentialfunktionen dargestellt. Es zeigt sich in diesem Kosten-Aufwands-Diagramm, daß die Methoden höherer Ordnung im relevanten Schrittweitenbereich stets vorzuziehen sind.

3.4.2 Der Toda-Lattice

Im numerischen Experiment betrachten wir den Toda-Lattice. Für weitere Anwendungen von Lie-Gruppen-Integratoren verweisen wir auf [31], [68]. Wir wählen $n = 3$ und periodische Randbedin-

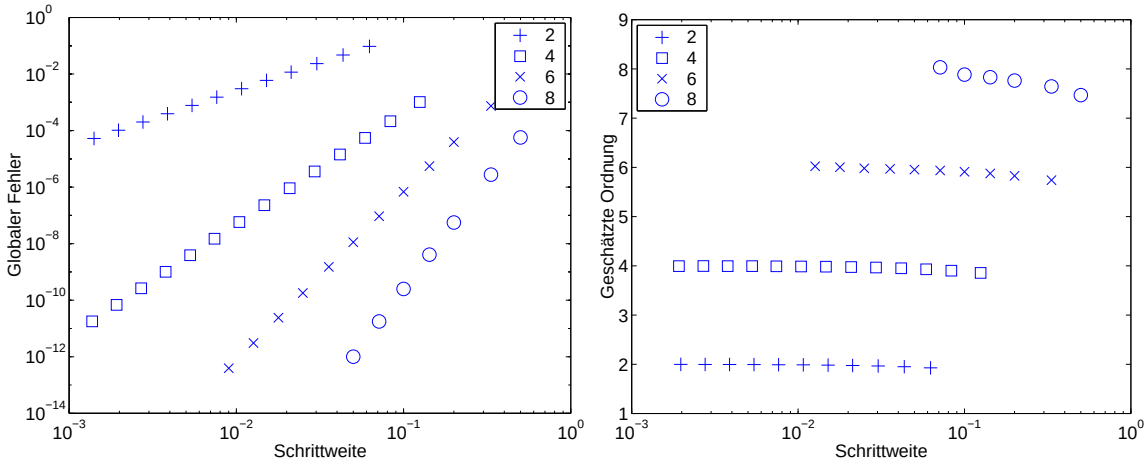


Abbildung 3.1: Globaler Fehler bei konstanter Schrittweite und geschätzte Ordnung

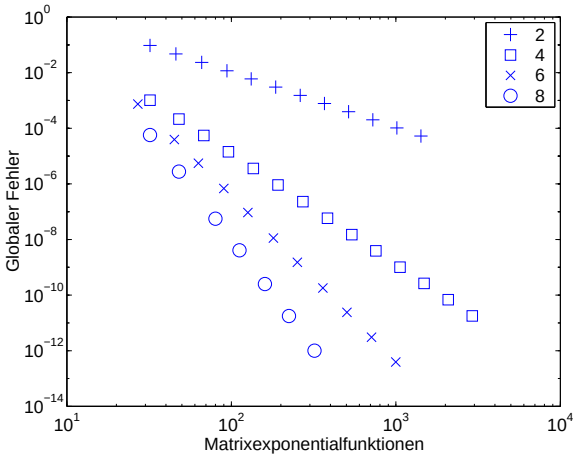


Abbildung 3.2: Fehler in Abhängigkeit der Anzahl der Matrixexponentialfunktionen

gungen, mit Startwerten wie in [47]

$$\begin{aligned} p_1 &= -1.5, & p_2 &= 1, & p_3 &= 0.5, \\ q_1 &= 1, & q_2 &= 2, & q_3 &= -1. \end{aligned}$$

Die Gleichungen ergeben sich zu $L' = [B(L), L]$ mit

$$L = \begin{pmatrix} a_1 & b_1 & b_3 \\ b_1 & a_2 & b_2 \\ b_3 & b_2 & a_3 \end{pmatrix}, \quad B(L) = \begin{pmatrix} 0 & l_{12} & -l_{31} \\ -l_{12} & 0 & l_{23} \\ l_{31} & -l_{23} & 0 \end{pmatrix} \tag{3.120}$$

wobei $a_i = -1/2p_i$, $b_i = 1/2\exp(1/2(q_k - q_{k+1}))$.

Die Exponentialabbildung berechnen wir über die Formel von Rodriguez, man siehe Abschnitt 1.3.7.

Formulierungen für ODE-Methoden zum Vergleich

Neben der numerischen Lösung der Toda-Gleichung mit Lie-Gruppen-Methoden (Bezeichnung: **Geometric**) betrachten wir folgende Formulierungen, auf die ODE-Methoden angewendet werden:

Hamilton Die Originalformulierung mit den Variablen p_k, q_k. Es handelt sich um ein Hamiltonsystem mit der Hamiltonfunktion

$$H = \sum_{k=1}^{3} \frac{1}{2} p_k^2 + \exp(q_k - q_{k+1}), \tag{3.121}$$

welches auf die Differentialgleichungen

$$p_k' = -\exp(q_k - q_{k+1}) + \exp(q_{k-1} - q_k), \quad q_k' = p_k, \quad k = 1, \ldots, 3. \tag{3.122}$$

führt.

Bracket Die Klammer-Gleichung (3.120) für die transformierten Variablen a_k, b_k

$$a_k' = 2(b_k^2 - b_{k-1}^2) \qquad b_k' = b_k(a_{k+1} - a_k). \tag{3.123}$$

Lie Die Lie-Gruppen-Formulierung $T' = B(L(T))T$.

Die exakte Lösung auf $[0, 1]$ sowie $[0, 10]$ wird in den Abbildungen 3.3 und 3.4 gezeigt. Im jeweils linken Bild sind die Komponenten a_1, a_2, a_3 zu sehen, im rechten die Komponenten b_1, b_2, b_3. In $[0, 1]$

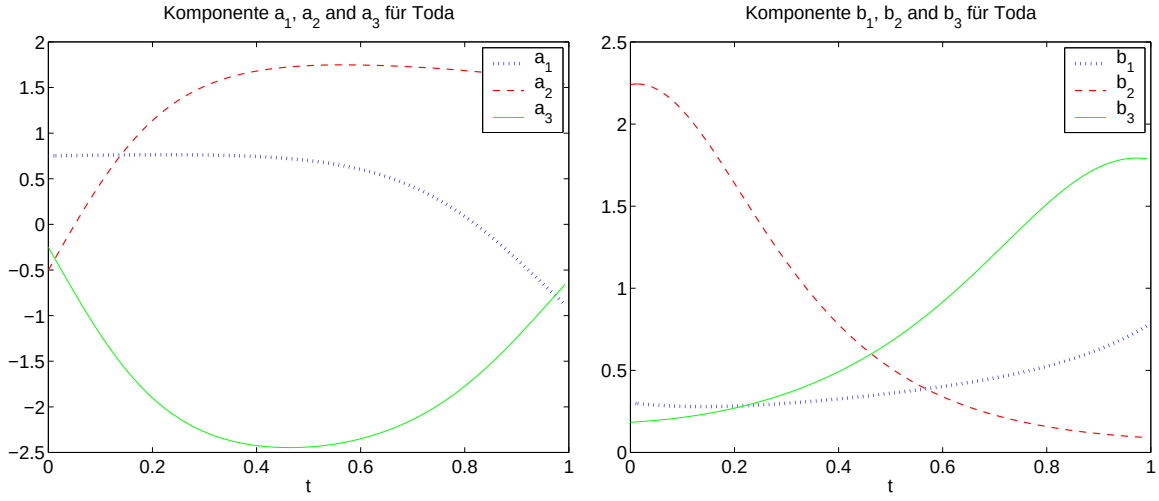


Abbildung 3.3: Lösung des Toda-Lattice auf $[0, 1]$.

ist die Lösung verhältnismäßig glatt, während in $[0, 10]$ eine quasi-Periodizität ausgemacht werden kann. Die Eigenwerte der Matrix L sind invariant, so daß man $a_1 + a_2 + a_3$ als Invariante erkennt.

Ergebnisse mit konstanter Schrittweite

Wir vergleichen hier die Lie-Gruppen-Extrapolationsverfahren als geometrische Integrationsmethode mit der klassischen Extrapolationsmethode gleicher Ordnung. Die nahezu exakte Vergleichslösung

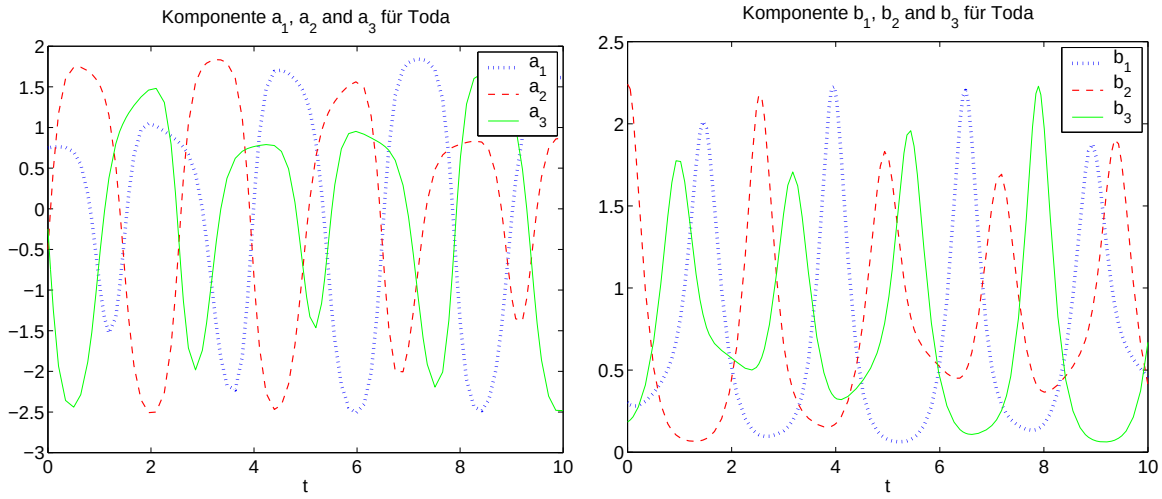


Abbildung 3.4: Lösung des Toda-Lattice auf $[0, 10]$.

am Endpunkt wurde mit DOPRI5 mit einer relativen Toleranz von 10^{-12} berechnet, betrachtet wird der globale Fehler im Endpunkt t_e. Wir haben das System für zwei verschiedene Endpunkte gelöst, einmal für $t_e = 1$ sowie für $t_e = 10$. Das klassische Extrapolationsverfahren wurde auf alle 3 ODE-Formulierungen angewandt, dargestellt sind **Hamilton** mit +, **Bracket** mit x und **Lie** mit einem Quadrat □. Die Lösung durch das Lie-Gruppen-Extrapolationsverfahren ist mit einem Kreis ∘ dargestellt.

Es kristallisiert sich heraus, daß von allen drei Formulierungen die Lie-Gruppen-Formulierung die günstigste ist. Für diese selbst schneiden das klassische Verfahren und die Lie-Gruppen-Extrapolationsmethode bei Ordnung 4 und 6 annähernd gleich gut ab, bei den Verfahren der Ordnung 8 ist die Lie-Gruppen-Extrapolationsmethode um den Faktor 10 genauer. Für längere Integrationsintervalle ($[0, 50]$) ergibt sich aus Abbildung 3.8, daß der Lie-Gruppen-Integrator am besten abschneidet.

In Abbildung 3.9 haben wir den Fehler in der Orthogonalität der Matrix T dargestellt. Die Resultate für den geometrischen Integrator sind durch Kreise dargestellt, während für das klassische Extrapolation-Verfahren, angewandt auf die Formulierung **Lie** , Quadrate verwendet wurden. Für die Intervalle $[0, 1]$ sowie $[0, 10]$ sind jeweils die Verfahren der Ordnung 4, 6 und 8 dargestellt, während für das Intervall $[0, 50]$ (unteres Bild) die Ergebnisse für die Verfahren der Ordnung 6 und 8 verglichen werden.

Wie erwartet bewegt sich der Fehler bei den klassischen Integratoren im Rahmen des globalen Fehlers, während der geometrische Integrator die Nebenbedingung im Rahmen der Rundungsfehlergenauigkeit erfüllt.

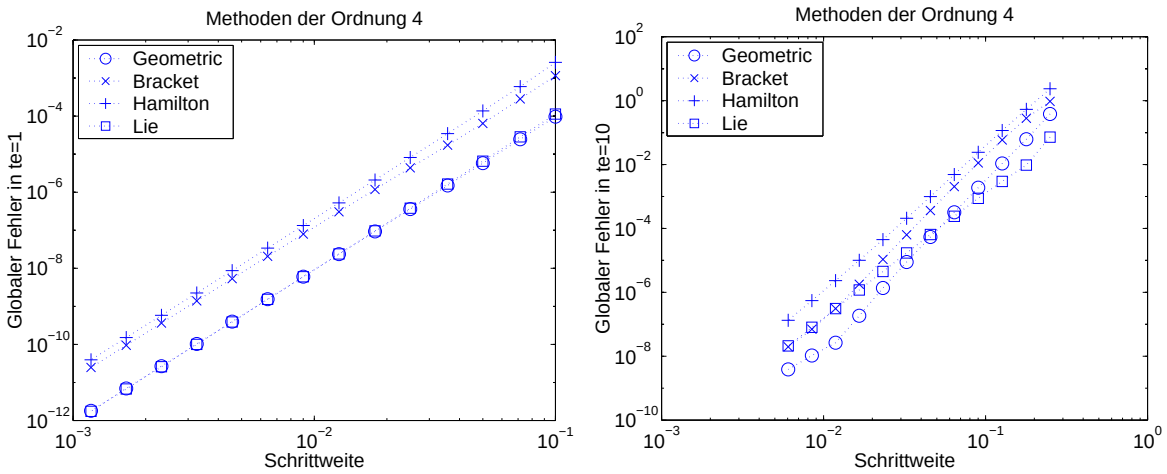


Abbildung 3.5: Globaler Fehler für Toda-Lattice bei konstanter Schrittweite für Methoden der Ordnung 4 in $[0, 1]$ sowie $[0, 10]$.

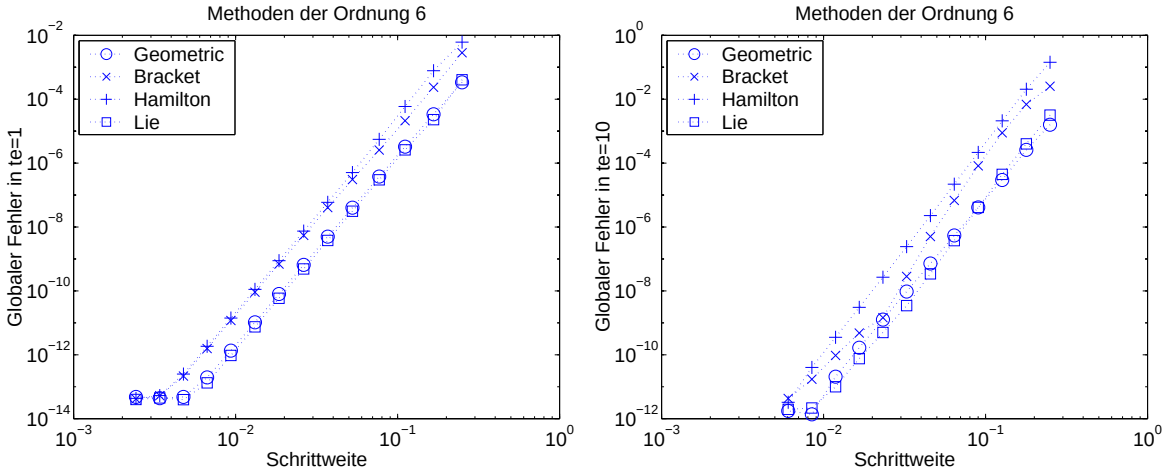


Abbildung 3.6: Globaler Fehler für Toda-Lattice bei konstanter Schrittweite für Methoden der Ordnung 6 in $[0, 1]$ sowie $[0, 10]$.

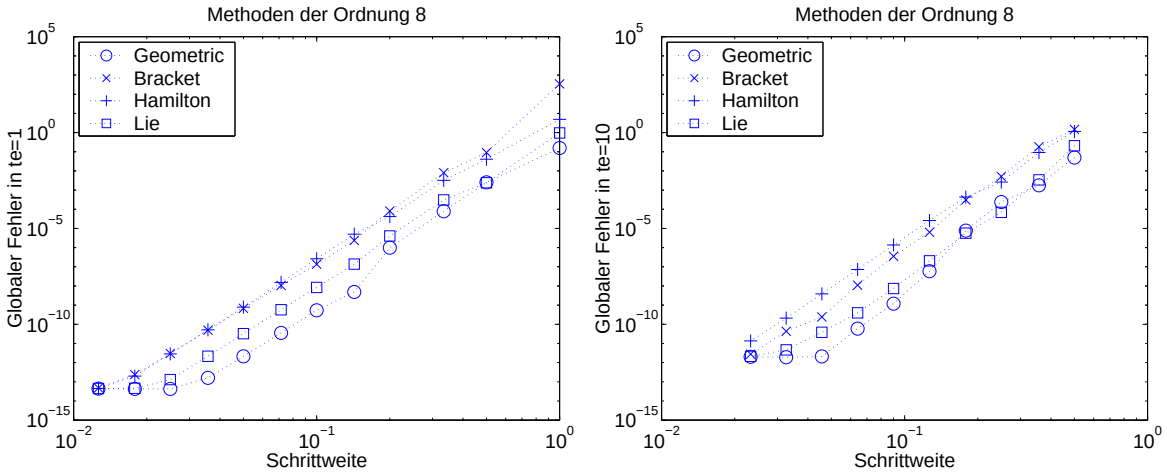


Abbildung 3.7: Globaler Fehler für Toda-Lattice bei konstanter Schrittweite für Methoden der Ordnung 8 in $[0, 1]$ sowie $[0, 10]$.

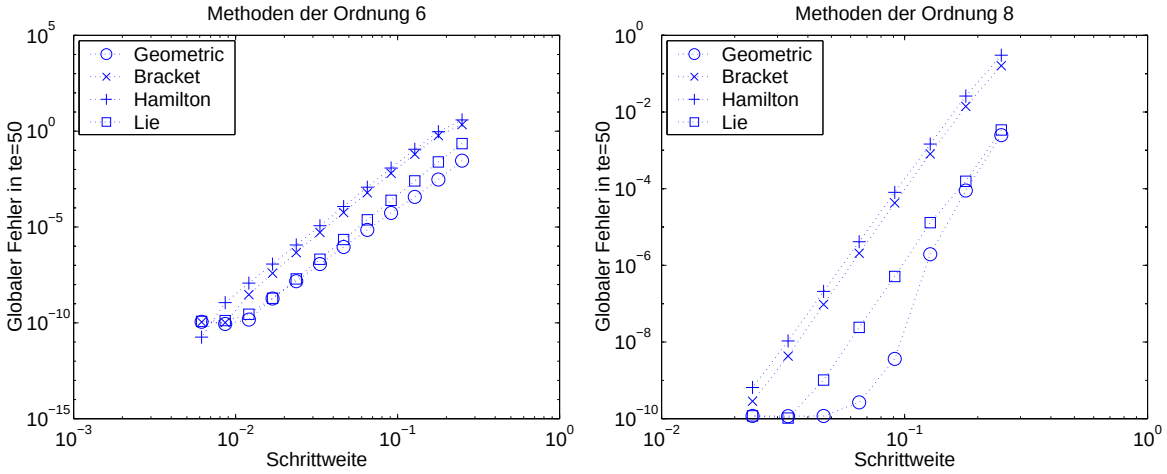


Abbildung 3.8: Globaler Fehler für Toda-Lattice für Methoden der Ordnung 6 und 8 im Intervall $[0, 50]$.

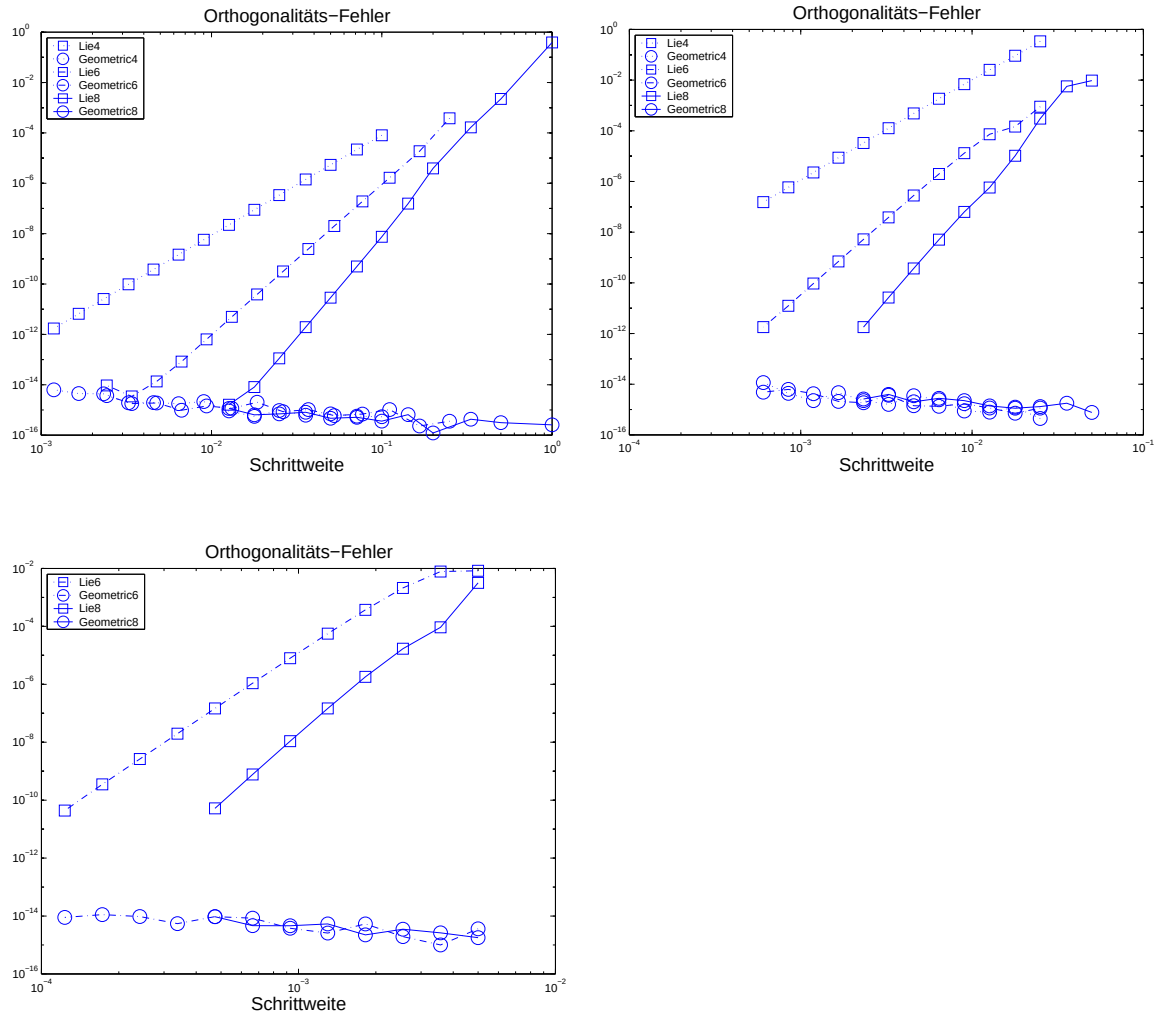


Abbildung 3.9: Abstand zur nächsten orthogonalen Matrix für die Lösung im Endpunkt $t_e = 1$, $t_e = 10$ sowie $t_e = 50$.

3.5 Zusammenfassung

Dieses Kapitel stellt die wichtigsten theoretischen Beiträge des Verfassers zur numerischen geometrischen Integration dar. Wir legen das theoretische Gerüst zur Entwicklung von Extrapolationsverfahren auf Lie-Gruppen im Geiste des Gragg-Bulirsch-Stoer-Verfahrens. Zum einen wird die Extrapolationsidee auf Lie-Gruppen übertragen, zum anderen wird eine quadratische Entwicklung für symmetrische Verfahren nachgewiesen. Auf dieser Basis werden Extrapolationsverfahren – sowohl für einfache wie für quadratische asymptotische Entwicklung – konstruiert. Im Gegensatz zum ODE-Fall sind hier zusätzlich Approximationen an die BCH-Formel herzuleiten. Während für kleine Approximationsordnungen eine analytische Auflösung der Ordnungsbedingungen noch möglich ist, ist für die höchste betrachtete Ordnung – Ordnung 8 – dann selbst die Ableitung der Ordnungsbedingungen kaum mehr mit vertretbarem Aufwand möglich. Davon motiviert, wurde das Paket GLIEALG für das Programm Mathematica entwickelt, mit dem Berechnungen in einer Lie-Algebra und der generierten Lie-Gruppe einfach möglich sind. So konnten erstmals Extrapolationsvorschriften bis zur Ordnung 8 hergeleitet werden, dies erforderte unter anderem Approximationen an die BCH-Formel mit bis zu 4 Faktoren. Der erste Teil der Arbeit schließt mit numerischen Tests der entwickelten geometrischen Integratoren. Im zweiten Teil werden wir sowohl geometrische Integratoren als auch klassische, problemangepaßte Zeitintegrationsverfahren in praxisnahen Anwendungsbeispielen im Einsatz sehen.

Teil II

Beiträge zur numerischen Simulation in den Naturwissenschaften

Kapitel 4

Magnus-Methoden zur Lösung der stationären Schrödinger-Gleichung

Wir stellen hier einen Algorithmus zur Lösung der stationären Schrödinger-Gleichung vor. Als Anwendung bestimmen wir die Energieeigenwerte des Wasserstoffatoms. Denkbar ist aber auch der Einsatz für komplexere Problemstellungen aus der Kernphysik im Rahmen der Dichtefunktionaltheorie.

Eine Darstellung der Schrödinger-Gleichung in Kugelkoordinaten führt nach Abspaltung des Drehimpulsoperators durch einen Ansatz mit Kugelflächenfunktionen auf eine lineare Differentialgleichung in einer Variablen, wo die Koeffizientenmatrix von der unabhängigen Variablen abhängt. Die Lösungen sind zum Teil stark oszillierende Funktionen.

Gesucht sind dabei die möglichen Energien der Elektronen, was auf Eigenwertprobleme für die Schrödinger- und die Dirac-Gleichung führt. Probleme dieser Art wurden für reguläre Randbedingungen in [60] betrachtet, dort wurden ebenso singuläre Randbedingungen für Randwertprobleme betrachtet. Hier liegt eine Kombination von beidem vor – ein Eigenwertproblem mit singulären Randbedingungen.

Zur Lösung solcher Gleichungen nutzen wir mit der Magnus-Methode ein Verfahren aus der Klasse der geometrischen Integratoren. Dieses Verfahren ist auf lineare Differentialgleichungen anwendbar und besonders effektiv für stark oszillierende Lösungen. Es basiert auf einer Reihenentwicklung für die exakte Lösung von linearen Differentialgleichungen mit zeitabhängigen Koeffizienten, die von W. Magnus [66] gefunden wurde. Darauf basierende Methoden wurden von verschiedenen Autoren entwickelt, man siehe [62, 18, 19, 65, 56, 57, 58].

Wie im allgemeinen bei abgeleiteten Problemstellungen ist die effiziente Lösung der zugrundeliegenden Differentialgleichung von entscheidender Bedeutung.

4.1 Grundlagen der Quantenmechanik

4.1.1 Welle-Teilchen-Dualismus

Bis zum Jahre 1905 waren die Objekte der Physik in zwei Klassen eingeteilt – es gab Objekte mit Wellennatur (Licht, Elektromagnetismus, Wärmestrahlung) und Teilchennatur (feste Materie).

Der photoelektrische Effekt

Beim Auftreffen von Licht auf bestimmte Metalloberflächen werden Elektronen emittiert. Experimentell läßt sich für die kinetische Energie dieser Elektronen eine lineare Abhängigkeit von der Frequenz

f des Lichts feststellen

$$E_{kin} = hf - W_0. \tag{4.1}$$

Mit der Wellennatur des Lichts ist ein solcher Zusammenhang allerdings nicht in Übereinstimmung zu bringen. Einstein gab 1905 eine Erklärung des photoelektrischen Effekts, die die reine Wellennatur des Lichts in Frage stellte und Lichtquanten als Träger der Energie postulierte. Die Energie eines solchen Photons ist proportional zur Frequenz, der Proportionalitätsfaktor ist das Plancksche Wirkungsquantum h. Sieht man W_0 als die Energie, die zum Herauslösen des Elektrons aus dem Atom nötig ist, an, so ergibt sich für die kinetische Energie des Elektrons gerade (4.1), wenn das Photon komplett vom Elektron absorbiert wird und dabei seine gesamte Energie auf das Elektron überträgt.

Wellencharakter der Materie

Auf der anderen Seite war der Wellencharakter des Lichts durch des Youngsche Doppelspaltexperiment hinreichend belegt. Der Widerspruch wurde zunächst gelöst, indem man vom Welle-Teilchen-Dualismus des Lichts sprach.

Hinzu kam im Jahre 1927 allerdings ein weiterer Widerspruch: An Kristallgittern konnten Interferenzerscheinungen von Elektronenstrahlen nachgewiesen werden. De Broglie hatte bereits im Jahre 1923 in seiner Dissertation postuliert, daß auch Elektronen ein Interferenzmuster beim Doppelspaltversuch erzeugen müßten.

Der scheinbar unlösbare Widerspruch der gleichzeitigen Wellen- und Teilchennatur der Materie konnte erst durch die Quantenmechanik befriedigend gelöst werden.

4.1.2 Quantenmechanik

Der Doppelspaltversuch

Der Doppelspaltversuch offenbart ein weiteres Paradoxon: Eine Wand mit Spalten wird mit Elektronen beschossen, die Intensität wird an einem dahinter aufgestellten Schirm gemessen. Selbst bei zeitlich diskretem Beschuß mit “Einzel-Elektronen” stellt man Interferenzbilder fest – die Teilchen verhalten sich wie Wellen. Wird ein Spalt abgedeckt, so passiert das Elektron stets den anderen Spalt, es entsteht natürlich kein Interferenzbild. Etwas ähnliches passiert allerdings auch, wenn man (bei zwei offenen Spalten) mit einem geeigneten Detektor versucht zu messen, welchen Spalt das Elektron passiert hat: Man kann nachweisen, daß das Elektron genau einen Spalt passiert hat – und in diesem Fall entsteht dann auch kein Interferenzbild auf dem Schirm! Der Ausgang des Versuchs hängt quasi davon ab, ob wir bestimmte Zwischenzustände beobachten oder nicht. Beobachten wir das Elektron beim Durchgang durch die Wand, verhält es sich wie ein Teilchen, beobachten wir es nicht, so verhält es sich wie eine Welle.

Die Wellenfunktion

Bei der Beschreibung von Ereignissen auf quantenmechanischer Ebene wird jedem möglichen Zustand eine Eintrittswahrscheinlichkeit zugeordnet. Die Dichtefunktion dieser Wahrscheinlichkeit ist das Betragsquadrat der komplexen Wellenfunktion, $P = |\Psi|^2$. Die Wellenfunktion Ψ selbst ist eine Funktion der Freiheitsgrade des Systems.

So wird ein Ein-Teilchen-System durch $\Psi(\mathbf{r})$, $\mathbf{r} \in \mathbb{R}^3$ beschrieben, ein N-Teilchen-System durch $\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_N)$. Im klassischen Ein-Teilchen-System wäre die Wellenfunktion gerade die Diracsche δ-Funktion.

Dieses Modell erlaubt es, Elementarteilchen sowohl mit Welleneigenschaften als auch mit Teilcheneigenschaften auszustatten. Beim Teilchenverhalten addieren sich die Eintrittswahrscheinlichkeiten, beim Wellenverhalten die Wellenfunktionen.

Das Unbestimmtheitsprinzip

Welche Eigenschaft – Teilchen oder Welle – wirksam wird, ist im Unbestimmtheitsprinzip von Heisenberg formuliert worden: Kann ein Ereignis auf zwei verschiedene (unvereinbare) Weisen mit Wellenfunktionen Ψ_1 und Ψ_2 zustande kommen, so addieren sich die Wellenfunktionen (wellenartig), das Ereignis besitzt die Wellenfunktion $\Psi_1 + \Psi_2$.

Wird jedoch mit einer geeigneten Apparatur festgestellt, welches Teilereignis eintritt, so addieren sich die Wahrscheinlichkeiten, das Ereignis tritt mit Wahrscheinlichkeit $P = |\Psi_1|^2 + |\Psi_2|^2$ ein.

Genauer quantifiziert wird dies in der Heisenbergschen Unschärferelation, die für die Ungenauigkeit Δq im Ort bzw. Δp im Impuls die Beziehung

$$\Delta p \Delta q \geq \hbar/2 \tag{4.2}$$

mit $\hbar = h/(2\pi)$ aufstellt.

Die Schrödinger-Gleichung

Stellt man elektromagnetische Wellen in der Form

$$\Psi(t, \mathbf{r}) = A \exp(i(\mathbf{kr} - \omega t)) \tag{4.3}$$

mit der Wellenzahl $\mathbf{k} = 2\pi/\lambda$, der Kreisfrequenz ω und der Amplitude A dar, so ergeben sich Energie und Impuls zu

$$E = \hbar\, \omega \tag{4.4}$$

$$\mathbf{p} = \hbar\, \mathbf{k}. \tag{4.5}$$

Um Energie und Impuls bei Superposition von Wellen verschiedener Frequenzen und Wellenlängen als Summe der Energien und Impulse zu erhalten, stellt man sie als lineare Operatoren $\hat{E}$, $\hat{\mathbf{p}}$ dar:

$$\hat{E} = i\hbar \frac{\partial}{\partial t} \tag{4.6}$$

$$\hat{\mathbf{p}} = -\, i\hbar \nabla. \tag{4.7}$$

Somit gilt $\hat{E}\Psi = E\Psi$ und $\hat{\mathbf{p}}\Psi = \mathbf{p}\Psi$. Mit der Darstellung von Energie und Impuls erhält man für das freie Elektron aus $E = E_{kin} = p^2/(2m)$ die Gleichung

$$i\hbar \frac{\partial \Psi}{\partial t} = -\frac{\hbar^2}{2m} \Delta \Psi. \tag{4.8}$$

Dies ist die freie zeitabhängige Schrödinger-Gleichung. Bewegt sich das Elektron in einem Potential $V(\mathbf{r})$, so erhält man aus der Beziehung $E = E_{kin} + E_{pot}$ die zeitabhängige Schrödinger-Gleichung

$$i\hbar \frac{\partial \Psi}{\partial t} = \left(-\frac{\hbar^2}{2m} \Delta + V \right) \Psi \tag{4.9}$$

für ein Ein-Teilchen-System. Die rechte Seite wird kurz in der Form $\hat{H}\Psi$ geschrieben mit dem Hamilton-Operator

$$\hat{H} := -\frac{\hbar^2}{2m}\Delta + V. \tag{4.10}$$

Damit ergibt sich die Verallgemeinerung der Schrödinger-Gleichung (4.9) auf Mehrteilchen-Systeme einfach zu

$$\hat{E}\Psi(\mathbf{r_1}, \ldots, \mathbf{r_N}) = \hat{H}\Psi(\mathbf{r_1}, \ldots, \mathbf{r_N}). \tag{4.11}$$

Der Hamilton-Operator $\hat{H}$ ergibt sich dabei, indem man zunächst die Hamilton-Funktion $H(p, q)$ des Systems (im klassischen Sinne) aufstellt und dann die Impulse durch den Impulsoperator (4.7) ersetzt.

4.2 Die stationäre Schrödinger-Gleichung in der Atomphysik

4.2.1 Die stationäre Schrödinger-Gleichung für das Wasserstoffatom

Mögliche stabile Zustände von Elektronen sind durch eine zeitlich konstante Wahrscheinlichkeitsdichte charakterisiert. Die Wellenfunktion Ψ hat daher die Form

$$\Psi(t, \mathbf{r}) = \exp(i\omega t)\Psi(\mathbf{r}). \tag{4.12}$$

Die Schrödinger-Gleichung (4.9) wird somit zur Eigenwertgleichung

$$-\frac{\hbar^2}{2m}\Delta\Psi + V\Psi = E\Psi, \tag{4.13}$$

wobei der Eigenwert E die Energie des Elektrons ist. Diese Gleichung bezeichnet man als stationäre Schrödinger-Gleichung. Sie beschreibt die möglichen Zustände der gebundenen Elektronen im Atom.

Wir bemerken noch, daß der Atomkern in dieser Betrachtung als ruhend und von der Elektronenbewegung unbeeinflußt angesehen wird (Born-Oppenheimer-Näherung).

Unter Einsetzen der elektrischen Potentiale des Atomkerns (im Gaußschen System) ergibt sich für das Wasserstoffatom

$$-\frac{\hbar^2}{2m}\Delta\Psi - \frac{e^2}{r}\Psi = E\Psi. \tag{4.14}$$

Wir suchen zunächst nach den Energien radialsymmetrischer Lösungen.

Radialsymmetrische Lösungen

Von Interesse sind dabei neben der Wellenfunktion Ψ vor allem die möglichen Energien des Elektrons. Für radialsymmetrische Lösungen mit dem Ansatz

$$\Psi(\mathbf{r}) = \psi(r), \quad \mathbf{r} \in \mathbb{R}^3, r = \|\mathbf{r}\|_2 \tag{4.15}$$

$$\tag{4.16}$$

ergibt sich der entsprechende Anteil des Laplace-Operators zu

$$\Delta\Psi(\mathbf{r}) = \frac{1}{r}\frac{\partial^2}{\partial r^2}(r\psi(r)). \tag{4.17}$$

Wir eliminieren nun (siehe [37]) die physikalischen Konstanten durch die Substitutionen $\rho = me^2/\hbar^2 r$, $E = me^4/(2\hbar^2)\epsilon$, $\tilde{\psi}(\rho) = \psi(r)\rho$: Dies führt auf das Eigenwertproblem

$$-\tilde{\psi}''(\rho) - \frac{2}{\rho}\tilde{\psi} = \epsilon\tilde{\psi}, \tag{4.18}$$

$$\tilde{\psi}(\rho) = 0 \text{ für } \rho \to 0, \rho \to \infty. \tag{4.19}$$

Die Skalierungsfaktoren der Energie und des Radius sind dabei die Rydberg-Energie bzw. der Bohrsche Radius. Der gesuchte Eigenwert ist die skalierte Energie ϵ. Die konkreten Randbedingungen in $\rho = 0$ und $\rho = \infty$ sichern $\Psi \in L^2(\mathbb{R}^3)$ sowie zusätzlich hinreichend glattes Verhalten in $\rho = 0$.

4.2.2 Dichtefunktionaltheorie

Für Atome mit mehreren Elektronen hängt die Wellenfunktion von den Koordinaten aller Elektronen ab. Ein solches System ist nicht mehr analytisch lösbar, man bedient sich Näherungsverfahren. Im Rahmen der Dichtefunktionaltheorie betrachtet man nur die Randverteilungen der Wellenfunktion, man beschreibt die N Elektronen durch eine Elektronendichte ϱ und gelangt zu einer Einteilchen-Schrödingergleichung für die Wahrscheinlichkeitsdichte. Dies führt auf die Gleichungen [102]

$$\varepsilon_i \psi_i(\mathbf{r}) = \left(-\Delta + V_{eff}(\mathbf{r})\right) \psi_i(\mathbf{r}), \tag{4.20}$$

$$V_{eff}(\mathbf{r}) = V_{ext}(\mathbf{r}) + \int \frac{2\varrho(\mathbf{r}')}{\|\mathbf{r} - \mathbf{r}'\|_2} dV' + V_{xc}(\mathbf{r}), \tag{4.21}$$

$$\varrho(\mathbf{r}) = \sum_{i=1}^{N} |\psi_i(\mathbf{r})|^2. \tag{4.22}$$

Dabei stellt V_{ext} das Potential des Kerns da, und V_{xc} berücksichtigt weitere Wechselwirkungen der Elektronen untereinander, insbesondere das Pauli-Prinzip. Die Gleichungen (4.20) werden typischerweise über eine Fixpunktiteration für das Effektivpotential V_{eff} gelöst. Dies beinhaltet die Bestimmung der Eigenwerte der Schrödingergleichung, die Lösung V_{eff} wird als selbstkonsistentes Potential bezeichnet.

Ziel unserer Simulationen ist letztendlich ein Beitrag zur Lösung der Gleichungen der Dichtefunktionaltheorie, daher betrachten wir zunächst die Eignung der Verfahren für den einfachen Fall des Wasserstoffatoms.

4.3 Die Magnus-Methode

4.3.1 Ansatz

Ausführliche Darstellungen hierzu finden sich in [56, 57, 58]. Wir betrachten die allgemeine lineare Differentialgleichung

$$y' = A(t)y \tag{4.23}$$

mit zeitabhängigen Koeffizienten.

Betrachten wir gleichzeitig mehrere Anfangswerte, wie es zum Beispiel bei der Lösung der Variationsgleichungen notwendig ist, so gelangen wir zu einer Matrix-Differentialgleichung

$$Y' = A(t)Y. \tag{4.24}$$

Damit befinden wir uns gleichzeitig wieder im Lie-Gruppen-Formalismus. Die rechte Seite stellt ein Vektorfeld auf der Lie-Gruppe dar, ist also als Element der Lie-Algebra zu betrachten. Dieses Element liegt in der Darstellung als zeitabhängiges rechtsinvariantes Vektorfeld vor, formal betrachtet handelt es sich also um eine Quadratur, da wir die rechtsinvarianten Vektorfelder als "konstante" Vektorfelder betrachten.

Die Lösung Y zum Zeitpunkt t setzen wir als Fluß eines vom Zeitpunkt t abhängigen rechtsinvarianten Vektorfeldes an

$$Y(t) = \exp(\sigma(t))Y(0). \tag{4.25}$$

Wird dieser Ansatz in die Differentialgleichung eingesetzt, so muß die Ableitung der Exponentialfunktion gebildet werden. Diese Ableitung ist eine lineare Abbildung vom Tangentialraum der Lie-Algebra (also die Lie-Algebra selbst, da sie ein Vektorraum ist) in den Tangentialraum an die Lie-Gruppe, also ebenfalls die Lie-Algebra. Es ergibt sich

$$Y'(t) = \operatorname{dexp}_{\sigma(t)}(\sigma'(t))Y(t) \tag{4.26}$$

mit der Abbildung dexp, die wie folgt definiert ist: Nach den Bezeichnungen aus Kapitel I ist $d\exp$ (Man unterscheide: dexp und $d\exp$!) das Differential der Exponentialabbildung

$$\begin{array}{rcl} \sigma & \xrightarrow{\exp} & \exp(\sigma) \\ T\mathfrak{g}|_\sigma \cong \mathfrak{g} & \xrightarrow{d\exp} & TG|_{\exp(\sigma)} = dR_{\exp(\sigma)}(\mathfrak{g}). \end{array} \tag{4.27}$$

Die Abbildung $d\exp$ bildet somit von der Lie-Gruppe in die Menge $dR_{\exp(\sigma)}(\mathfrak{g})$ ab. Wir bemerken, daß die Abbildung der Tangentialräume außerdem noch vom Argument σ abhängt. Lassen wir das Differential der Rechtsmultiplikation weg, so erhalten wir eine Abbildung $\mathfrak{g} \to \mathfrak{g}$. Wir bezeichnen diese Abbildung mit dexp.

Definition 4.3.1. *Die Abbildung* $\operatorname{dexp}_B : \mathfrak{g} \to \mathfrak{g}$ *ist gegeben durch*

$$dR_B \circ \operatorname{dexp}_B := d\exp|_B \tag{4.28}$$

oder äquivalent in Matrixdarstellung durch

$$\operatorname{dexp}_B(C) := \left.\frac{d}{dt}\right|_{t=0} \exp(B + tC)\exp(-B). \tag{4.29}$$

Die Abbildung dexp läßt sich als Potenzreihe im Operator $\mathrm{ad}_A := [A, \cdot\,]$ schreiben:

Satz 4.3.1. *Es gilt die Entwicklung*

$$\mathrm{dexp}_B(C) = \sum_{k=0}^{\infty} \frac{1}{(k+1)!} \mathrm{ad}_B^k(C), \tag{4.30}$$

oder äquivalent mit $\phi(x) = \frac{e^x - 1}{x}$

$$\mathrm{dexp}_B(C) = \phi(\mathrm{ad}_B)(C). \tag{4.31}$$

In [47] findet sich ein recht technischer Beweis durch Vergleich der Reihenentwicklungen beider Seiten. Wir führen hier einen Beweis unter Ausnutzung der dexp-Differentialgleichung an:

Definition 4.3.2. *Wir bezeichnen das Anfangswertproblem*

$$Z' = [B, Z] + C = \mathrm{ad}_B(Z) + C, \quad Z(0) = 0 \tag{4.32}$$

als dexp*-Differentialgleichung.*

Die Bedeutung der Gleichung (4.32) wird sofort klar durch

Lemma 4.3.2. *Sei das Anfangswertproblems* (4.32) *in einem Vektorraum* V *mit* $C \in V$ *gegeben, und sei* ad_B *ein beliebiger (beschränkter) linearer Operator. Dann ergibt sich die Lösung zu*

$$Z = \sum_{k=0}^{\infty} \frac{1}{(k+1)!} \mathrm{ad}_B^k(C) = \phi(\mathrm{ad}_B)(C). \tag{4.33}$$

Zum Beweis ist einfach die klassische Lösungstechnik für lineare Differentialgleichungen mit konstanten Koeffizienten anzuwenden. Nun aber zum Beweis von Satz 4.3.1:

Beweis. Sei

$$\mathrm{dexp}_B(C) := \left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} \exp(B + \varepsilon C)\exp(-B).$$

Wir stellen die Matrixexponentialfunktionen als Lösung von Anfangswertproblemen dar:

$$\begin{aligned} Z_\varepsilon'(t) &= (B + \varepsilon C) Z_\varepsilon(t), \quad Z_\varepsilon(0) = I \quad &\Rightarrow Z_\varepsilon(1) &= \exp(B + \varepsilon C) \\ T'(t) &= -T(t) B, \qquad\qquad T(0) = I \quad &\Rightarrow T(1) &= \exp(-B) \end{aligned}$$

Damit erhalten wir auch für

$$Y(t) := \left.\frac{\partial}{\partial\varepsilon}\right|_{\varepsilon=0} (Z_\varepsilon(t) T(t)) \tag{4.34}$$

eine Differentialgleichung:

$$\begin{aligned} Y'(t) &= \left.\frac{\partial}{\partial\varepsilon}\right|_{\varepsilon=0} \left(Z_\varepsilon'(t) T(t) + Z_\varepsilon(t) T'(t)\right) \\ &= \left.\frac{\partial}{\partial\varepsilon}\right|_{\varepsilon=0} ((B + \varepsilon C) Z_\varepsilon(t) T(t) - Z_\varepsilon(t) T(t) B) \\ &= B Y(t) + C Z_0(t) T(t) - Y(t) B \\ &= [B, Y(t)] + C. \end{aligned}$$

Für $t = 0$ ergibt sich aus (4.34) gerade $Y(0) = 0$, für $t = 1$ ergibt sich $Y(1) = \mathrm{dexp}_B(C)$. Mit Lemma 4.3.2 ergibt sich gerade die Behauptung. □

Aus Gleichung (4.26) erhalten wir nun durch Vergleich mit (4.24) und Invertierung von dexp das Anfangswertproblem

$$\sigma'(t) = \mathrm{dexp}^{-1}_{\sigma(t)}(A(t)). \tag{4.35}$$

Die Funktion $\mathrm{dexp}^{-1}_{\sigma}$ können wir ebenso wie dexp in eine Potenzreihe im Operator ad entwickeln, wir haben lediglich die Potenzreihe der reziproken Funktion $1/\phi$ (nicht der inversen Funktion!) zu bestimmen. Diese ergibt sich (siehe [1]) zu

$$\frac{x}{e^x - 1} = \sum_{k=0}^{\infty} \frac{B_k}{k!} x^k \tag{4.36}$$

mit den Bernoulli-Zahlen $(B_0, B_1, \dots) = (1, -1/2, 1/6, 0, 1/30, \dots)$. Dabei verschwinden außer B_1 alle Bernoulli-Zahlen mit ungeradem Index.

4.3.2 Die Magnus-Reihe

Wir haben die Gleichung (4.24) mit dem Ansatz (4.25) auf die Gleichung (4.35) zurückgeführt. In diese setzen wir nun die Reihendarstellung (4.36) ein

$$\sigma'(t) = A(t) - \frac{1}{2}[\sigma(t), A(t)] + \frac{1}{12}[\sigma(t), \sigma(t), A(t)] + \frac{1}{720}[\sigma(t), \sigma(t), \sigma(t), A(t)] + \dots \tag{4.37}$$

Im Gegensatz zur Reihenentwicklung von dexp ist die Reihe (4.37) nicht uneingeschränkt konvergent. Die Funktion $\phi(x) = \frac{e^x - 1}{x}$ ist analytisch, besitzt aber in $x = 2\pi i$ eine Nullstelle. Somit ist der Konvergenzradius für $1/\phi(x)$ höchstens 2π. Da $2\pi i$ offensichtlich die betragskleinste Nullstelle von ϕ ist, konvergiert die Potenzreihe für das Reziproke $1/\phi(x)$ in jedem Kreis mit Radius $r < 2\pi$ gleichmäßig.

In Gleichung (4.37) muß also $\| \mathrm{ad}_\sigma \| < 2\pi$ in einer geeigneten zugeordneten Norm (für Operatoren $\mathfrak{g} \to \mathfrak{g}$) gelten. Wählen wir als Norm in $\mathfrak{g}$ eine multiplikative Matrixnorm, so ergibt sich für den Operator ad die Abschätzung

$$\| \mathrm{ad}_B \| = \max_{\|Y\|=1} \|BY - YB\| \leq 2\|B\| \tag{4.38}$$

Konvergenz können wir somit für $\|B\| \leq \pi$ garantieren.

Zur Magnus-Reihe [66] gelangen wir, wenn wir die Gleichung (4.37) durch Picard-Iteration lösen. Wir beginnen mit dem Startwert $\sigma^{(0)}(t) = 0$ und erhalten

$$\begin{aligned} \sigma^{(1)}(t) &= \int_0^t A(\tau) d\tau \\ \sigma^{(2)}(t) &= \int_0^t A(\tau) d\tau - \int_0^t \frac{1}{2} \left[\int_0^\tau A(\tau_1) d\tau_1, A(\tau) \right] d\tau \pm \dots \end{aligned} \tag{4.39}$$

Weitere Iterationen führen zu mehrdimensionalen Integralen über iterierte Kommutatoren. Nach dem ersten Iterationschritt besteht σ aus dem einfachen Integral über A. Nach dem zweiten Iterationsschritt sind alle einfachen Kommutatoren in σ exakt. Dies können wir induktiv fortsetzen, nach dem k-ten Schritt sind alle $(k-1)$-fach iterierten Kommutatoren exakt. Dies legt nahe, im k-ten Schritt nur die $(k-1)$-fach iterierten Kommutatoren zu berechnen, man nutzt also in der Entwicklung von dexp^{-1} nur die Kommutatoren, die höchstens $(k-1)$-fach iteriert sind.

Nach einer weiteren Picard-Iteration erhalten wir auch alle Ausdrücke mit zweifach-iterierten Kommutatoren exakt. Der Übersichtlichkeit halber stellt man die Integrationen nach vorn und numeriert die Integrationsvariablen systematisch – das Resultat ist die Magnus-Reihe, siehe [66]:

$$\begin{aligned}\sigma^{(3)}(t) = &\int_0^t A(\tau_1)d\tau_1 - \frac{1}{2}\int_0^t \int_0^{\tau_1} [A(\tau_2), A(\tau_1)]\, d\tau_2 d\tau_1 \\ &+ \frac{1}{4}\int_0^t \int_0^{\tau_1} \int_0^{\tau_2} [[A(\tau_3), A(\tau_2)], A(\tau_1)]\, d\tau_3 d\tau_2 d\tau_1 \\ &+ \frac{1}{12}\int_0^t \int_0^{\tau_1} \int_0^{\tau_2} [[A(\tau_3), A(\tau_2)], A(\tau_1)]\, d\tau_3 d\tau_2 d\tau_1 + \dots\end{aligned} \tag{4.40}$$

Sie ist die Grundlage für die Magnus-Methoden.

4.3.3 Darstellung der Magnus-Reihe mit Wurzelbäumen

Interessanterweise kann man die Magnus-Entwicklung unter Verwendung der Ansätze für Bäume aus Definition 2.2.1 und (2.63) darstellen. Zur Darstellung genügen die klassischen Bäume für gewöhnliche Differentialgleichungen (Definition 2.2.1), wobei wie in (2.63) bei jedem Elternknoten die Reihenfolge der angehängten Söhne berücksichtigt wird. Eine Darstellung mit weitaus größerer Knotenzahl findet man in [58].

Definition 4.3.3. *Die Menge der Wurzelbäume ist rekursiv gegeben als Menge T durch*

- $[\,] \in T$
- *mit $s_1, \dots, s_k$ ist auch $[s_1, \dots, s_k] \in T$.*

Wir bemerken noch einmal, daß hier eine unterschiedliche Reihenfolge der Söhne $s_1, \dots, s_k$ zu einem anderen Baum führt. Wir ordnen den Bäumen iterierte Integrale und Koeffizienten zu:

Definition 4.3.4. *Bezeichne $s = [s_1, \dots, s_k]$ einen Wurzelbaum. Dann ist die Funktion $F(s)$ rekursiv gegeben durch*

$$F([\,])(t) = \int_0^t A(\tau_1)d\tau_1 \tag{4.41}$$

$$F(s)(t) = \int_0^t [F(s_1)(\tau_1), \dots, F(s_k)(\tau_1), A(\tau_1)]. \tag{4.42}$$

Die Koeffizienten $\gamma(s)$ ergeben sich rekursiv zu

$$\gamma([\,]) = 1 \tag{4.43}$$

$$\gamma(s) = \frac{B_k}{k!}\prod_{i=1}^{k}\gamma(s_i) \tag{4.44}$$

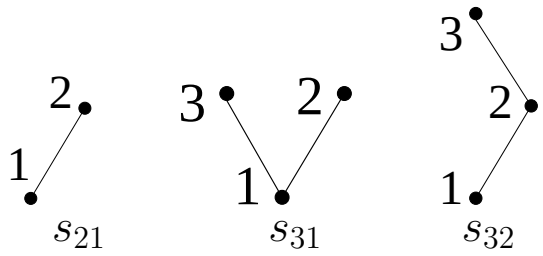
Man erkennt leicht, daß diese Rekursion gerade die bei einer fortgesetzten Picard-Iteration entstehenden iterierten Integrale mit den korrekten Koeffizienten generiert. Wir erhalten daher (ohne Beweis)

Satz 4.3.3. *Die Magnus-Reihe ergibt sich zu*

$$\sigma(t) = \sum_{s\in T}\gamma(s)(t)F(s)(t). \tag{4.45}$$

Wir bemerken noch, daß hier einige Bäume gegenüber dem klassischen Fall wegfallen, weil die Bernoulli-Zahlen für ungeraden Index $k \geq 3$ verschwinden. Ein Knoten hat also entweder genau einen Sohn oder eine gerade Anzahl von Söhnen. Auf der anderen Seite wird hier die Reihenfolge der Söhne berücksichtigt, was wiederum die Anzahl verschiedener Bäume erhöht.

Möchte man die Funktionen $F(s)$ explizit generieren, so empfiehlt sich eine monotone Indizierung der Knoten des Baumes s. Die Knoten eines Baums sind monoton indiziert, wenn die Numerierung vom Vater zum Sohn aufsteigend ist, wobei die Wurzel den Index 1 bekommt. Für den Baum [ ] mit einem Knoten ergibt sich $\int_0^t A(\tau_1)d\tau_1$. Für die Bäume mit 2 und 3 Knoten



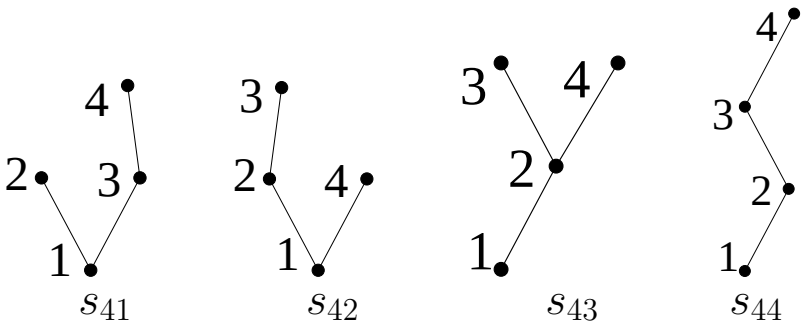
(4.46)

ergeben sich damit die Ausdrücke

$$\begin{aligned}
\gamma(s_{21})F(s_{21})(t) &= -\frac{1}{2}\int_0^t \int_0^{\tau_1} [A(\tau_2), A(\tau_1)]d\tau_2 d\tau_1 \\
\gamma(s_{31})F(s_{31})(t) &= \frac{1}{12}\int_0^t \int_0^{\tau_1} \int_0^{\tau_1} [[A(\tau_3), A(\tau_2)], A(\tau_1)]d\tau_3 d\tau_2 d\tau_1 \\
\gamma(s_{32})F(s_{32})(t) &= \frac{1}{4}\int_0^t \int_0^{\tau_1} \int_0^{\tau_2} [[A(\tau_3), A(\tau_2)], A(\tau_1)]d\tau_3 d\tau_2 d\tau_1.
\end{aligned} \tag{4.47}$$

Die Konstruktion der Ausdrücke ist anhand obiger Beispiele leicht nachvollziehbar: Für jeden Knoten (von k Knoten insgesamt) erhalten wir eine Integration, die Integrationsvariablen sind dabei (außen beginnend) τ_1, τ_2, …, τ_k. Die obere Grenze der Integration ergibt sich für die Integration nach τ_i gerade aus dem Vaterknoten vom Knoten i. Die Integranden werden rekursiv aus Kommutatoren gemäß der Definition 4.3.3 generiert, dem Knoten i wird dabei $A(\tau_i)$ zugeordnet. Die Rekursion für den Vorfaktor γ liefert das Produkt der Bernoulli-Zahlen $B_{j_i}/j_i!$, wobei i über alle Vaterknoten läuft und j_i die Anzahl der Söhne bezeichnet.

Für die Bäume mit 4 Knoten



(4.48)

erhalten wir

$$\begin{aligned}
\gamma(s_{41})F(s_{41})(t) &= -\frac{1}{24}\int_0^t\int_0^{\tau_1}\int_0^{\tau_1}\int_0^{\tau_3}[[[A(\tau_4,A(\tau_3)],A(\tau_2)],A(\tau_1)]d\tau_4 d\tau_3 d\tau_2 d\tau_1\\
\gamma(s_{42})F(s_{42})(t) &= -\frac{1}{24}\int_0^t\int_0^{\tau_1}\int_0^{\tau_2}\int_0^{\tau_1}[[[A(\tau_3,A(\tau_2)],A(\tau_4)],A(\tau_1)]d\tau_4 d\tau_3 d\tau_2 d\tau_1\\
\gamma(s_{43})F(s_{43})(t) &= -\frac{1}{24}\int_0^t\int_0^{\tau_1}\int_0^{\tau_2}\int_0^{\tau_2}[[[A(\tau_3,A(\tau_4)],A(\tau_2)],A(\tau_1)]d\tau_4 d\tau_3 d\tau_2 d\tau_1\\
\gamma(s_{44})F(s_{44})(t) &= -\frac{1}{8}\int_0^t\int_0^{\tau_1}\int_0^{\tau_2}\int_0^{\tau_3}[[[A(\tau_4,A(\tau_3)],A(\tau_2)],A(\tau_1)]d\tau_4 d\tau_3 d\tau_2 d\tau_1.
\end{aligned} \tag{4.49}$$

4.3.4 Numerische Approximation der Magnus-Reihe

Für einen Integrationsschritt im Intervall $[0,h]$ ist $\sigma(h)$ zu approximieren. Es bieten sich 3 Vorgehensweisen an:

- Man setzt A als Taylorreihe an und bestimmt unter Nutzung des Paketes GLIEALG in Mathematica eine Darstellung für σ aus Gleichung (4.37). Dabei muß man berücksichtigen, daß für Ordnung k die Lie-Algebra aus k Atomen generiert wird. Setzen wir dann σ als Linearkombination iterierter Kommutatoren an, so ist mit einer nicht mehr handhabbaren Anzahl von Variablen und Gleichungen zu rechnen.
- Man wertet die Magnus-Entwicklung direkt aus, indem man Formeln für multivariate Quadratur einsetzt, man siehe [56, 57, 58].
- Man folgt den Ideen aus [18, 19], wo Integralansätze unter Ausnutzung von Zeitsymmetrie zu sehr einfachen Darstellungen führen.

Wir nutzen für Verfahren niederer Ordnung die erste Variante, für höhere Ordnung bietet aber die dritte Variante einen elegantere Darstellung.

Taylorreihen-Ansatz

Sei A in eine Taylorreihe

$$A(t) = A_0 + A_1 t + A_2\frac{t^2}{2} + \dots \tag{4.50}$$

entwickelt. Wir setzen dies in

$$\sigma' = A - \frac{1}{2}[\sigma, A] + 1/12[\sigma,\sigma,A] + \dots \tag{4.51}$$

ein. Bevor wir mit einer rekursiven Generierung der Lösung σ beginnen, stellen wir zunächst die Ordnung in t-Potenzen fest. Wir folgern sukzessiv

$$\begin{aligned}
\sigma(t) &= \mathcal{O}(t)\\
\sigma(t) &= tA(0) + \mathcal{O}\left(t^2\right)\\
[\sigma(t), A(t)] &= \mathcal{O}\left(t^2\right)\\
[\underbrace{\sigma(t),\dots,\sigma(t)}_{k-\text{mal}}, A(t)] &= \mathcal{O}\left(t^{k+1}\right).
\end{aligned} \tag{4.52}$$

Für Ordnung 4 genügen somit die ersten 3 Summanden in (4.37), zur Ordnung 6 die ersten vier.

Wir führen wieder Picard-Iteration durch. Durch Integration der rechten Seite erhöht sich die Ordnung (in t-Potenzen) in jedem Term um eins. Wir arbeiten daher in der bewerteten freien Lie-Algebra, erzeugt durch

$$Y_1 = tA_0, Y_2 = t^2 A_1, Y_3 = t^3 A_2, \ldots. \tag{4.53}$$

Die Picard-Iteration starten wir mit $\sigma^{(0)} = 0$ aus der bewerteten Lie-Algebra. Eine Iteration ist gleichbedeutend mit

$$\sigma^{(k+1)}(t) = \int_0^t \left(\mathrm{dexp}^{-1}_{\sigma^{(k)}(\tau)}(A(\tau)) \right) d\tau \tag{4.54}$$

Die Abbildung dexp^{-1} kann man unter Nutzung des Paketes GLIEALG leicht als Potenzreihe in ad implementieren. Die Integration reduziert sich darauf, Terme Y_l vom Grad m durch Y_l/m zu ersetzen. So erhalten wir für Ordnung 4

$$\sigma = Y_1 + \frac{1}{2}Y_2 + \frac{1}{3}Y_3 + \frac{1}{4}Y_4 - \frac{1}{12}[Y_1, Y_2 + Y_3]. \tag{4.55}$$

Bei vorhandenen Approximationen an die Y_i genügt also ein Kommutator für Ordnung 4. Für höhere Ordnung kann man wie in Kapitel 3 versuchen, mit allgemeinen Ansätzen geeignete Berechnungsvorschriften zu gewinnen. Die Anzahl der freien Parameter steigt jedoch schnell an, so sind für Ordnung 6 vier Kommutatoren zu berechnen, und in der freien bewerteten Lie-Algebra sind 6 Basiselemente vorhanden. Der letzte handhabbare Fall in Kapitel 3 enthielt 3 Basiselemente, daher nehmen wir hier Abstand von dieser Vorgehensweise und umreißen im folgenden kurz den eleganten Ansatz von Blanes, Casas und Ros ([18], [19]).

Approximation mit Integralformeln

Blanes, Casas und Ros stellen die Reihe (4.40) durch zeitsymmetrische Ausdrücke mit Hilfe von Integralen dar.

Symmetrische Entwicklung Die Matrix A wird in eine Taylorreihe um den Intervallmittelpunkt entwickelt:

$$A(t) = \sum_{k=0}^{\infty} a_k \left(t - \frac{h}{2} \right)^k. \tag{4.56}$$

Diese Entwicklung wird in die Gleichung (4.37) eingesetzt und die Picard-Iteration durchgeführt. Der symmetrische Ansatz hat zur Folge, daß nach der Integration nur die ungeraden Potenzen von h verbleiben.

$$\begin{aligned} \sigma = & ha_0 + h^3 \frac{1}{12} a_2 + h^5 \frac{1}{80} a_4 - h^3 \frac{1}{12}[a_0, a_1] + h^5 \left(\frac{1}{80}[a_0, a_3] + \frac{1}{240}[a_1, a_2] \right) \\ & + h^5 \left(\frac{1}{360}[a_0, a_0, a_2] - \frac{1}{240}[a_1, a_0, a_1] + \frac{1}{720}[a_0, a_0, a_0, a_1] \right) + \mathcal{O}\left(h^7\right) \end{aligned} \tag{4.57}$$

Integralansatz Die Entwicklung (4.57) wird weiter vereinfacht, wenn wir die Integrale

$$B^{(i)} = \frac{1}{h^{i+1}} \int_0^h \left(t - \frac{h}{2}\right)^i A(t)\ dt \qquad i = 0, 1, 2, ... \tag{4.58}$$

einsetzen. Die ersten drei dieser Integrale sind gegeben durch

$$B^{(0)} = a_0 + \frac{1}{12}h^2 a_2 + \frac{1}{80}h^4 a_4 + \frac{1}{448}h^6 a_6 + ... \tag{4.59}$$

$$B^{(1)} = \frac{1}{12}h\, a_1 + \frac{1}{80}h^3 a_3 + \frac{1}{448}h^5 + ... \tag{4.60}$$

$$B^{(2)} = \frac{1}{12}a_0 + \frac{1}{80}h^2 a_2 + \frac{1}{448}h^4 a_4 + \frac{1}{2304}h^6 a_6 + ... \tag{4.61}$$

Ein Vorteil dieser Herangehensweise ist, daß der Algorithmus in zwei unabhängige elementare Teilaufgaben zerlegt wurde – die numerische Approximation der Integrale und die Approximation von σ durch die Integrale $B^{(i)}$. Der zweite Schritt ist dabei von der im ersten Schritt konkret gewählten Quadraturformel, also insbesondere von den Knoten, unabhängig.

Ordnung 4 Für eine Approximation der Ordnung 4 ist lediglich ein Kommutator zu berücksichtigen. Man wählt nach [18] $\sigma = \omega_1 + \omega_3$ wobei ω_1, ω_3 durch

$$\omega_1 = hB^{(0)} \tag{4.62}$$

$$\omega_3 = -h^2[B^{(0)}, B^{(1)}] \tag{4.63}$$

gegeben sind.

Ordnung 6 Die Näherung 6-ter Ordnung $\sigma^{(6)}$ ist durch

$$\sigma^{(6)} = \omega_1 + \omega_3 + \omega_5 \tag{4.64}$$

gegeben, wobei ω_i Ausdrücke der Ordnung i enthält:

$$\begin{aligned}
\omega_1 =& ha_0 \\
\omega_3 =& h^3 \left(\frac{1}{12}a_2 + \frac{-1}{12}[a_0, a_1] \right) \\
\omega_5 =& h^5 \left(\frac{1}{80}a_4 + \frac{1}{80}[a_0, a_3] + \frac{1}{240}[a_1, a_2] + \frac{1}{360}[a_0, a_0, a_2] \right. \\
& \left. -\frac{1}{240}[a_1, a_0, a_1] + \frac{1}{720}[a_0, a_0, a_0, a_1] \right).
\end{aligned} \tag{4.65}$$

Der Ausdruck (4.65) wird nun durch $B^{(i)}$ mit Ordnung 6 approximiert. Hierzu wird ebenso ein Ansatz mit iterierten Kommutatoren gewählt. Durch die Darstellung mit den Integralen $B^{(i)}$ kann man sich hier auf wenige Parameter im Ansatz beschränken. Allerdings sind mindestens 3 Kommutatoren zur Berechnung notwendig, wie man aus dem Auftreten des Terms $[a_0, a_0, a_0, a_1]$ folgert.

Approximiert wird (4.65) nach Casas et al. [18] mit 4 Kommutatoren durch

$$\begin{aligned}
\omega_1 =& hB^{(0)} \\
\omega_3 =& h^2 \left[B^{(1)}, \frac{3}{2}B^{(0)} - 6B^{(2)} \right] \\
\omega_5 =& \left[\omega_1, \left[\omega_1, \frac{1}{2}hB^{(2)} - \frac{1}{60}\omega_3 \right] \right] + \frac{3}{5}h \left[B^{(1)}, \omega_3 \right].
\end{aligned} \tag{4.66}$$

Quadratur Zur Minimierung des Aufwandes bietet sich Gauß-Legendre-Quadratur an, da somit die höchstmögliche Ordnung erzielt wird. Auch kann die Symmetrie der Knoten für bestimmte Anwendungen von Vorteil sein. Als Knoten wählen wir somit die Nullstellen der verschobenen Legendre-Polynome (hier ohne Normierung)

$$L_s(x) := \frac{d^s}{dx^s}\left(x^s(1-x)^s\right). \tag{4.67}$$

Die Ordnung dieser Formeln ist $2s$ für s Knoten, wir benötigen also 2 bzw. 3 Funktionsauswertungen pro Schritt für eine Methode der Ordnung 4 bzw. 6. Die Knoten c und Gewichte b^T sind dabei:

$$s = 2: \quad c = \left(\frac{3-\sqrt{3}}{6}, \quad \frac{3-\sqrt{3}}{6}\right)^T, \quad b^T = \left(\frac{1}{2}, \frac{1}{2}\right) \tag{4.68}$$

$$s = 3: \quad c = \left(\frac{1}{2} - \frac{\sqrt{15}}{10}, \frac{1}{2}, \frac{1}{2} + \frac{\sqrt{15}}{10}\right)^T, \quad b^T = \left(\frac{5}{18}, \frac{4}{9}, \frac{5}{18}\right) \tag{4.69}$$

Die Approximationen an $B^{(k)}$ sind gegeben durch

$$B^{(k)} = \sum_{i=1}^{3} b_i (c_i - \frac{1}{2})^k A(c_i h). \tag{4.70}$$

4.4 Numerische Tests

4.4.1 Schrittweitensteuerung für die Magnus-Methode

Zu einer effizienten Implementierung eines Zeitintegrationsverfahrens ist für viele Anwendungen eine Schrittweitensteuerung unumgänglich. Wir haben hier als typische Anwendung Gleichungen mit oszillierender Lösung und Schrittweiten, die ein Vielfaches der Periode betragen sollen, vor Augen. Eine Schrittweitensteuerung ist in diesem Falle sinnvoll, wenn die Periode oder die Abweichung von der reinen Sinus-Schwingung (diese wird exakt für beliebige Schrittweiten gelöst) variiert. Auch kann man a priori nur schwer eine Schrittweite festlegen, so daß selbst bei Rechnungen mit nahezu konstanter Schrittweite eine Schrittweitensteuerung sinnvoll ist.

Grundlage einer Schrittweitensteuerung bildet das Konzept der Gleichverteilung des lokalen Fehlers. Dazu bedarf es einer Schätzung des lokalen Fehlers, wozu das Prinzip der Einbettung oder aber Richardson-Extrapolation genutzt werden kann. Während Richardson-Extrapolation prinzipiell für jedes Verfahren direkt zur Verfügung steht, erfordert Einbettung neben dem Grundverfahren mit der Ordnung p ein zweites Verfahren (eingebettetes Verfahren) von der Ordnung q. Die Differenz der beiden numerischen Lösungen wird als Fehlerschätzer genutzt.

Bei der Magnus-Methode treten zwei Arten von Diskretisierungsfehlern auf: Zum einen tritt in den Quadraturformeln zur Berechnung der $B^{(k)}$ der klassische Quadraturfehler auf, zum anderen resultiert ein Abbruch-Fehler aus der Magnus-Entwicklung. Der letztere läßt sich sehr einfach schätzen. Der obige Algorithmus nach [18] liefert neben einer Lösung 6. Ordnung auch eine Lösung 4. Ordnung. Allerdings nutzt diese Näherung 4. Ordnung die gleichen Quadraturformeln – der Quadraturfehler geht somit nicht in die Fehlerschätzung ein. Außerdem ist der so geschätzte Fehler von Ordnung 4, so daß sich die Methode bei Änderungen der vorgegebenen Toleranz wie ein Verfahren 4. Ordnung verhält – bei kleinen Toleranzen wird der Fehler überschätzt, das Verfahren arbeitet mit zu kleinen Schrittweiten.

Ein möglicher Ausweg ist die Verwendung einer zweiten Quadraturformel. Leider ist eine Nutzung der 3 Gaußknoten nicht zweckmäßig. Wir können nämlich mit drei neuen Knoten Ordnung 5 erreichen (Radau-Knoten) oder mit 2 neuen Knoten Ordnung 4 (Gauß-Knoten) erreichen. Die Verwendung der bereits berechneten Werte in den Gauß-Knoten ist nicht von Vorteil, da wir mit einem zusätzlichen Knoten (zu den drei vorhandenen Gauß-Knoten) nur Ordnung 3 erreichen können. Denn für Ordnung 4 mit 4 Knoten sind die Gewichte eindeutig bestimmt – und da die Gauß-Formel selbst eine mögliche Lösung ist, muß das Gewicht des neuen Knotens 0 sein. Auch läßt sich mit zwei zusätzlichen Knoten nicht die Ordnung 5 erreichen. Wir können wieder analog schließen, daß für Ordnung 5 die Gewichte eindeutig bestimmt sind, und damit die Gewichte der zusätzlichen Knoten verschwinden müssen.

Wir schlagen folgende vier Alternativen zur Schrittweitensteuerung vor:

- Der Vorschlag aus [18] – es werden Approximationen unterschiedlicher Genauigkeit an die Magnus-Reihe genutzt. Vorteil ist der geringe Aufwand, Nachteile sind wie oben beschrieben, daß der Quadraturfehler nicht berücksichtigt wird und der Schätzer nur 4. Ordnung ist. Ist der Kommutator identisch Null, so versagt der Schätzer.

- Einbettung einer Radau-Formel mit 3 Knoten. Diese Variante hat den höchsten Zusatzaufwand, ein weiterer Nachteil ist, daß der Schätzer nicht mehr zeitsymmetrisch ist. Von Vorteil ist die Ordnung 5 des Schätzers.

- Einbettung der zweistufigen Gauß-Formel der Ordnung 4. So erhalten wir die Zeitsymmetrie der Methode.

- Richardson-Extrapolation. Der Mehraufwand kann optimistisch mit 50% geschätzt werden, wenn wir die beiden Halbschritte als Einzelschritt werten. Hier ist der Schätzer sogar von Ordnung 6, man würde von dieser Variante also höchste Effizienz für kleine Toleranzen erwarten. Zusätzlich bietet sich noch die Möglichkeit der Extrapolation auf höhere Ordnung.

4.4.2 Die Bessel-Funktion

In diesem Beispiel wollen wir uns davon überzeugen, daß die Magnusmethode in der Tat zur Integration von Gleichungen mit stark oszillierenden Lösungen geeignet ist. Gegeben ist die Anfangswertaufgabe

$$\begin{aligned} y''(x) &= -\left(100 + \frac{1}{4x^2}\right) y(x) \\ y(1) = J_0(10), &\quad y'(1) = \frac{1}{2} J_0(10) - 10 J_1(10). \end{aligned} \tag{4.71}$$

Dabei bezeichnet J_k die Bessel-Funktion erster Art [1]. Die analytische Lösung von Gleichung (4.71) ist durch

$$y(x) = \sqrt{x} J_0(10x) \tag{4.72}$$

gegeben. Diese Funktion oszilliert. Schreiben wir die Differentialgleichung (4.71) als System erster Ordnung, so können wir die Magnus-Methode darauf anwenden.

Zuverlässigkeit der Schrittweitensteuerung

Wir haben die vier oben erwähnten Varianten am Problem (4.71) getestet. Für verschiedene Toleranzen tragen wir jeweils den Quotienten aus erreichter Genauigkeit (maximaler globaler Fehler) und vorgegebener Toleranz auf. Um die eigentliche Effizienz einschätzen zu können, haben wir außerdem noch die erreichte Genauigkeit gegen den rechnerischen Aufwand aufgetragen. In der Legende steht

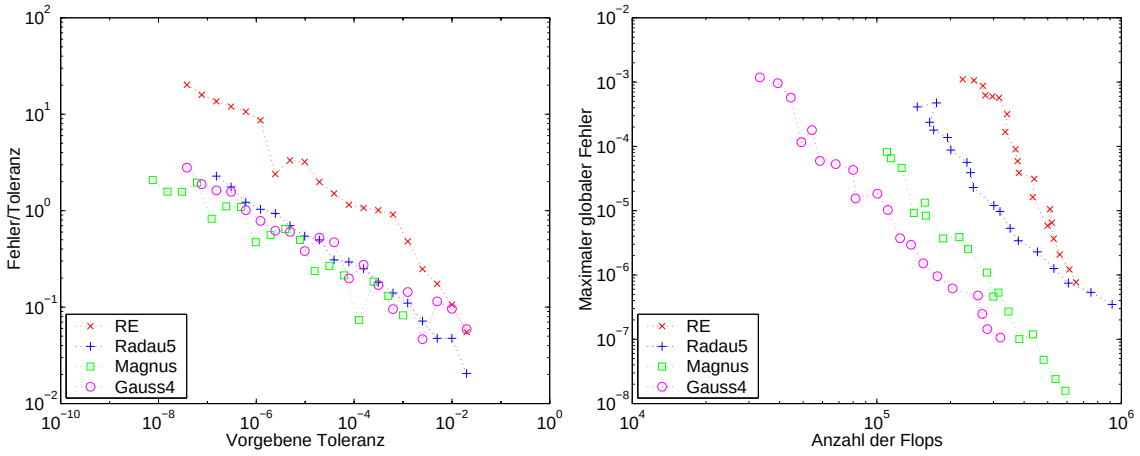


Abbildung 4.1: Vergleich der 4 Varianten zur Schrittweitensteuerung am Problem Bessel

RE für Richardson-Extrapolation, Radau5 für die Schätzung des Quadraturfehlers durch eine eingebettete Radau-Methode der Ordnung 5, Magnus für Schätzung des Fehlers einzig in der Magnus-Reihe und Gauss4 für Einbettung der 2-stufigen Gauß-Methode. Die letztere Variante schneidet offensichtlich am besten ab, sie ist sowohl zuverlässiger als auch effizienter bei vergleichbarer Genauigkeit.

Bei Richardson-Extrapolation wird der Fehler zumeist unterschätzt, das Aufwandsdiagramm macht deutlich, daß diese Variante zur Schrittweitensteuerung nicht zu empfehlen ist.

Vergleich mit klassischen Integratoren

Wir vergleichen die Magnus-Methode der Ordnung 6, basierend auf Gauß-Quadratur, mit

- der klassischen Gauß-Formel selbst (implizit, Ordnung 6)
- der Runge-Kutta-Methode DOPRI5 (explizit, Ordnung 5)
- der klassischen Runge-Kutta-Methode (explizit, Ordnung 4)

Das Problem wird auf dem Intervall von $[1, 100]$ durch alle 4 Methoden gelöst. Zunächst wenden wir die Methoden mit konstanter Schrittweite $h = 1.E - 1$ an, was 990 Schritten entspricht. Der globale Fehler im zeitlichen Verlauf ist in Abbildung 4.2 dargestellt, aufgrund des oszillatorischen Verlaufs der exakten Lösung folgt der globale Fehler einem fast regulären Muster.

Die Magnus-Methode hält den globalen Fehler gleichmäßig unterhalb von $4.E - 8$, während für die 3 anderen Methoden eine Akkumulation des globalen Fehlers festzustellen ist. Dies weist darauf hin, daß bei der Magnus-Methode keine Phasenverschiebung auftritt. Auch ist der globale Fehler bei der Magnus-Methode um Größenordnungen geringer als bei den klassischen Verfahren. Insbesondere bei der klassischen Runge-Kutta-Methode ist die Lösung im Endpunkt komplett unbrauchbar, was daraus resultiert, daß die Amplitude der numerischen Lösung fällt, während die analytische Lösung eine konstante Amplitude besitzt.

In Abbildung 4.3 werden die Ergebnisse eines ähnlichen Experiments gezeigt, nur wurden jetzt alle 4 Methoden mit Schrittweitensteuerung angewandt. Wir haben relative Toleranzen von $1.E - 4$ und absolute Toleranzen von $1.E-6$ vorgeschrieben. Zur Fehlerschätzung haben wir für die klassische Runge-Kutta-Methode Richardson-Extrapolation verwendet, für die anderen drei Methoden haben wir Einbettung verwendet. DOPRI5 ist bereits mit einem eingebetteten Fehlerschätzer der Ordnung 4 versehen, während wir für die Gauß-Methode der Ordnung 6 die zweistufige Gauß-Methode der Ordnung 4 zur Fehlerschätzung herangezogen haben.

Mit Ausnahme der Magnus-Methode haben die anderen drei Methoden ungefähr 2000 Schritte zur Lösung benötigt. Die Magnus-Methode löst das gleiche Problem in 160 Schritten.

In einem dritten Versuch vergleichen wir die Effizienz der Methoden bei Schrittweitensteuerung. Das linke Diagramm in Abbildung 4.4 zeigt den maximalen globalen Fehler gegen die Anzahl der ausgeführten Schritte aufgetragen, wobei Schrittwiederholungen nicht mitgezählt wurden. Die Methoden schneiden fast wie erwartet ab: Die Magnus-Methode ist deutlich effizienter als die anderen Methoden. Diese rangieren entsprechend ihrer Ordnung, die Gauß-Methode der Ordnung 6 vor DOPRI5 und der klassischen Runge-Kutta-Methode.

Zur realistischen Einschätzung muß natürlich der eigentliche rechnerische Aufwand berücksichtigt werden. Auf dem rechten Diagramm in Abbildung 4.4 haben wird daher die ausgeführten Flops auf der Abszisse abgetragen. Wir bemerken, daß die Schrittweitensteuerung für alle Verfahren zufriedenstellend funktioniert, die Anzahl der Wiederholungen liegt unterhalb von 10%. Die Magnus-Methode schneidet auch hier am besten ab. Die Matrixexponentialfunktionen fallen nicht ins Gewicht, da für Matrizen aus $\mathbb{R}^{2\times 2}$ eine effiziente Implementierung der Matrixexponentialfunktion möglich ist, man vergleiche Abschnitt 1.3.7. Hier schneidet DOPRI5 besser als die Gauß-Methode ab, denn für das implizite Verfahren ist ein lineares Gleichungssystem der Dimension 6×6 pro Schritt zu lösen.

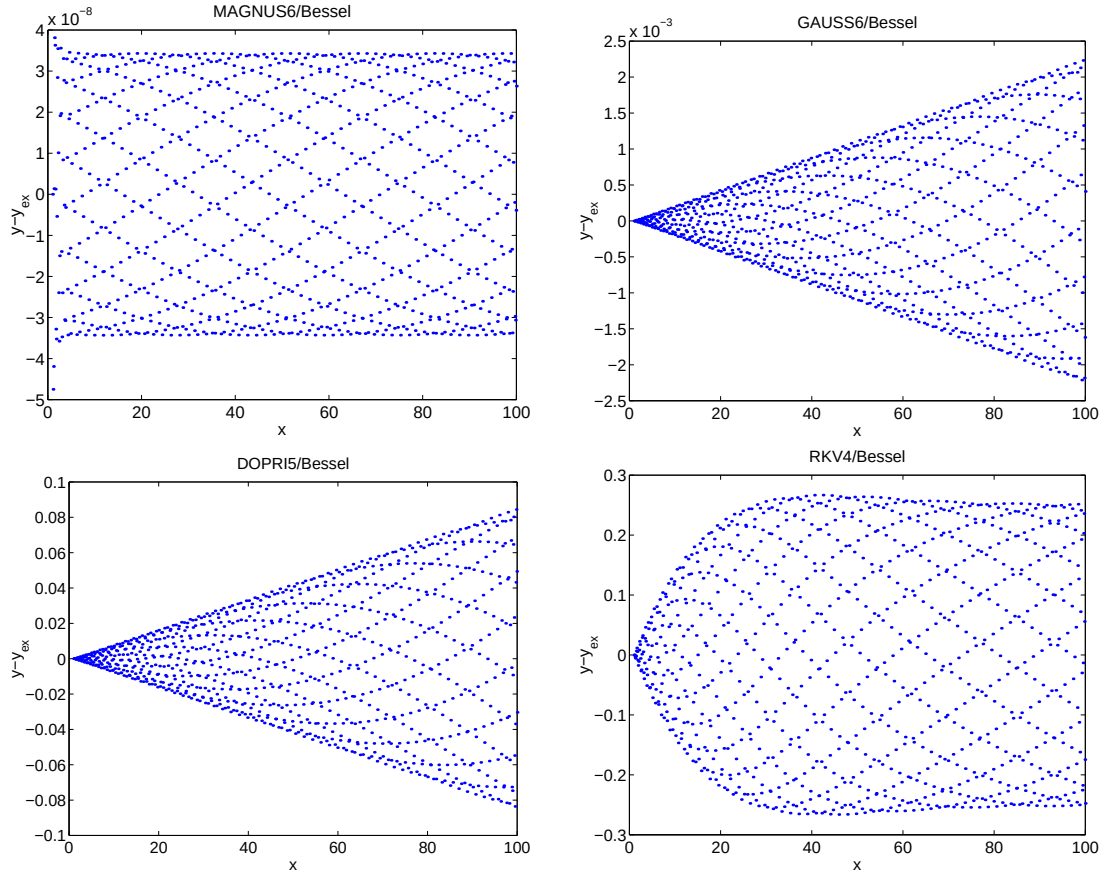


Abbildung 4.2: Globaler Fehler bei Bessel, Rechnung mit konstanter Schrittweite $h = 1.E - 1$, $x_e = 100$.

4.4.3 Die radiale Schrödinger Gleichung für das Wasserstoffatom

Zur Bestimmung der möglichen Energien für drehimpulsfreie Zustände der Elektronen (also komplett radialsymmetrische Wellenfunktionen) lösen wir das Eigenwertproblem

$$-\psi''(\rho) - \frac{2}{\rho}\psi = \epsilon\psi \tag{4.73}$$

mit den Randbedingungen (4.19). Weitere Anwendungen wie die Behandlung des relativistischen Falls mit der Dirac-Gleichung findet man in [122].

Für eine erste Näherung betrachten wir das asymptotische Verhalten der Lösungen am Rand: Unter der Voraussetzung, daß ψ in $\rho = 0$ in eine Potenzreihe entwickelbar ist, ergibt sich zunächst $\psi(0) = 0$. Wir setzen o.B.d.A. $\psi'(0) = 1$. Der Koeffizientenvergleich ergibt dann

$$\psi(\rho) = \rho - \rho^2 + \mathcal{O}\left(\rho^2\right). \tag{4.74}$$

Zur Bestimmung des asymptotischen Verhaltens in ∞ vernachlässigen wir den Term $-2/\rho\psi$. Die so erhaltene lineare Differentialgleichung mit konstanten Koeffizienten hat die Lösungen $\exp(\pm\sqrt{-\epsilon}\rho)$, von denen nur die abklingende Lösung

$$\psi(\rho) \approx \exp(-\sqrt{-\epsilon}\rho) \tag{4.75}$$

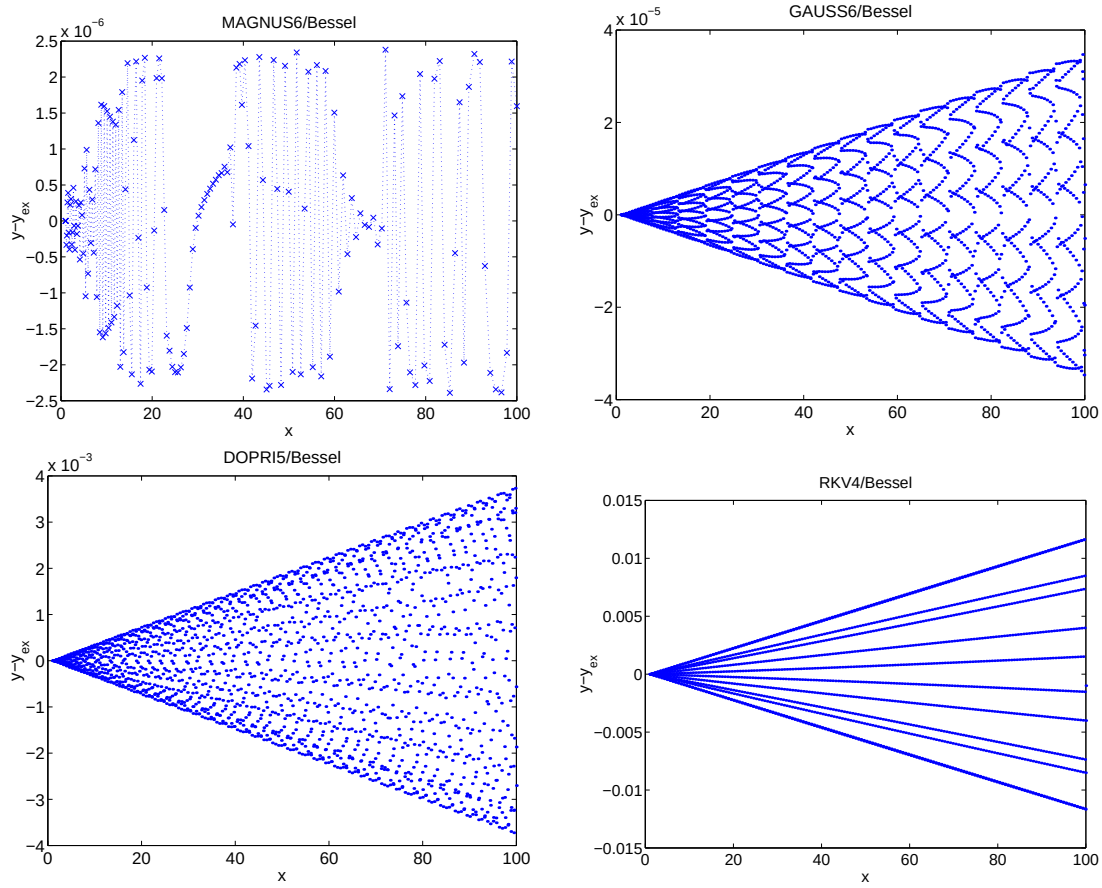


Abbildung 4.3: Der globale Fehler bei Bessel bei Rechnung mit Schrittweitensteuerung, $RTOL = 1.E-4$, $x_e = 100$

der Randbedingung genügt.

Zur numerischen Lösung des Problems setzen wir die Randwerte für vorgegebene Energie ϵ einmal in $\rho_{\min} \ll 1$ nach (4.74), einmal für $\rho_{\max} > 1$ nach (4.75), so daß sich abklingende Lösungen ergeben. Konkret setzen wir

$$\psi(\rho_{\min}) = \rho_{\min} \qquad \psi(\rho_{\max}) = 1 \tag{4.76}$$

$$\psi'(\rho_{\min}) = 1 \qquad \psi(\rho_{\max}) = -\sqrt{-\epsilon}. \tag{4.77}$$

Dann integrieren wir von $\rho_{\min}$ nach rechts bis ρ_{mid} und erhalten eine Näherung ψ_L in ρ_{mid}, sowie von ρ_{max} nach links bis ρ_{mid} und erhalten eine Näherung ψ_R in ρ_{mid}.

Zur Integration nutzen wir die Magnus-Methode mit einer Toleranz von $1.E-8$. Dazu wird Gleichung (4.73) durch Einführung einer zusätzlichen Variablen für ψ' in ein System erster Ordnung überführt. Somit erhalten wir auch Approximationen für die ersten Ableitungen von ψ in ρ_{mid}, jeweils rechtsseitig (ψ'_R) und linksseitig (ψ'_L).

Ist ϵ nun ein Eigenwert, so müssen die linksseitige und die rechtsseitige Lösung so skalierbar sein, daß sie in ρ_{mid} stetig anschließen. Die Vektoren (ψ_L, ψ'_L) und (ψ_R, ψ'_R) müssen linear abhängig sein.

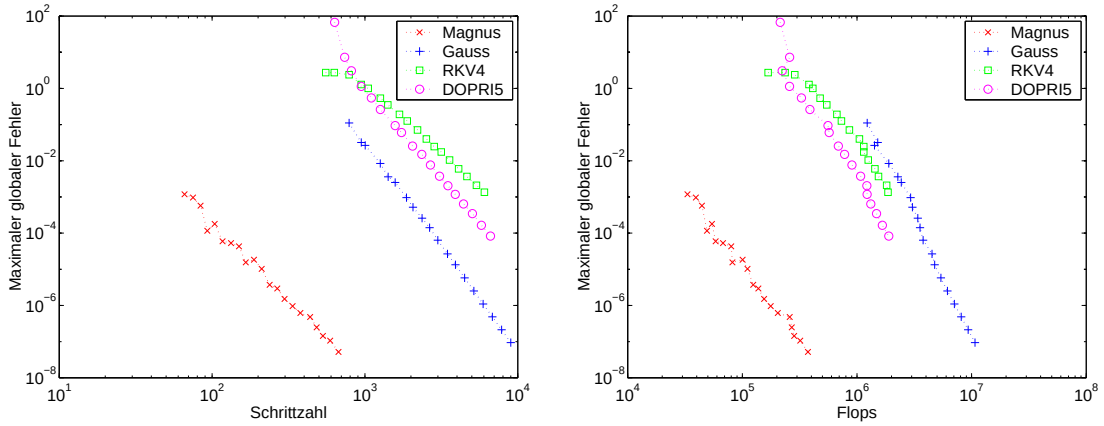


Abbildung 4.4: Maximaler globaler Fehler bei Bessel gegen Schrittweite und Anzahl der Flops für alle 4 Methoden.

Wir bestimmen daher den Sinus $s = \sin\phi$ des eingeschlossenen Winkels ϕ durch

$$\begin{aligned} z =& (\psi_L + i\psi'_L)/(\psi_R + i\psi'_R) \\ s =& \operatorname{Im} z/|z|. \end{aligned} \tag{4.78}$$

Für Energien ϵ im Bereich $[-1, 0)$ erhalten wir s wie in Abbildung 4.5 dargestellt. Die Nullstellen liegen sehr nahe bei den exakten Werten $-1/n^2$, $n = 1, 2, 3, \ldots$. Die exakten Energien bestimmen

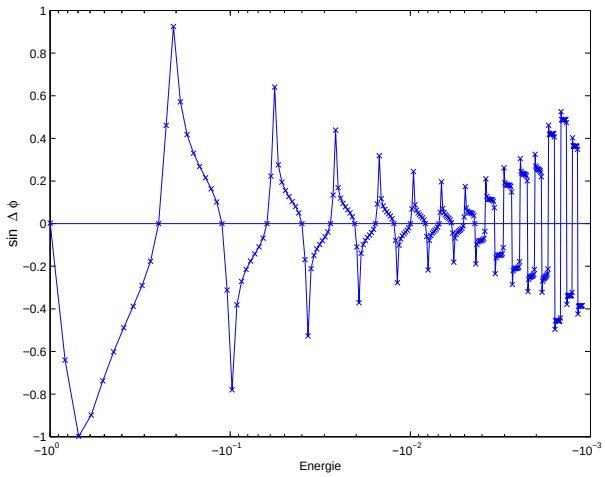


Abbildung 4.5: Nullstellen von $\sin\phi$ sind Energien der gebundenen Zustände.

wir mit dem Sekanten-Verfahren, welches für skalare Gleichungen effektiver ist als das Newton-Verfahren. Wir bestimmen die 10 kleinsten Energien. Für die Intervalle mit Vorzeichenwechsel starten wir eine Iteration mit den beiden Endpunkten des Intervalls, wobei zuerst der Endpunkt mit betragsgrößerem Funktionswert ersetzt wird. Die Iteration konvergiert für gewöhnlich in 4 bis 6 Schritten.

Nr.	Energie	rel. Fehler
1	-0.99919906	$8.009e-04$
2	-0.24999690	$1.240e-05$
3	-0.11111109	$2.334e-07$
4	-0.06250000	$5.889e-09$
5	-0.04000000	$4.346e-09$
6	-0.02777778	$4.889e-09$
7	-0.02040816	$8.696e-09$
8	-0.01562500	$5.365e-08$
9	-0.01234568	$1.303e-07$
10	-0.00999999	$1.353e-06$

(4.79)

Die exakten Werte werden im Rahmen der Fehler in den Randbedingungen sehr gut approximiert. Zur Bestimmung genauerer Lösungen kann zum einen das Integrationsintervall ausgedehnt werden (sowohl in Richtung 0 als auch in Richtung ∞), zum anderen kann die Singularität im Randpunkt auf andere Weise berücksichtigt werden, so zum Beispiel durch eine asymptotische Entwicklung, man siehe auch [60].

Wir verwenden die Randwerte

$$\psi(\rho_{\min}) = \rho_{\min} - \rho_{\min}^2 \qquad \psi(\rho_{\max}) = 1 \tag{4.80}$$

$$\psi'(\rho_{\min}) = 1 - 2\rho_{\min} \qquad \psi(\rho_{\max}) = -\sqrt{-\epsilon} + \frac{1}{\rho_{\max}\sqrt{-\epsilon}}. \tag{4.81}$$

Diese ergeben sich am linken Rand $\rho_{\min}$ einfach durch Berücksichtigung eines weiteren Terms in der Potenzreihenentwicklung. Am rechten Randpunkt $\rho_{\max}$ verwenden wir einen Produktansatz $\psi(\rho) = \rho^\alpha \exp(-\sqrt{-\epsilon}\rho)$, der bei Abgleich der niedrigsten Potenz in ρ auf $\alpha = 1/\sqrt{-\epsilon}$ führt.

Wir erhalten jetzt Ergebnisse, die in der Größenordnung der Toleranz der Magnus-Methode liegen:

Nr.	Energie	rel. Fehler
1	-1.00000000	$2.285e-11$
2	-0.24999982	$7.368e-07$
3	-0.11111111	$1.903e-08$
4	-0.06250000	$1.495e-09$
5	-0.04000000	$2.529e-09$
6	-0.02777778	$3.679e-09$
7	-0.02040816	$6.312e-09$
8	-0.01562500	$4.182e-08$
9	-0.01234568	$6.851e-08$
10	-0.00999999	$7.532e-08$

(4.82)

Es zeigt sich daß die Lösung des Eigenwertproblems unter Nutzung der Magnus-Methode robust und effizient möglich ist. Die Übertragung des obigen Algorithmus auf Mehr-Teilchen-Probleme im Rahmen der Dichtefunktionaltheorie ist Gegenstand aktueller Forschung.

4.5 Zusammenfassung

Die Beschreibung der Struktur fester Materie erfordert die Lösung von Eigenwertproblemen für die stationäre Schrödinger-Gleichung für Mehrteilchensysteme. Der Atomkern wird als fest angenommen (Born-Oppenheimer-Näherung), während die Aufenthaltswahrscheinlichkeit der Elektronen durch die Wellenfunktion beschrieben wird. Die Elektronen besetzen nun die Zustände mit den niedrigsten Energien – gesucht sind also die kleinsten Eigenwerte der stationären Schrödinger-Gleichung. Ein Ansatz mit Kugelflächenfunktionen separiert die radiale Komponente von der Drehimpuls-Komponente. Für die radiale Komponente führen die Eigenwertprobleme auf lineare gewöhnliche Differentialgleichungen mit veränderlichen Koeffizienten, die oszillierende Lösungen besitzen. Wir verwenden zu ihrer Lösung ein problemangepaßtes Verfahren, einen geometrischen Integrator – die Magnus-Methode. Die erforderliche Reihenentwicklung der dexp-Abbildung wird hergeleitet.

Das zu lösende Eigenwertproblem besitzt Singularitäten am Rand. Wir nutzen Reihenentwicklungen verschiedener Aproximationsordnung, um die Resultate sukzessiv zu verbessern. Wir testen verschiedene Varianten der Schrittweitensteuerung für die Magnus-Methode, und verwenden die effektivste zur Lösung des Eigenwertproblems. Die Magnus-Methode zeigt sich für diese Probleme den Standard-Verfahren – seien sie nun implizit oder explizit – als deutlich überlegen.

Kapitel 5

Krylov-Typ Rosenbrock-Methoden für differential-algebraische Systeme vom Index 1

5.1 Steife Anfangswertprobleme

5.1.1 Steifheit

Bei vielen Anwendungen treten sogenannte steife Differentialgleichungen auf. Betrachten wir eine Anfangswertaufgabe

$$\begin{aligned} y'(t) =& f(y(t)), \quad y(t) \in \mathbb{R}^{n_y}, t \in [t_0, t_e] \\ y(t_0) =& y_0. \end{aligned} \tag{5.1}$$

Eigentlich ist bereits der Begriff steife Differentialgleichung ungenau, denn Steifheit stellt ein Phänomen dar, das bei der numerischen Lösung von Anfangswertaufgaben auftritt. So hängt die Steifheit neben der Differentialgleichung auch von der geforderten Genauigkeit in der Lösung, der Länge des Integrationsintervalls sowie unter Umständen auch von den gegebenen Anfangswerten ab.

Steifheit bedeutet, daß in der Lösung neben langsam veränderlichen Komponenten schnell gedämpfte Komponenten auftreten, die für das Problem nicht von Interesse sind. Steife Probleme treten meist im Zusammenhang mit sich sehr schnell einstellenden Gleichgewichten auf. So zum Beispiel in der Reaktionskinetik, wenn langsame Reaktionen zusammen mit schnellen Reaktionen einhergehen, oder im Falle von Ausgleichsprozessen wie Diffusion oder Wärmeleitung.

Schnell gedämpfte Lösungskomponenten entsprechen Eigenwerten in der Jacobi-Matrix mit großem negativen Realteil, Eigenwerte mit großem Imaginärteil führen zu schnellen Oszillationen. Beide Problemklassen führen zu einer großen Lipschitz-Konstanten L der rechten Seite des Systems.

Grundlegend für die Übertragung der Konvergenzanalyse (Schrittweite $h \to 0$) für explizite Runge-Kutta-Verfahren auf endliche Schrittweiten h ist die Annahme, daß $hL = \mathcal{O}(1)$ gilt, denn unter dieser Voraussetzung sind die Verfahren stabil. Für Probleme mit großer Lipschitz-Konstanten L, die aber trotzdem gutartig gestellt sind, sind explizite Verfahren nicht effektiv, da $hL = \mathcal{O}(1)$ die Schrittweiten h stark einschränkt. Man greift daher auf spezielle problemangepaßte Verfahren zurück, im Falle steifer Differentialgleichungen sind dies implizite Runge-Kutta-Verfahren und linear-implizite Runge-Kutta-Verfahren, sowie die hier nicht näher betrachteten BDF-Methoden als Mehrschrittverfahren. Ausführliche Darstellungen und Analysen geeigneter Verfahren findet man in [24], [46], [99].

5.1.2 Stabilität von Einschrittverfahren

Bei der Behandlung steifer Differentialgleichungen sind implizite Verfahren trotz des höheren Aufwands pro Integrationsschritt den expliziten Verfahren überlegen, da sie wesentlich größere Schrittweiten zulassen. Dies liegt an den schlechteren Stabilitätseigenschaften expliziter Verfahren, wie man bereits an der einfachen linearen Testgleichung

$$y' = \lambda y, \quad \lambda \in \mathbb{C} \tag{5.2}$$

nachvollziehen kann. Bei Anwendung eines Runge-Kutta-Verfahrens auf die Gleichung (5.2) mit der Schrittweite h ergibt sich eine Rekursion der Form

$$\begin{aligned} y_{m+1} &= R(h\lambda) y_m \text{ mit} \\ R(z) &= 1 + b^T z (I - zA)^{-1} \mathbb{1}. \end{aligned}$$

Die Funktion $R(z)$ ist rational, für explizite Verfahren ist sie ein Polynom. Die lineare Gleichung (5.2) ist für $\mathrm{Re}\,\lambda \leq 0$ kontraktiv, d.h., für zwei Lösungen $y_1(t), y_2(t)$ gilt

$$|y_1(t+h) - y_2(t+h)| \leq |y_1(t) - y_2(t)| \text{ für } h \geq 0. \tag{5.3}$$

Bei linearen Systemen ist dies äquivalent zu $|y(t+h)| \leq |y(t)|$, und dies fordert man auch von einem Diskretisierungsverfahren, im Idealfall also $|R(z)| \leq 1$ für $\mathrm{Re}\, z \leq 0$.

Die Menge $\{z \in \mathbb{C} : |R(z)| \leq 1\}$ bezeichnet man als Stabilitätsgebiet. Verfahren, deren Stabilitätsgebiet die linke komplexe Halbebene (hier ist gerade die lineare Testgleichung (5.2) stabil) umfaßt, heißen A-stabil. Bei Stabilität in einem Sektor der Größe 2α um die negative reelle Achse spricht man von $A(\alpha)$-Stabilität, und liegt Stabilität nur auf der reellen Achse vor, spricht man von A_0-Stabilität. A-stabile Verfahren mit $R(\infty) = 0$ bezeichnet man als L-stabil. Schärfere Stabilitätsbegriffe legen nichtautonome lineare Systeme (AN-Stabilität) und nichtlineare kontraktive Systeme (B-Stabilität) zugrunde. Ordnungsreduktionsphänomene bei steifen Problemen werden mit Hilfe der B-Konvergenz-Ordnung erfaßt.

Bei differential-algebraischen Systemen von höherem Index gestaltet sich die Übertragung der Stabilitätskonzepte äußerst schwierig, man vergleiche [111], [114]. Für die von uns zu betrachtenden DAEs vom Index 1 in Hessenbergform beschränken wir uns auf das Konzept der A-Stabilität, um effiziente Verfahren auszuwählen.

5.1.3 Implizite Runge-Kutta-Verfahren

Bei expliziten Einschritt-Verfahren ist die Stabilitätsfunktion $R(z)$ ein Polynom, daher kann ein solches Verfahren nicht A-stabil sein. Implizite Verfahren besitzen rationale Stabilitätsfunktionen, die Approximationen an die Exponentialfunktion (Lösung von (5.2)) darstellen. Für s-stufige Verfahren erhält man im Zähler und Nenner Polynome vom Maximalgrad s. Für die Gauß- bzw. Radau-Verfahren [46] ergeben sich die (s,s)- bzw. $(s-1,s)$-Padé-Approximationen an die Exponentialfunktion als Stabilitätsfunktionen. Beide sind bekanntermaßen A-stabil. Die $(s-1,s)$-Padé-Approximation ist zudem auch L-stabil, da der Nennergrad größer als der Zählergrad ist und somit $R(\infty) = 0$ gilt.

Bei impliziten Runge-Kutta-Verfahren müssen die Verfahrensgleichungen iterativ gelöst werden. Eine einfache Funktionaliteration verschenkt den gewonnenen Vorteil, da Kontraktivität gleichbedeutend mit einer Einschränkung $hL < C$ an die Schrittweite ist – aber L ist für steife Systeme eben sehr groß. Typischerweise wird ein vereinfachtes Newton-Verfahren zur Lösung der Verfahrensgleichungen verwendet, wobei die Jacobi-Matrix $f_y(y)$ zumeist am Startwert $y = y_n$ approximiert wird

oder (insofern das Konvergenzverhalten der Iteration zufriedenstellend ist) eine Jacobi-Matrix aus vorhergehenden Schritten verwendet wird.

Dies führt in jedem Newton-Schritt auf ein lineares Gleichungssystem der Dimension $n_y \times s$, was für große Systeme mit einem hohen Aufwand verbunden ist. Es liegt daher nahe, die Verfahrensvorschrift zu modifizieren, um die guten Stabilitätseigenschaften mit effizient auflösbaren Verfahrensgleichungen zu kombinieren. Ein erster Schritt in diese Richtung ist der Übergang zu diagonal-impliziten Verfahren, man siehe [2], [42], [20], mit den Stufen

$$Y_{mi} = y_m + h \sum_{j=1}^{i} a_{ij} f(Y_{mj}). \tag{5.4}$$

Die einzelnen Stufengleichungen können nun sukzessive gelöst werden. Ein vereinfachter Newton-Schritt für das Inkrement $g_i := Y_{mi} - y_m$ in der i-ten Stufe ergibt sich mit $J \approx f_y(y_m)$ zu

$$-(I - h a_{ii} J)(g_i^{(l+1)} - g_i^{(l)}) = g_i^{(l)} - h \sum_{j=1}^{i-1} a_{ij} f(Y_{mj}) - h a_{ii} f(y_m + g_i^{(l)}) \tag{5.5}$$

Es sind somit sukzessive s nichtlineare Systeme der Dimension n_y zu lösen. Wählt man die Diagonalelemente a_{ii} der Verfahrensmatrix alle gleich, so ist nur eine LU-Zerlegung pro Schritt erforderlich. Diese Verfahren heißen SDIRK-Verfahren (**s**ingly **d**iagonally **i**mplicit **R**unge-**K**utta).

5.2 Linear-implizite Verfahren

Bei der Implementierung impliziter Verfahren werden die Verfahrensgleichungen iterativ mit dem vereinfachten Newtonverfahren gelöst, wobei die Iteration zweckmäßigerweise abgebrochen wird, sobald die Genauigkeit der Lösung im Bereich der geforderten Toleranz (oder um einen bestimmten Faktor darunter) liegt. Dabei wird das Konvergenzverhalten von der Schrittweite abhängen, mit kleinerer Schrittweite verfügt man außerdem über einen genaueren Startwert.

Die Konvergenz der vereinfachten Newton-Iteration kann somit durch die Schrittweite h gesteuert werden. Man fixiert einfach eine Newton-Iteration pro Stufe — und gelangt zu linear-impliziten Verfahren. Wählt man $J = f_y(y_m)$, so bezeichnet man die Verfahrensklasse als Rosenbrock-Methoden [85, 107, 46, 99]. Einen allgemeineren Ansatz bieten die von Strehmel und Weiner entwickelten adaptiven Runge-Kutta-Methoden, die sich durch gute Stabilitätseigenschaften auszeichnen [95, 97, 8, 98, 99].

5.2.1 Rosenbrock-Methoden

Mit der Jacobi-Matrix $J = f_y(y_m)$ ergibt sich ein vereinfachter Newton-Schritt für die internen Ableitungen $k_i = f(y_m + h \sum_j a_{ij} k_j)$ zu

$$(I - h\gamma J)(k_i^{(1)} - k_i^{(0)}) = -k_i^{(0)} + f(y_m + h \sum_{j=1}^{i-1} a_{ij} k_j + h\gamma k_i^{(0)}). \tag{5.6}$$

Bei der Wahl des Startwertes $k_i^{(0)}$ bietet sich eine Kombination der vorhergehenden Stufen an. Wir erhalten somit als Stufenlösung $k_i = k_i^{(1)}$ nach einem Newtonschritt

$$k_i^{(0)} = -\sum_{j=1}^{i-1} \frac{\gamma_{ij}}{\gamma} k_j \Rightarrow$$

$$(I - h\gamma J)(k_i + \sum_{j=1}^{i-1} \frac{\gamma_{ij}}{\gamma} k_j) = f(y_m + h\sum_{j=1}^{i-1}(a_{ij} - \gamma_{ij})k_j) + \sum_{j=1}^{i-1} \frac{\gamma_{ij}}{\gamma} k_j. \tag{5.7}$$

Die Verfahrensvorschrift ergibt sich mit $\alpha_{ij} = a_{ij} - \gamma_{ij}$, $\underline{\mathrm{k}}_i := -k_i^{(0)}$ und der üblichen Aufdatierung mit Gewichten b_i zu

$$Y_{mi} = y_m + h\sum_{j=1}^{i-1} \alpha_{ij} k_j \tag{5.8a}$$

$$(I - h\gamma J)(k_i + \underline{\mathrm{k}}_i) = f(Y_{mi}) + \underline{\mathrm{k}}_i \tag{5.8b}$$

$$y_{m+1} = y_m + h\sum_{i=1}^{s} b_i k_i. \tag{5.8c}$$

Rosenbrock-Methoden erweisen sich als sehr effiziente Verfahren zur Lösung großer steifer Differentialgleichungen. Sie reproduzieren näherungsweise die guten Stabilitätseigenschaften impliziter Verfahren durch eine sehr einfache Verfahrensvorschrift. Außerdem haben sie den Vorteil, leicht implementierbar zu sein. Einige Abstriche müssen in der B-Konvergenzordnung [96] gemacht werden, da sie Abkömmlinge einfach diagonal-impliziter Verfahren sind, die naturgemäß nicht die vereinfachende Bedingung $C(2)$ erfüllen können. Im Code ROS4 [46] ist mit der Methode GRK4T von Kaps/Rentrop [59] ein sehr effizientes Verfahren vierter Ordnung implementiert.

Rosenbrock-Methoden können außerdem effizient zur Lösung differential-algebraischer Systeme eingesetzt werden, so für Probleme vom Index 1 in [83], [82], [84] und [3], und direkt für die Index-3-Formulierung bei mechanischen Mehrkörpersystemen in [112], [113], [114], [115].

5.2.2 W-Methoden

In vielen Anwendungen kann man die steifen Komponenten der Lösung lokalisieren. Der Aufwand zur Lösung der linearen Gleichungssysteme, wie sie bei Rosenbrock-Methoden auftreten, kann entscheidend gesenkt werden, wenn man nicht die komplette Jacobi-Matrix, sondern nur eine Approximation T von niederem Rang verwendet, so daß lineare Gleichungssysteme mit der Matrix $(I - h\gamma T)$ einfach zu lösen sind. Die Matrix J in (5.6) wird dann durch die Matrix T ersetzt. Ordnungsbedingungen für die so erhaltenen W-Methoden [93] werden unabhängig von der Matrix T formuliert, man erhält zu den Ordnungsbedingungen für Rosenbrock-Methoden zusätzliche Konsistenzbedingungen. Für den speziellen Fall $T = f_y(y_m) + \mathcal{O}(h)$ kann mit einer kleineren Menge zusätzlicher Konsistenzbedingungen gearbeitet werden. Dieser Fall bildet gleichzeitig die theoretische Grundlage für die bei der Implementierung von Rosenbrock-Methoden zumeist verwendete Strategie, die gleiche Jacobi-Matrix für mehrere aufeinanderfolgende Schritte zu verwenden.

W-Methoden bieten außerdem eine einfache Möglichkeit zur Konstruktion partitionierter Verfahren [109], man setzt in der Jacobi-Matrix einfach gewisse Blöcke zu Null.

5.2.3 Krylov-ROW-Methoden

Bei vielen Anwendungen ist nur eine gewisse Teilmenge der Lösungskomponenten steif. Um den Aufwand zur Lösung der linearen Gleichungssysteme bei Rosenbrock-Methoden zu senken, kann man zum einen partitionierte Verfahren über W-Methoden, wie oben beschrieben, konstruieren. Dies erfordert allerdings die explizite Kenntnis der steifen und nichtsteifen Komponenten.

Einen Ansatz zur automatischen Partitionierung bieten Krylov-ROW-Methoden. Wie bei den W-Methoden wird J in (5.8b) zunächst durch eine Approximation T von niederem Rang ersetzt. Allerdings lassen wir jetzt unterschiedliche Matrizen $T = T_i$ in den Stufen $i = 1, \ldots, s$ zu. Die Matrizen T_i sind Krylovraum-Approximationen (siehe [104], [86], [101]) an J, und somit gilt im allgemeinen $T_i = J + \mathcal{O}(1)$. Durch geeignete Approximation kann jedoch erreicht werden, daß keine zusätzlichen Konsistenzbedingungen im Vergleich zum Fall $T_i = J$ (Rosenbrock-Methode) auftreten.

Ausgangspunkt der Konstruktion ist eine Rosenbrock-Methode. Dort ist in jeder Stufe i ein lineares Gleichungssystem der Form $(I - h\gamma J)x_i = w_i$ zu lösen. Löst man dieses näherungsweise mit FOM (siehe Abschnitt 5.3.2), so ergibt sich die Lösung x_i im (niedrig-dimensionalen) Krylovraum $\mathcal{K}_{\kappa_i}(J, w_i) \ni w_i$ durch

$$(I - hQ_{\kappa_i}Q_{\kappa_i}^T J)x_i = w_i, \tag{5.9}$$

wobei die Matrix Q_{κ_i} durch einen Arnoldi-Prozeß generiert wird (siehe Abschnitt 5.3.1). Eine Krylov-ROW-Methode ist somit gegeben durch

$$Y_{mi} = y_m + h\sum_j \alpha_{ij} k_j \tag{5.10a}$$

$$(I - h\gamma Q_{\kappa_i}Q_{\kappa_i}^T J)\,(k_i + \underline{\mathbf{k}}_i) = f(Y_{mi}) + \underline{\mathbf{k}}_i \tag{5.10b}$$

$$y_{m+1} = y_m + h\sum_i b_i k_i. \tag{5.10c}$$

Die Koeffizienten ergeben sich dabei aus der zugrundeliegenden Rosenbrock-Methode. Die reduzierte Jacobi-Matrix bezeichnen wir mit $T_i = Q_{\kappa_i}Q_{\kappa_i}^T J$.

Schmitt und Weiner [91] zeigen, daß unter geeigneten Voraussetzungen die Krylov-ROW-Methode dieselbe Konvergenzordnung wie die zugrundeliegende Rosenbrock-Methode hat.

Satz 5.2.1. *Seien y_{m+1}, k_i, Y_{mi} mit der Krylov-ROW-Methode berechnet, und $\tilde{y}_{m+1}, \tilde{k}_i, \tilde{Y}_{mi}$ die entsprechenden Werte der Rosenbrock-Methode. Es sei*

$$(T_i^l - J^l)w_i = \mathcal{O}\left(h^{p-l}\right),\ l = 1, \ldots, p-1,\ i = 1, \ldots, s. \tag{5.11}$$

Dann gilt

$$k_i - \tilde{k}_i = \mathcal{O}(h^p)$$
$$y_{m+1} - \tilde{y}_{m+1} = \mathcal{O}\left(h^{p+1}\right).$$

Für einen detaillierten Beweis sowie mögliche Abschwächungen der Voraussetzungen verweisen wir auf [91]. Die Grundidee des Beweises besteht darin, zur Abschätzung der Differenz zwischen Krylov- und Rosenbrock-Lösung die Inversen von $(I - h\gamma J)$ und $(I - h\gamma T_i)$ in Neumann-Reihen zu entwickeln und unter Ausnutzung von (5.11) deren Differenz abzuschätzen.

Nach obiger Darstellung werden die Stufengleichungen unabhängig voneinander durch FOM gelöst. Diese Vorgehensweise ist unbefriedigend, denn die Krylov-Räume der ersten Stufen enthalten

Informationen über die Eigenraumstruktur der Jacobi-Matrix, die in weiteren Stufen genutzt werden kann. Für effiziente Implementierungen versucht man daher, den im ersten Schritt gewonnenen Krylovraum sukzessive mit weiteren Stufen zu erweitern. Dabei fordert man:

- In Stufe i suchen wir eine Näherungslösung der Stufengleichung $(I - h\gamma J)x_i = w_i$ in einem κ_i-dimensionalen Raum $\mathcal{K}_{\kappa_i} \ni w_i$ (nicht notwendigerweise ein Krylovraum!).
- Der Raum $\mathcal{K}_{\kappa_i}$ werde durch die Spalten der orthogonalen Matrix Q_{κ_i} aufgespannt.
- Zur Bestimmung der Lösung nutzen wir eine Galerkin-Bedingung (FOM), d.h., das Residuum sei orthogonal zu $\mathcal{K}_{\kappa_i}$.

Die aufspannenden Matrizen Q_i können mit einem mehrfachen Arnoldi-Prozeß berechnet werden, man siehe Abschnitt 5.3.1.

Die Implementierung des mehrfachen Arnoldi-Prozeß für Krylov-ROW-Methoden durch Weiner, Schmitt und Podhaisky im Code ROWMAP [110] resultiert in einem effizienten und robusten Löser für große Systeme steifer Differentialgleichungen. Weitere Anwendungen von Krylov-Unterraum-Techniken finden sich in den Arbeiten von Lubich und Hochbruck zu exponentiellen Integratoren, die speziell für steife und stark oszillierende Probleme konstruiert sind, man siehe [52], [53], [54]. Die Nutzung iterativer Lösungstechniken für BDF-Verfahren wurde von Brown, Hindmarsh und Petzold in [7] beschrieben, eine Implementierung findet sich im Code VODPK.

5.3 Iterative Lösung linearer Gleichungssysteme

Wir wollen in diesem Abschnitt kurz die für die Krylov-ROW-Methoden benötigten Techniken zur iterativen Lösung linearer Gleichungssysteme umreißen. Für ausführlichere Darstellungen verweisen wir auf die Literatur [104], [86], [101].

5.3.1 Der Arnoldi-Prozeß

Gesucht sei eine Faktorisierung $A = QHQ^T$ mit einer Hessenbergmatrix H und einer orthogonalen Matrix Q. Bei der orthogonalen Strukturierung werden orthogonale Transformationsmatrizen auf A angewandt, und schrittweise die strukturierte Matrix H erzeugt. Umgekehrt führt eine strukturierte Orthogonalisierung zur vollständigen Arnoldi-Iteration [101].

Mit $AQ = QH$ ergibt sich für die Spalten q_k von Q gerade

$$Aq_k = \sum_{j=1}^{k+1} h_{jk} q_j, \tag{5.12}$$

wobei h_{jk} die Einträge der Hessenbergmatrix H sind. Beim Arnoldi-Prozeß nutzt man (5.12) als Ansatz für q_{k+1} und orthogonalisiert wie bei der modifizierten Gram-Schmidt-Orthogonalisierung [61] q_{k+1} gegen $q_1, \ldots, q_k$ mit anschließender Normierung. Der Arnoldi-Prozeß berechnet auf diese Art eine Gram-Schmidt-Orthogonalisierung der Krylov-Folge $q_1, Aq_1, A^2q_1, \ldots$. Das Verfahren bricht ab, wenn der Krylovraum degeneriert, d.h., wenn $A^k q_1$ linear abhängig von $q_1, \ldots, A^{k-1}q_1$ ist.

Nach n Schritten ergibt sich mit $(q_1, \ldots, q_n)$ eine orthogonale Basis des Krylovraums

$$\mathcal{K}_n = \mathcal{K}_n(A, q_1) := \mathrm{span}\{q_1, Aq_1, \ldots A^{n-1}q_1\}. \tag{5.13}$$

Bezeichnen wir mit Q_n die aus den ersten q_n orthogonalen Vektoren bestehende Matrix, mit $H_n \in \mathbb{R}^{n\times n}$ bzw. $H_{n+1,n} \in \mathbb{R}^{(n+1)\times n}$ linke obere Blöcke von H, so ergibt sich

$$AQ_n = Q_{n+1}H_{n+1,n} \tag{5.14}$$

$$Q_n^T AQ_n = H_n. \tag{5.15}$$

Der Orthoprojektor in den Krylovraum ist durch $Q_nQ_n^T$ gegeben.

Wir erwähnen noch kurz den mehrfachen Arnoldi-Prozeß wie er im Code ROWMAP [110] implementiert ist, für eine ausführliche Darstellung verweisen wir auf die Arbeiten [91], [110]. Beim Aufbau des Krylovraumes wird dieser nach κ_1 Schritten nicht durch Aq_{κ_1} erweitert, sondern durch einen beliebigen Vektor w_2. Wir orthogonalisieren dann w_2 gegen $\mathcal{K}_{\kappa_1}$ und erhalten q_{κ_1+1}. Die folgenden Erweiterungen des Raumes ergeben sich dann durch Aq_{κ_1}, woraus sich nach Orthogonalisierung q_{κ_1+2} ergibt, usw., d.h., Aq_{κ_1+j} wird gegen $q_1, \ldots, q_{\kappa_1+j+1}$ orthogonalisiert, woraus sich dann q_{κ_1+j+2} ergibt. Im Prinzip ergibt sich eine Art Shift, Aq_j wird (nach Orthogonalisierung) nicht direkt nach q_j eingefügt, sondern in der i-ten Stufe direkt nach q_{j+i-1} (wobei wir den allgemeinen Fall voraussetzen, daß die rechten Seiten w_i der Stufen nicht im bereits generierten Ansatzraum enthalten sind).

In der Matrix $Q_{\kappa_i}^T AQ_{\kappa_i}$ füllen sich durch den Shift weitere Nebendiagonalen auf, für jede Stufe $i > 1$ kommt ab Zeile κ_i eine weitere Nebendiagonale hinzu.

5.3.2 FOM

Ein lineares Gleichungssystem $Ax = b$, $A \in \mathbb{R}^{m\times m}$, kann näherungsweise gelöst werden, indem es in den Krylovraum $\mathcal{K}_n(A, b)$ projiziert wird. Dies bedeutet, daß eine Näherungslösung x im Krylovraum gesucht ist, so daß die orthogonale Projektion des Residuums in den Krylovraum verschwindet. Das so definierte Verfahren heißt **F**ully **O**rthogonal **M**ethod. Bezeichnen wir die mit dem Arnoldi-Prozeß nach n Schritten berechnete Orthonormalbasis des Krylovraumes mit Q_n, so bestimmt sich die so definierte Näherungslösung x_n durch

$$Q_n^T(Ax_n - b) = 0 \text{ mit } x_n = Q_n w_n. \tag{5.16}$$

Daraus ergeben sich w_n und x_n zu

$$H_n w_n = Q_n^T b \tag{5.17}$$

$$x_n = Q_n H^{-1} Q_n^T b. \tag{5.18}$$

Dies kann auch so formuliert werden: A wird durch die niedrigdimensionale Approximation $Q_nH_nQ_n^T$ ersetzt, und für das resultierende singuläre Gleichungssystem wird die Minimum-Norm-kleinste-Quadrate-Lösung verwendet. Diese löst hier das Gleichungssystem exakt, da b zum Spaltenraum von Q_n gehört.

5.3.3 GMRES

Ein weitere Möglichkeit zur iterativen Lösung eines linearen Gleichungssystems $Ax = b$ stellt GMRES (**G**eneralized **M**inimum **R**esidual) dar. Zum Krylovraum $\mathcal{K}_n(A, b)$ bestimmt man die Lösung $x_n \in \mathcal{K}_n(A, b)$ durch die Forderung

$$\|Ax_n - b\|_2 \to min. \tag{5.19}$$

Mit dem Ansatz $x_n = Q_n w_n$ ergibt sich

$$\begin{aligned}\|AQ_n w_n - b\|_2 =& \|Q_{n+1} H_{n+1,n} w_n - b\|_2 \\ =& \|H_{n+1,n} w_n - Q_{n+1}^T b\|_2,\end{aligned}$$

man erhält w_n (und somit die Näherung $x_n = Q_n w_n$) als Lösung des Quadratmittelproblems

$$\|H_{n+1,n} w_n - Q_{n+1}^T b\|_2 \to \min . \tag{5.20}$$

Läßt man die letzte Komponente weg, so erhält man das lösbare Gleichungssystem (5.17).

Ist eine größere Anzahl von Iterationen durchzuführen, so erweist sich ein Neustart des Verfahrens nach einer gewissen Zahl von Iterationen als vorteilhaft, denn der Aufwand für die Orthogonalisierung wächst von Schritt zu Schritt linear an. Im Falle eines Neustarts wird dann das zuletzt berechnete Residuum als Startwert verwendet.

5.4 Krylov-Typ Rosenbrock-Methoden für DAE's vom Index 1

Wir betrachten ein differential-algebraisches System vom Index 1 in Hessenbergform:

$$y' = f(y,z),\ y \in \mathbb{R}^{n_y},\ z \in \mathbb{R}^{n_z}, \tag{5.21a}$$

$$0 = g(y,z). \tag{5.21b}$$

Ist die Jacobi-Matrix g_z regulär, so ist obiges System vom Index 1. Die Nebenbedingung kann in einer Umgebung der Lösung nach $z = G(y)$ aufgelöst werden. Einsetzen in die Differentialgleichung führt zu einer gewöhnlichen Differentialgleichung. Wir setzen voraus, daß die entstehende Differentialgleichung $y' = f(y, G(y))$ steif ist. Solche Systeme entstehen insbesondere bei der Semidiskretisierung parabolischer Probleme, und sie sind dann von hoher Dimension. Oft ist ein direkter Einbau der Randbedingungen nicht zweckmäßig, so daß diese als algebraische Nebenbedingungen an das System gekoppelt werden. Im letzten Abschnitt dieses Kapitels werden wir mit den Problemen der Viskoelastizität eine Anwendungsklasse kennenlernen, die auf große Systeme vom Index 1 mit einer großen Zahl von Nebenbedingungen führt.

5.4.1 Adaption auf DAEs

Bei der Herleitung von Verfahren für differential-algebraische Systeme bieten sich prinzipiell zwei Vorgehensweisen an, die als direkter und indirekter Zugang bezeichnet werden.

Der indirekte Zugang

Der indirekte Zugang ist auf semi-explizite Index-1-Systeme beschränkt. Die Nebenbedingung (5.21b) wird zu $z = G(y)$ nach z aufgelöst und in die Differentialgleichung (5.21a) eingesetzt. Diese gewöhnliche Differentialgleichung wird dann mit dem Grundverfahren gelöst. Für eine s-stufige Rosenbrock-Methode (5.8) ist $f(Y_{mi})$ durch $f(Y_{mi}, G(Y_{mi}))$ zu ersetzen. Die Jacobi-Matrix der gewöhnlichen Differentialgleichung ergibt sich mit $g(y, G(y)) = 0$ zu $J = f_y + f_z G_y = f_y - f_z g_z^{-1} g_y$. Damit wird

(5.8) zu

$$y_{m+1} = y_m + h\sum_i b_i k_i, \quad 0 = g(y_{m+1}, z_{m+1}) \qquad \text{mit} \tag{5.22a}$$

$$Y_{mi} = y_m + h\sum_{j=1}^{i-1} \alpha_{ij} k_j, \quad 0 = g(Y_{mi}, Z_{mi}), \qquad i = 1, \ldots, s, \tag{5.22b}$$

$$\left(I - h\gamma(f_y - f_z g_z^{-1} g_y)\right)(k_i + \underline{\mathrm{k}}_i) = f(Y_{mi}, Z_{mi}) + \underline{\mathrm{k}}_i, \qquad i = 1, \ldots, s. \tag{5.22c}$$

Die Bestimmung der Umkehrfunktion $z = G(y)$ wird durch die implizite Relation in den Gleichungen (5.22b), (5.22a) realisiert. Man kann das so konstruierte Verfahren aber auch mit der Original-Jacobi-Matrix schreiben. Gleichung (5.22c) muß dann durch die äquivalente Gleichung

$$\left[\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} - h\gamma \begin{pmatrix} f_y & f_z \\ g_y & g_z \end{pmatrix}\right] \begin{pmatrix} k_i + \underline{\mathrm{k}}_i \\ l_i \end{pmatrix} = \begin{pmatrix} f(Y_{mi}, Z_{mi}) + \underline{\mathrm{k}}_i \\ 0 \end{pmatrix} \tag{5.23}$$

ersetzt werden. Die Variablen l_i sind lediglich Dummy-Variablen, sie werden nicht weiter verwendet.

Der direkte Zugang

Beim direkten Zugang wird das System formal regularisiert durch

$$\begin{aligned} y' &= f(y, z) \\ \varepsilon z' &= g(y, z). \end{aligned} \tag{5.24}$$

Auf das ODE-System (5.24) wird die Verfahrensvorschrift angewandt, und dann der Grenzübergang für $\varepsilon \to 0$ durchgeführt. Untersuchungen zur Konvergenz so konstruierter Methoden findet man in einer ganzen Reihe von Arbeiten, man siehe [83], [82], [84] und [3].

Bemerkung 5.4.1. *Für implizite Runge-Kutta-Verfahren sind direkter und indirekter Zugang nahezu äquivalent, man vergleiche [43, 46]. Der direkte Zugang führt zu Gleichungen $0 = g(Y_{mi}, Z_{mi})$ für die internen Werte Y_{mi}, Z_{mi}, und dies ist de facto eine Auflösung der Nebenbedingungen nach der Variablen z. Die Lösungen stimmen in y_{m+1}, Y_{mi}, Z_{mi} überein, lediglich die z_{m+1} können unterschiedlich sein, da sich beim direkten Zugang $z_{m+1} = R(\infty) z_m + \sum_i b_i \omega_{ij} Z_{mj}$ ergibt, während sich beim indirekten Zugang z_{m+1} aus $0 = g(y_{m+1}, z_{m+1})$ bestimmt. Für steif-genaue Verfahren stimmt aber auch dies überein.*

Das Resultat beim direkten Zugang ergibt sich wie folgt:

$$y_{m+1} = y_m + h\sum_{i=1}^{s} b_i k_i \qquad z_{m+1} = z_m + h\sum_{i=1}^{s} b_i l_i. \tag{5.25a}$$

$$Y_{mi} = y_m + h\sum_{j=1}^{i-1} \alpha_{ij} k_j \qquad Z_{mi} = z_m + h\sum_{j=1}^{i-1} \alpha_{ij} l_j \tag{5.25b}$$

$$\left[\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} - h\gamma \begin{pmatrix} f_y & f_z \\ g_y & g_z \end{pmatrix}\right] \begin{pmatrix} k_i + \underline{\mathrm{k}}_i \\ l_i + \underline{\mathrm{l}}_i \end{pmatrix} = \begin{pmatrix} f(Y_{mi}, Z_{mi}) + \underline{\mathrm{k}}_i \\ g(Y_{mi}, Z_{mi}) \end{pmatrix} \tag{5.25c}$$

Die Größen $\underline{\mathrm{l}}_i$ sind analog zu den $\underline{\mathrm{k}}_i$ durch $\underline{\mathrm{l}}_i = \sum_{j=1}^{i-1} \gamma_{ij}/\gamma l_j$ gegeben. Ein Vorteil des direkten Zugangs ist, daß die Komponente z nicht durch Auflösung der Nebenbedingung bestimmt werden muß.

Im Gegensatz zum indirekten Zugang erfolgt hier eine Aufdatierung der Komponente z mit internen Stufen, die Nebenbedingung $g(y,z)=0$ ist nur in der Größenordnung der Diskretisierungsfehler des Verfahrens erfüllt. Um die Stabilität der Aufdatierung für z zu garantieren, wird in typischen Konvergenzaussagen $|R(\infty)|<1$ gefordert [46]. Die Koeffizientenmatrix des linearen Gleichungssystems (5.25c) für die internen Steigungen k_i, l_i stimmt mit der des Gleichungssystems (5.23) für die internen Steigungen beim indirekten Zugang überein, lediglich die rechten Seiten unterscheiden sich, da in (5.25) noch interne Steigungen für die algebraischen Variablen berechnet werden müssen.

5.4.2 Krylov-ROW-Methoden für DAEs

Wir können zum einen einfach die Krylov-ROW-Methoden mit den bereits erwähnten Techniken geeignet verallgemeinern, um sie auf DAEs anwenden zu können. Zum anderen können wir aber auch wieder einen Schritt zurück zu den Rosenbrock-Methoden gehen, diese auf DAEs erweitern, und die linearen Gleichungssysteme iterativ lösen. Für eine Überblicksdarstellung der beiden Vorgehensweisen siehe man [121].

Der indirekte Zugang

Unsere ODE hat jetzt die Gestalt $y'=f(y,G(y))$, wobei $G(y)$ implizit durch $g(y,G(y))=0$ definiert ist. Wir gehen hier davon aus, daß im Verfahren diese Nebenbedingung stets mit hinreichender Genauigkeit aufgelöst wird.

Mit der Jacobi-Matrix $J=f_y+f_zG_y=f_y-f_zg_z^{-1}g_y$ können wir dann das Rosenbrock-Verfahren formulieren und zur Krylov-ROW-Methode übergehen. Ebenso kann man direkt in die Krylov-ROW-Methode für ODEs die Gleichung $y'=f(y,G(y))$ einsetzen, das Diagramm

$$\begin{array}{ccc} \text{ROW(ODE)} & \xrightarrow{\text{indirekt}} & \text{ROW(DAE)} \\ \Big\downarrow \text{Krylov} & & \Big\downarrow \text{Krylov} \\ \text{Krylov(ODE)} & \xrightarrow{\text{indirekt}} & \text{Krylov(DAE)} \end{array} \tag{5.26}$$

kommutiert im Falle des indirekten Zugangs.

Die komplette Verfahrensvorschrift ergibt sich aus (5.22), indem in (5.22c) die Jacobi-Matrix $J=f_y-f_zg_z^{-1}g_y$ durch $Q_iQ_i^TJ$ ersetzt wird.

Der direkte Zugang für die Rosenbrock-Methode

Beim direkten Zugang müssen wir die Reihenfolge der Operationen beachten. Das folgende Diagramm verdeutlicht noch einmal die zwei möglichen Wege:

$$\begin{array}{ccc} \text{ROW}(\varepsilon\text{-ODE}) & \xrightarrow{\text{direkt}} & \text{ROW(DAE)} \\ \Big\downarrow \text{Krylov} & & \Big\downarrow \text{Krylov} \\ \text{Krylov}(\varepsilon\text{-ODE}) & \xrightarrow{\text{direkt}} & \text{Krylov(DAE)} \end{array} \tag{5.27}$$

Zu Beginn wird stets die DAE über die ε-Einbettung in eine ODE überführt und die Rosenbrock-Methode angewandt. Wird jetzt zunächst der Grenzübergang $\varepsilon\to 0$ durchgeführt, so gelangt man zu einer Rosenbrock-Methode für DAEs. Die auftretenden linearen Gleichungssysteme können dann mit Krylov-Techniken gelöst werden – dies ist der rechte obere Weg in (5.27). Auf dem Weg links-unten

wird zuerst die Krylov-Technik auf die singulär gestörte ODE angewandt, und dann im resultierenden Verfahren der Grenzübergang $\varepsilon \to 0$ vollzogen. Wir werden beide Ansätze in den nächsten Abschnitten analysieren.

Die Krylov-Lösung von ROW(DAE)

Wir setzen also zuerst $\varepsilon = 0$, und wenden dann das Krylov-Verfahren an. Mit (5.8) ergibt sich für das regularisierte System, nachdem $\varepsilon = 0$ gesetzt wurde,

$$\left(\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} - h\gamma \begin{pmatrix} f_y & f_z \\ g_y & g_z \end{pmatrix} \right) \begin{pmatrix} k_i + \underline{\mathrm{k}}_i \\ l_i + \underline{\mathrm{l}}_i \end{pmatrix} = \begin{pmatrix} f(Y_{mi}, Z_{mi}) + \underline{\mathrm{k}}_i \\ g(Y_{mi}, Z_{mi}) + 0 \end{pmatrix}. \tag{5.28}$$

Beim Übergang zur Krylov-ROW-Methode müssen wir die Jacobi-Matrix durch eine geeignete niedrigdimensionale Approximation ersetzen. Ziel ist, wie bei gewöhnlichen Differentialgleichungen, daß die Krylov-Lösung hinreichend nahe bei der ROW-Lösung liegt. Wir können uns dann auf die Konsistenztheorie für ROW-Methoden zurückziehen. Bei ODEs ist diese Herangehensweise von Erfolg gekrönt – die Matrix des linearen Gleichungssystems ist von der Form $I + \mathcal{O}(h)$. Ein iteratives Verfahren erhöht daher in jedem Schritt die Approximationsordnung der Inversen durch den Krylovraum um eins.

Hier liegt leider ein anderer Fall vor – nur im Unterraum der differentiellen Komponenten ist die Jacobi-Matrix von der Form $I + \mathcal{O}(h)$. Im Unterraum der algebraischen Komponenten liegt allgemein die Matrix g_z vor – ohne weitere Informationen über diese lassen sich keine Aussagen zum Konvergenzverhalten machen.

Man kann nun also auf verschiedene Arten weiter fortfahren:

KDGL In den algebraischen Gleichungen wird keine Dimensionsreduktion durchgeführt, diese werden in jedem Schritt exakt gelöst. Dieser Fall war schon beim indirekten Zugang diskutiert worden. Er ist für $n_y \gg n_z$ sicher praktikabel, man kann im wesentlichen auf die Krylov-Verfahren zurückgreifen.

KDAE Man wendet die Krylovraum-Technik auf das System (5.28) an.

PREC Insbesondere für die algebraischen Gleichungen wird ein Vorkonditionierer verwendet.

Letzteres erscheint als beste Lösung, führt aber über den Rahmen dieses Kapitels hinaus.

Nach Vorschlag **KDAE** ergibt sich durch Elimination der algebraischen Variablen aus der oberen Gleichung in (5.28) mit $f_i = f(Y_{mi}, Z_{mi})$ und $g_i = g(Y_{mi}, Z_{mi})$ wie in (5.10b)

$$\begin{pmatrix} I - hQ_iQ_i^T(f_y - f_z g_z^{-1} g_y) & 0 \\ -hg_y & -hg_z \end{pmatrix} \begin{pmatrix} k_i + \underline{\mathrm{k}}_i \\ l_i + \underline{\mathrm{l}}_i \end{pmatrix} = \begin{pmatrix} f_i - f_z g_z^{-1} g_i + \underline{\mathrm{k}}_i \\ g_i \end{pmatrix}.$$

Der obere Block des Systems, der der differentiellen Gleichung entspricht, wird iterativ mit einer Krylov-Methode gelöst. Wir setzen dabei analog zum Standard-ODE-Fall $f_i - f_z g_z^{-1} g_i + \underline{\mathrm{k}}_i$ im Spaltenraum von Q_i enthalten voraus.

Die Güte des Vorschlags **KDAE** hängt sicher vom jeweiligen System ab. Wir werden ihn später im numerischen Experiment untersuchen.

ε-Einbettung der Krylov-Methode

Die Krylov-Methode ist für eine explizite Differentialgleichung formuliert. Durch den direkten Zugang wird aus unserer DAE formal das singulär gestörte Problem

$$\begin{aligned} y' =& f(y,z) \\ z' =& 1/\varepsilon g(y,z). \end{aligned} \tag{5.29}$$

Die Stufen der Krylov-Methode ergeben sich zu

$$\left(\begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} - h\gamma \begin{pmatrix} f_y & f_z \\ \frac{1}{\varepsilon} g_y & \frac{1}{\varepsilon} g_z \end{pmatrix} \right) \begin{pmatrix} k_i + \underline{\mathrm{k}}_i \\ l_i + \underline{\mathrm{l}}_i \end{pmatrix} = \begin{pmatrix} f(Y_{mi}, Z_{mi}) \\ \frac{1}{\varepsilon} g(Y_{mi}, Z_{mi}) \end{pmatrix} + \begin{pmatrix} \underline{\mathrm{k}}_i \\ \underline{\mathrm{l}}_i \end{pmatrix}. \tag{5.30}$$

Das Gleichungssystem (5.30) der Form $A(\varepsilon)x = b(\varepsilon)$ wird nun iterativ gelöst, wobei Krylov-Räume $\mathcal{K}(A(\varepsilon), b(\varepsilon))$ generiert werden. Uns interessiert der Grenzwert der Krylov-Räume für $\varepsilon \to 0$.

Satz 5.4.1. *Möge der Krylovraum $\mathcal{K}_{n_z}(g_z, g)$ nicht degenerieren, d.h., die maximale Dimension n_z (Dimension der algebraischen Variablen z) besitzen. Dann erhält man bei Krylov-Iterationen für* (5.30) *im Grenzfall $\varepsilon \to 0$ eine Folge von Räumen*

$$\lim_{\varepsilon \to 0} \mathcal{K}_n(A(\varepsilon), b(\varepsilon)) = \mathrm{span}\{u_1, \ldots, u_n\} \text{ für } n = 1, \ldots, \tag{5.31}$$

die durch eine Folge von Vektoren u_n aufgespannt werden mit

$$u_n = \begin{pmatrix} 0 \\ g_z^{n-1} g \end{pmatrix} \qquad \text{für } n \le n_z, \tag{5.32}$$

$$u_n = \begin{pmatrix} f + \underline{\mathrm{k}} - f_z g_z^{-1} g \\ 0 \end{pmatrix} \qquad \text{für } n = n_z + 1, \tag{5.33}$$

$$u_n = \begin{pmatrix} I - h\gamma(f_y - f_z g_z^{-1} g_y) & 0 \\ 0 & 0 \end{pmatrix} u_{n-1} \quad \text{für } n > n_z + 1 \tag{5.34}$$

Beweis. Eine Multiplikation des gesamten Gleichungssystems (5.30) mit ε läßt die Krylovräume unverändert. Wir betrachten den Krylovraum $\mathcal{K}(A, b)$ mit $A = A_0 + \varepsilon A_1$ und $b = b_0 + \varepsilon b_1$, wobei

$$A_0 = -h\gamma \begin{pmatrix} 0 & 0 \\ g_y & g_z \end{pmatrix} \qquad A_1 = I - h\gamma \begin{pmatrix} f_y & f_z \\ 0 & 0 \end{pmatrix} \tag{5.35}$$

$$b_0 = \begin{pmatrix} 0 \\ g \end{pmatrix} \qquad b_1 = \begin{pmatrix} f + \underline{\mathrm{k}} \\ \underline{\mathrm{l}} \end{pmatrix}. \tag{5.36}$$

Für die generierenden Vektoren des Krylovraumes $u_n = A^{n-1} b$, $n = 1, \ldots$, erhalten wir die für $n > 1$ gültige ε-Entwicklung

$$u_n = \begin{pmatrix} \varepsilon(-h\gamma)^{n-1} f_z g_z^{n-2} g \\ (-h\gamma)^{n-1} g_z^{n-1} g \end{pmatrix} + \begin{pmatrix} \mathcal{O}(\varepsilon^2) \\ \mathcal{O}(\varepsilon) \end{pmatrix}. \tag{5.37}$$

Um dies auch auf $n = 1$ auszudehnen, schreiben wir

$$u_n = \begin{pmatrix} \varepsilon(-h\gamma)^{n-1} f_z g_z^{n-2} g \\ (-h\gamma)^{n-1} g_z^{n-1} g \end{pmatrix} + \delta_{n1}\varepsilon \begin{pmatrix} f + \underline{\mathrm{k}} - f_z g_z^{-1} g \\ 0 \end{pmatrix} + \begin{pmatrix} \mathcal{O}(\varepsilon^2) \\ \mathcal{O}(\varepsilon) \end{pmatrix}. \tag{5.38}$$

Fassen wir die Vektoren $u_1, \ldots, u_n$ als Spalten der Matrix U_n auf, so ergibt sich

$$U_n = \begin{pmatrix} \varepsilon V_n \\ W_n \end{pmatrix} + \begin{pmatrix} \mathcal{O}\left(\varepsilon^2\right) \\ \mathcal{O}\left(\varepsilon\right) \end{pmatrix}, \tag{5.39}$$

wobei die Beziehung

$$V_n = f_z g_z^{-1} W_n + \begin{pmatrix} f + \underline{\mathrm{k}} - f_z g_z^{-1} g \\ 0 \end{pmatrix} e_1^T + \begin{pmatrix} \mathcal{O}\left(\varepsilon^2\right) \\ \mathcal{O}\left(\varepsilon\right) \end{pmatrix} \tag{5.40}$$

gilt.

Beim Aufbau des Krylovraumes mit dem Arnoldi-Prozeß wird in jedem Schritt normalisiert und orthogonalisiert. Wir ignorieren dies für die Vektoren u_k mit $k \leq n_z$ vollständig, und fassen diese als Spalten zu der Matrix U_{n_z} zusammen. Aufgrund der Voraussetzung an g_z besitzt U_{n_z} Vollrang, das Bild von U_{n_z} ist für $\varepsilon \to 0$ gerade der Raum der algebraischen Variablen. Wie sehen nun die weiteren Iterierten aus?

Für Vektoren u_n mit $n > n_z$ orthogonalisieren wir gegen die Spalten von U_{n_z}, allerdings nur soweit, daß

$$u_n^T q_k = \mathcal{O}\left(\varepsilon \|u_n\| \|q_k\|\right) \tag{5.41}$$

gilt. Dann normieren wir auf $\|q_n\| = \mathcal{O}\left(1\right)$, somit gilt dann $u_n^T q_k = \mathcal{O}\left(\varepsilon\right)$.

Betrachten wir zunächst allgemein eine (partielle) Orthogonalisierung eines Vektors u gegen U_{n_z}. Da im Arnoldi-Prozeß $u = Au'$, können wir u in der Form

$$u = \begin{pmatrix} \varepsilon v \\ w \end{pmatrix} + \begin{pmatrix} \mathcal{O}\left(\varepsilon^2\right) \\ \mathcal{O}\left(\varepsilon\right) \end{pmatrix}. \tag{5.42}$$

ansetzen. Eine partielle Orthogonalisierung gegen U_{n_z} wird in 2 Schritten realisiert. Zunächst werden die $\mathcal{O}\left(1\right)$-Terme in der z-Komponente eliminiert:

$$\begin{aligned} u \mapsto & u - \begin{pmatrix} \varepsilon V_{n_z} + \mathcal{O}\left(\varepsilon^2\right) \\ W_{n_z} + \mathcal{O}\left(\varepsilon\right) \end{pmatrix} W_{n_z}^{-1} w && (5.43) \\ = & \begin{pmatrix} \varepsilon(v - V_{n_z} W_{n_z}^{-1} w) + \mathcal{O}\left(\varepsilon^2\right) \\ \mathcal{O}\left(\varepsilon\right) \end{pmatrix} && (5.44) \\ = & \begin{pmatrix} \varepsilon(v - f_z g_z^{-1} w + (f + \underline{\mathrm{k}} - f_z g_z^{-1} g) e_1^T W_{n_z}^{-1} w) + \mathcal{O}\left(\varepsilon^2\right) \\ \mathcal{O}\left(\varepsilon\right) \end{pmatrix} && (5.45) \end{aligned}$$

In einem weiteren Schritt eliminieren wir die $\mathcal{O}\left(\varepsilon\right)$-Einträge in der z-Komponente:

$$\begin{aligned} u \mapsto & u - \begin{pmatrix} \varepsilon V_{n_z} + \mathcal{O}\left(\varepsilon^2\right) \\ W_{n_z} + \mathcal{O}\left(\varepsilon\right) \end{pmatrix} W_{n_z}^{-1} \mathcal{O}\left(\varepsilon\right) && (5.46) \\ = & \begin{pmatrix} \varepsilon(v - f_z g_z^{-1} w + (f + \underline{\mathrm{k}} - f_z g_z^{-1} g) e_1^T W_{n_z}^{-1} w) + \mathcal{O}\left(\varepsilon^2\right) \\ \mathcal{O}\left(\varepsilon^2\right) \end{pmatrix} && (5.47) \end{aligned}$$

Damit gilt zunächst die Bedingung (5.41). Eine Normierung auf $\mathcal{O}\left(1\right)$ liefert dann q

$$\begin{pmatrix} \varepsilon v \\ w \end{pmatrix} + \begin{pmatrix} \mathcal{O}\left(\varepsilon^2\right) \\ \mathcal{O}\left(\varepsilon\right) \end{pmatrix} \mapsto q = \begin{pmatrix} v - f_z g_z^{-1} w + (f + \underline{\mathrm{k}} - f_z g_z^{-1} g) e_1^T W_{n_z}^{-1} w \\ 0 \end{pmatrix} + \mathcal{O}\left(\varepsilon\right). \tag{5.48}$$

Betrachten wir also jetzt die Anwendung der Orthogonalisierung nach (5.48) von u_{n_z+1}, welches durch (5.38) gegeben ist. Der entsprechende Term $v - f_z g_z^{-1} w$ verschwindet, so daß sich nach einer Normierung

$$u_{n_z+1} \mapsto q_{n_z+1} = \begin{pmatrix} f + \underline{\mathrm{k}} - f_z g_z^{-1} g \\ 0 \end{pmatrix} + \mathcal{O}(\varepsilon) \tag{5.49}$$

ergibt. Dabei müssen wir voraussetzen, daß $e_1^T W_{n_z}^{-1} w_{n_z+1} \neq 0$. Dies ist in der Tat erfüllt, da wegen der Regularität von g_z der konstante Term im charakteristischen Polynom von g_z von Null verschieden ist.

Alle weiteren Vektoren besitzen nach partieller Orthogonalisierung (5.48) und Normierung ebenso wie u_{n_z+1} die Gestalt

$$u = \begin{pmatrix} v \\ 0 \end{pmatrix} + \mathcal{O}(\varepsilon). \tag{5.50}$$

Bei Durchführung eines Krylov-Schritts wird u mit A multipliziert und nach (5.48) projiziert:

$$Au = \begin{pmatrix} \varepsilon(I - f_y)v \\ g_y v \end{pmatrix} + \begin{pmatrix} \mathcal{O}(\varepsilon^2) \\ \mathcal{O}(\varepsilon) \end{pmatrix} \to q \tag{5.51}$$

$$q = \begin{pmatrix} (I - (f_y - f_z g_z^{-1} g_y))v \\ 0 \end{pmatrix} + e_1^T W_{n_z}^{-1} g_y v \begin{pmatrix} (f + \underline{\mathrm{k}} - f_z g_z^{-1} g) \\ 0 \end{pmatrix} + \mathcal{O}(\varepsilon). \tag{5.52}$$

Der zweite Summand verschwindet bei einer Orthogonalisierung mit u_{n_z+1}, womit der letzte Teil des Beweises erbracht ist. □

Dieser Zugang führt also auf den Vorschlag **KDGL** aus dem vorhergehenden Abschnitt, wo die direkte iterative Lösung der Gleichungen (5.28) diskutiert wurde: Führen wir $n_z + \kappa$ Iterationen durch, so werden die Nebenbedingungen exakt gelöst und für die differentiellen Komponenten werden κ Iterationen ausgeführt.

Da beide Wege im Diagramm (5.27) in engem Zusammenhang stehen, folgern wir, daß bei Anwendung von Krylov-Techniken auf die Gleichung (5.28) nach Variante **KDAE** eine Begrenzung der Maximalzahl von Iterationen auf Werte in der Größenordnung 10 bis 30, wie bei gewöhnlichen Differentialgleichungen mit Erfolg praktiziert, hier nicht zu empfehlen ist. Bei einer geringen Zahl algebraischer Bedingungen erwachsen aus Satz 5.4.1 keine wesentlichen Einschränkungen. Ist die Zahl der algebraischen Bedingungen jedoch groß, kann möglicherweise eine hohe Anzahl von Krylov-Schritten erforderlich sein.

Wir werden Relevanz der erhaltenen theoretischen Ergebnisse im nächsten Abschnitt auf den Prüfstand stellen. Ein reales Anwendungsproblem – viskoelastische Defomation einer Scheibe – führt auf große Systeme mit einer hohen Anzahl von algebraischen Variablen.

5.5 Anwendungen in der Viskoelastizität

5.5.1 Grundlagen der linearen Elastizität deformierbarer Festkörper

Das mathematische Modell

Ein deformierbarer Festkörper wird mathematisch im Grundzustand (undeformiert) als Teilmenge Ω des $\mathbb{R}^3$ modelliert. Eine Deformation wird durch die Verschiebung $u : \Omega \to \mathbb{R}^3$ beschrieben – ein Punkt $x \in \Omega$ wird durch die Deformation in den Punkt $x + u(x)$ überführt. Dies führt im allgemeinen zu lokalen Verzerrungs- und Spannungszuständen. Diese werden durch tensorielle Felder beschrieben - den Verzerrungstensor $\boldsymbol{\varepsilon}$ und den Spannungstensor $\boldsymbol{\sigma}$. Die Verzerrung $\boldsymbol{\varepsilon}$ hängt von der Jacobi-Matrix ∇u ab – im Falle kleiner Verzerrungen betrachtet man nur die linearen Anteile

$$\boldsymbol{\varepsilon} = Du := \frac{1}{2}\left(\nabla u + \nabla u^T\right) \tag{5.53}$$

Die 3 Komponenten der Verschiebungen u sind die eigentlichen Unbekannten des Systems. Aus diesen ergibt sich mit Gleichung (5.53) der lineare Verzerrungstensor $\boldsymbol{\varepsilon}$. Über das konstitutive Gesetz ergibt sich aus $\boldsymbol{\varepsilon}$ dann der Spannungstensor $\boldsymbol{\sigma}$, welcher wiederum 3 Gleichungen (Kräftegleichgewicht) genügt.

Die konstitutiven Gleichungen

Im einfachen Fall einer eindimensionalen homogenen Verzerrung - einer Feder - entspricht $\boldsymbol{\varepsilon}$ der relativen Auslenkung der Feder und $\boldsymbol{\sigma}$ der entsprechenden Kraft. Das klassische Grundgesetz - Kraft $\sim$ Auslenkung - findet seine Entsprechung im Hookeschen Materialgesetz [6], klassisch mit den Lamé-schen Konstanten μ, λ formuliert:

$$\boldsymbol{\sigma} = C\boldsymbol{\varepsilon} := 2\mu\boldsymbol{\varepsilon} + \lambda(\operatorname{spur}\boldsymbol{\varepsilon})I. \tag{5.54}$$

Bei einer Interpretation als Feder ist eine Darstellung mit dem Elastizitätsmodul E und der Querkontraktionszahl ν nützlich:

$$\boldsymbol{\varepsilon} = \frac{1+\nu}{E}\boldsymbol{\sigma} - \frac{\nu}{E}(\operatorname{spur}\boldsymbol{\sigma})I. \tag{5.55}$$

Der Elastizitätsmodul entspricht der Federkonstanten, die Zahl ν gibt die relative Querkontraktion bei einachsiger Belastung an.

Für viskoelastische Effekte ist insbesondere der kompressionsfreie Anteil der Deformation von Bedeutung. Dieser ergibt sich aus dem Deviator

$$\boldsymbol{\varepsilon}^D := \boldsymbol{\varepsilon} - \frac{1}{3}(\operatorname{spur}\boldsymbol{\varepsilon})I. \tag{5.56}$$

Mit dem Schubmodul $G = \mu$ und der Kompressionszahl $K = \lambda + 2\mu$ ergibt sich

$$\boldsymbol{\sigma} = 2G\boldsymbol{\varepsilon}^D + K(\operatorname{spur}\boldsymbol{\varepsilon})I, \tag{5.57}$$

Die Gleichungen (5.57) separieren mit diesem Ansatz einfach zu $\boldsymbol{\sigma}^D = 2G\boldsymbol{\varepsilon}^D$ und $\operatorname{spur}\boldsymbol{\sigma} = 3K \operatorname{spur}\boldsymbol{\varepsilon}$.

Gleichgewichtsbedingungen

Vorab einige Worte zur physikalischen Bedeutung von $\boldsymbol{\sigma}$. Unter der Annahme, daß sich das System in Ruhe befindet, hängt die auf ein infinitesimales Flächenstück wirkende Kraft linear von der Flächennormalen ab – diese lineare Abbildung wird durch $\boldsymbol{\sigma}$ beschrieben:

$$F = \boldsymbol{\sigma}\mathbf{n}. \tag{5.58}$$

Die Annahme statischen Gleichgewichts unter der Wirkung einer Volumenkraftdichte f (zumeist ausschließlich die Gravitation) liefert 3 Gleichungen für $\boldsymbol{\sigma}$:

$$\operatorname{div} \boldsymbol{\sigma} + f = 0. \tag{5.59}$$

Bemerkung 5.5.1. *Genaugenommen folgen 3 weitere Gleichungen – nämlich die Symmetrie des Spannungstensors – aus dem Momentengleichgewicht. Wir haben dies durch das spezielle konstitutive Gesetz bereits berücksichtigt.*

Starke Formulierung mit Randbedingungen

Die Gleichungen (5.53), (5.54), (5.59) ergeben nach Elimination von $\boldsymbol{\varepsilon}, \boldsymbol{\sigma}$ eine partielle Differentialgleichung zweiter Ordnung für u.

Als Randbedingungen sind im einfachsten Fall Dirichlet-Bedingungen auf Γ_D (Verschiebung u bekannt) und Neumann-Bedingungen auf Γ_N (Oberflächenkräfte g bekannt) gegeben. Wir setzen hier $\Gamma_D \cup \Gamma_N = \partial\Omega$, $\Gamma_D \cap \Gamma_N = \emptyset$ voraus. Das Problem der linearen Elastizität lautet somit

$$\operatorname{div} CDu + f = 0 \tag{5.60a}$$

$$u(x) = u_D(x) \text{ für } x \in \Gamma_D \tag{5.60b}$$

$$(CDu) \cdot \mathbf{n}(x) = g(x) \text{ für } x \in \Gamma_N. \tag{5.60c}$$

Natürlich fehlen hier die Räume, aus denen $u, \boldsymbol{\varepsilon}, \boldsymbol{\sigma}, f, g$ zu wählen sind. Fordert man starke Lösungen, so ist dies generell ein sehr schwieriges Problem.

Bemerkung 5.5.2. *Bereits die Poisson-Gleichung $-\Delta u = f$ unter homogenen Dirichlet-Bedingungen besitzt nicht für alle $f \in C(\Omega)$ eine Lösung $u \in C^2(\Omega)$, wohl aber existieren schwache Lösungen $u \in W^{2,p}$. Um Lösungen in C^2 zu erhalten, kann man $f \in C^\alpha(\Omega)$ fordern, dann erhält man sogar $u \in C^{2+\alpha}(\Omega)$.*

Das Problem (5.60) ist ähnlich gelagert, in geeigneter Formulierung entpuppt es sich als elliptisches Problem. Wir greifen daher bei der Formulierung auf schwache Lösungen zurück.

Variationsformulierungen

Die Gleichgewichtsbedingung $\nabla \cdot \boldsymbol{\sigma} + f = 0$ folgt für hinreichend glatte Lösungen aus einem Variationsproblem. Minimiert wird die Gesamtenergie, die sich aus der Deformationsenergie

$$\Phi = \frac{1}{2}\boldsymbol{\varepsilon} : \boldsymbol{\sigma} = \frac{1}{2}\boldsymbol{\varepsilon} : C\boldsymbol{\varepsilon} \tag{5.61}$$

und der negativen mechanischen Arbeit der Volumen- und Flächenkräfte zusammensetzt. Die Neumann-Bedingungen sind an das Funktional gekoppelt, während die Dirichlet-Bedingungen als Eigenschaft des Lösungsraums

$$H^1_{\Gamma_D}(u_D) := \{u \in H^1(\Omega) : u(x) = u_D(x) \text{ für } x \in \Gamma_D\} \tag{5.62}$$

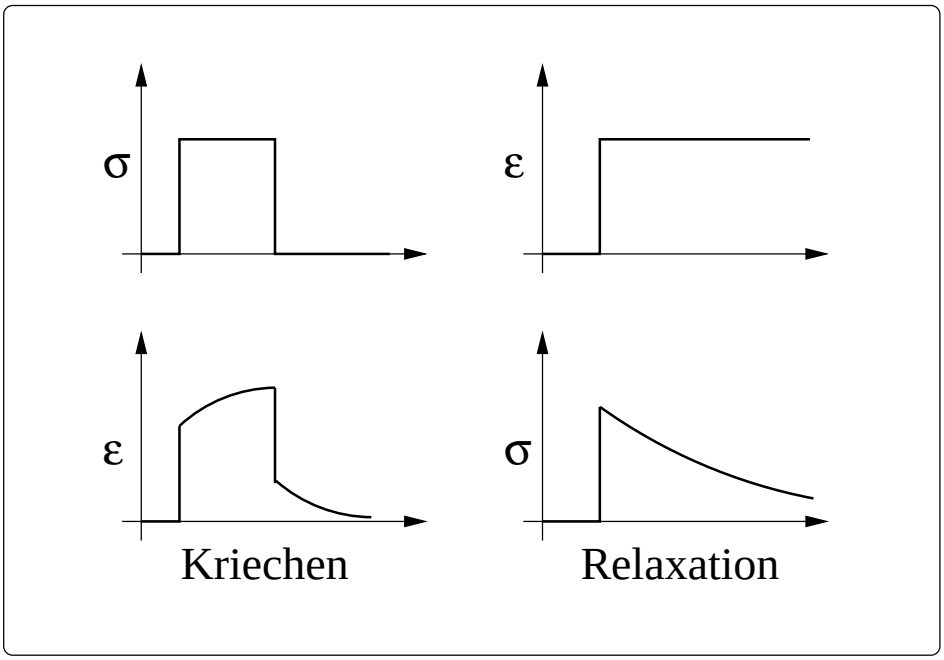


Abbildung 5.1: Typische Phänomene viskoelastischer Medien

formuliert werden. Wir erhalten das zu (5.60) äquivalente Variationsproblem

$$\int_\Omega \Phi - fudx - \int_{\partial\Omega} u \cdot gdA \rightarrow \min_{u \in H^1_{\Gamma_D}(u_D)}. \tag{5.63}$$

Damit ergibt sich $u \in H^1_{\Gamma_D}(u_D)$ als Lösung der schwachen Formulierung

$$\int_\Omega Du : CDv - fvdx - \int_{\Gamma_N} gvdA = 0 \quad \forall v \in H^1_{\Gamma_D}(0). \tag{5.64}$$

Die Kornsche Ungleichung [6] besagt, daß die Bilinearform

$$a(u,v) := \int_\Omega Du : CDvdx \tag{5.65}$$

elliptisch ist, sofern $|\Gamma_D| > 0$.

5.5.2 Viskoelastizität

Phänomenologische Beschreibung

Im Falle reiner Elastizität ist eine eindeutige Beziehung zwischen Spannungen und Verzerrungen vorhanden. Plastizität und Viskoelastizität sind dadurch gekennzeichnet, daß die Spannungs-Verzerrungs-Relation nicht unabhängig von den vorhergehenden Deformationen des Materials ist.

In der Praxis verknüpft man Viskoelastizität mit mehreren Phänomenen, die über ein mit reiner Elastizität beschreibbares Verhalten hinausgehen. Zu den wichtigsten zählen Kriechen und Relaxation, siehe Abbildung 5.1.

Kriechen beschreibt ein Phänomen bei festgelegter Spannung. Bei einer sehr schnellen Spannungsänderung stellt sich zunächst unmittelbar der elastische Anteil der Verzerrung ein. Die tatsächliche Verzerrung nähert sich in Form eines Kriechprozesses dem endgültigen Verzerrungszustand. Wird die Spannung wieder auf Null gesetzt, verschwindet auch unmittelbar der elastische Anteil der Verzerrung. Der inelastische Anteil klingt langsam ab.

Bei der Relaxation hingegen ist der Verzerrungszustand festgelegt. Der endgültige Verzerrungszustand wird durch eine relativ schnelle Deformation hergestellt und fixiert. Dies führt zu überhöhten Spannungen, die erst im folgenden zeitlichen Verlauf wieder abgebaut werden, bis das elastische Niveau erreicht ist. Dieses Phänomen wird Relaxation genannt.

Atome und 1D-Modelle der Viskoelastizität

In der Literatur sind verschiedene Modelle zur Reproduktion typischer Phänomene der Viskoelastizität beschrieben, man siehe zum Beispiel [49]. Während sich lineare Elastizität im eindimensionalen Fall auf die klassische Spannungs-Dehnungs-Relation für die elastische Feder zurückführen läßt, verwendet man zur Modellierung viskoelastischen Verhaltens Dämpfungsglieder, deren Spannung proportional zur Änderungsgeschwindigkeit der relativen Auslenkung ist.

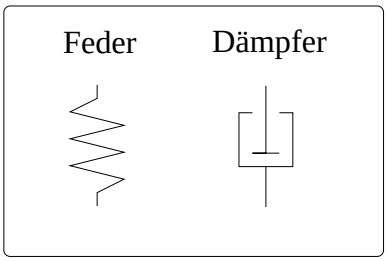


Abbildung 5.2: Die Atome der Viskoelastizität

Man ordnet diese Dämpfungsglieder parallel oder in Reihe mit Federn an, und gelangt so zu verschiedenen Modellen. Das Maxwell-Modell besteht aus einer Feder in Reihe mit einem Dämpfer, das Kelvin-Voigt-Modell hingegen schaltet beide parallel. Letzteres ist auch unter dem Namen Kelvin-Körper bekannt. Diese einfachen Modelle sind nicht zur qualitativ korrekten Modellierung viskoelastischen Verhaltens geeignet. Das Maxwell-Modell gibt zwar qualitativ korrektes Relaxationsverhalten wieder, liefert aber ein lineares Kriechenverhalten. Das Kelvin-Voigt-Modell wiederum liefert qualitativ korrektes Kriechen, ist aber nicht in der Lage, Relaxationsverhalten wiederzugeben.

Die Spannungs-Dehnungs-Relation läßt sich für die Modelle wie folgt herleiten. Für die Federn gelten die Relationen (Hookesches Gesetz)

$$\boldsymbol{\sigma} = E\boldsymbol{\varepsilon}, \tag{5.66}$$

für die Dämpfer hingegen (Newtonsche Flüssigkeit)

$$\boldsymbol{\sigma} = \eta\dot{\boldsymbol{\varepsilon}}. \tag{5.67}$$

Offensichtlich addieren sich die Verzerrungen bei Anordnung in Reihe, bei paralleler Anordnung sind sie dagegen gleich. Bei den Spannungen verhält es sich genau umgekehrt: Bei paralleler Anordnung addieren sich die Spannungen (mehrere Kräfte greifen gewissermaßen an einem Punkt an), bei Anordnung in Reihe sind die Spannungen gleich (Kraft=Gegenkraft).

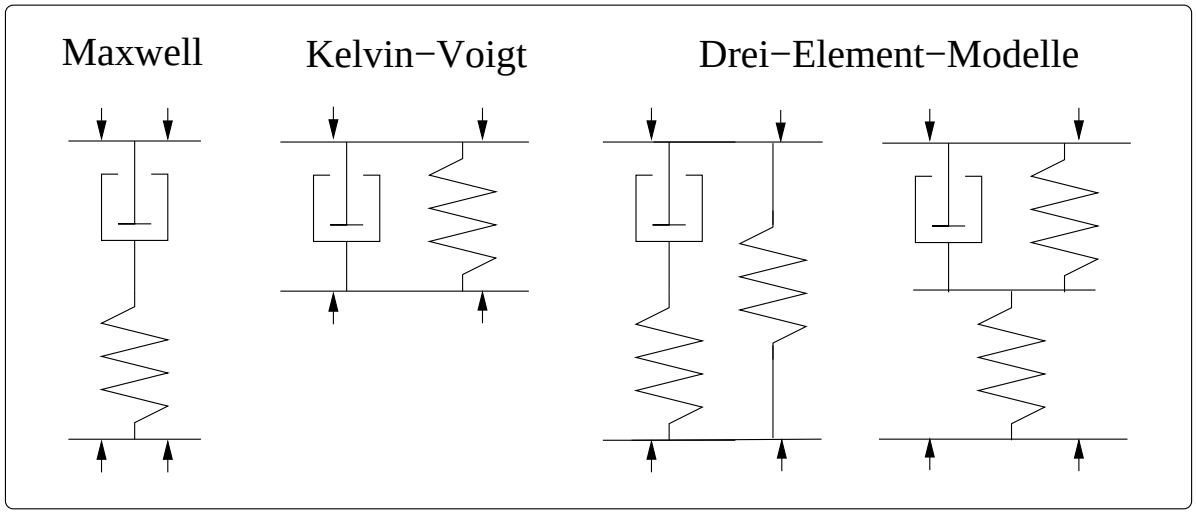


Abbildung 5.3: Klassische Modelle der Viskoelastizität

Die Eigenschaften beider Modelle werden in verschiedenen 3-Element-Modellen (siehe Abbildung 5.3 rechts) vereinigt, wo Federn sowohl in Reihe als auch parallel zum Dämpfer auftreten. Wir wollen uns in den weiteren Betrachtungen auf die Parallelschaltung eines Maxwell-Elements mit einer Feder beschränken (linkes Drei-Element-Modell).

Wir bezeichnen die Auslenkungen der Feder bzw. des Dämpfers des Maxwell-Gliedes mit ε_M bzw. q, den zugehörigen Elastizitätsmodul bzw. die Viskosität mit E_M bzw. η und die Spannung am Maxwell-Glied mit $\boldsymbol{\sigma}_M$. An der parallelgeschalteten Einzelfeder sei die Spannung mit $\boldsymbol{\sigma}_0$, der Elastizitätsmodul mit E_0 und die Verzerrung mit $\varepsilon_0 = \varepsilon$ (=Gesamtverzerrung) bezeichnet. Die Gesamtspannung $\boldsymbol{\sigma} = \boldsymbol{\sigma}_0 + \boldsymbol{\sigma}_M$ ergibt sich dann in Abhängigkeit von der Gesamtverzerrung ε und der internen Variablen (anelastische Verzerrung) q, während die internen Variablen einer gewöhnlichen Differentialgleichung genügen.

$$\boldsymbol{\sigma} = E_0\varepsilon + E_M(\varepsilon - q) = E\varepsilon - E_M q \tag{5.68}$$

$$q' = \frac{E_M}{\eta}(\varepsilon - q). \tag{5.69}$$

Dabei entspricht E_0 der Spannungs-Dehnungs-Relation im Grenzfall $t \to \infty$ (Deformationsgeschwindigkeit 0, dann gilt $q = \varepsilon$, da $q' = 0$), und $E = E_0 + E_M$ der Spannungs-Dehnungs-Relation für die unmittelbare elastische Reaktion des Systems bei schnellen Deformationen, da dann $q = 0$ gilt.

Im eindimensionalen Fall ergibt sich $\boldsymbol{\sigma}$ aus den äußeren Kräften, so daß eine DAE vom Index 1 vorliegt.

Beim Kriechen ergibt sich mit Anlegen einer Spannung $\boldsymbol{\sigma}$ zunächst die elastische Verzerrung $\varepsilon = \boldsymbol{\sigma}/E$, im Grenzfall $t \to \infty$ die Verzerrung $\varepsilon = \boldsymbol{\sigma}/E_0$. Der typische zeitliche Horizont für eine viskoelastische Verformung ist $t = \mathcal{O}\left(\frac{\eta}{E_M}\right)$, d.h., in einer solchen Zeitspanne hat sich die Änderung der internen Variablen im wesentlichen vollzogen.

Auch Relaxation wird vom Modell befriedigend wiedergegeben, bei Vorgabe von ε fällt $\boldsymbol{\sigma}$ von $\boldsymbol{\sigma} = E\varepsilon$ exponentiell auf $\boldsymbol{\sigma} = E_0\varepsilon$. Der zeitliche Horizont ändert sich dabei im Vergleich zum Kriechen nicht.

Das lineare Standardmodell in 3D

Zur Modellierung viskoelastischen Materialverhaltens in drei Dimensionen verwenden wir die Formulierung des Materialgesetzes mit Schub- und Kompressionsmodul. Wie die Erfahrung zeigt, kann viskoelastisches Verhalten im allgemeinen nur für den volumenerhaltenden (deviatorischen) Anteil der Verzerrung beobachtet werden. Wir verwenden also das lineare Modell (5.68) für ε^D, $\boldsymbol{\sigma}^D$, und setzen für die reine Volumendeformation $\operatorname{spur}\varepsilon$ linear-elastisches Verhalten voraus. Die konstitutiven Gesetze für unser Modell, in der Formulierung mit Schubmodul G und Kompressionsmodul K, lauten

$$\boldsymbol{\sigma} = 2G\varepsilon^D - 2G_M q^D + K(\operatorname{spur}\varepsilon)I \tag{5.70}$$

$$\eta {q'}^D = 2G_M(\varepsilon^D - q^D) \tag{5.71}$$

Mit den Gleichungen (5.60a) und (5.53) lassen sich aus der Gleichung (5.70) die Variablen $\boldsymbol{\sigma}$, ε in Abhängigkeit von q bestimmen. Es liegt somit ein differential-algebraisches System vom Index 1 vor.

2D-Formulierung als ebener Verzerrungszustand

Unter der Annahme von Symmetrien in z-Richtung gelangen wir zum Scheiben-Modell. Für die elastischen Anteile gibt es zwei prinzipielle Ansätze – den ebenen Spannungszustand und den ebenen Verzerrungszustand. Wir verwenden den letzteren.

Die z-Komponente der Verrückung möge verschwinden, und die x, y-Komponenten mögen von der z-Koordinate unabhängig sein. Für den Verzerrungstensor gilt demzufolge $\varepsilon_{i3} = \varepsilon_{3i} = 0$ für $i = 1, 2, 3$, sowie $\varepsilon_{ij}(x, y, z) = \varepsilon_{ij}(x, y)$. Aus der Materialgleichung ist ersichtlich, daß der Spurterm Spannungen in z-Richtung hervorruft. Realer Hintergrund ist ein zwischen starren Platten eingespanntes elastisches/ viskoelastisches Material.

Im elastischen Fall geschieht die Übertragung am einfachsten in der Laméschen Formulierung $\boldsymbol{\sigma} = 2G\varepsilon + (K - 2/3G)\operatorname{spur}\varepsilon I$, diese kann unverändert auf den oberen 2×2-Block eingeschränkt werden. Formal sorgt der Spurterm zusätzlich für $\boldsymbol{\sigma}_{33} \neq 0$. Diese Komponente entspricht Spannungen in z-Richtung, für die wir uns hier nicht interessieren. Wir betrachten daher einfach nur die oberen 2×2-Blöcke von ε und $\boldsymbol{\sigma}$.

Im viskoelastischen Fall ist die Übertragung ähnlich, allerdings liegen unsere Gleichungen in der Form mit Schub- und Kompressionsmodul vor. Zu beachten ist ferner, daß Deviator und die Spur eine 2×2-Matrix A nicht mehr in zwei unabhängige Anteile zerlegen, vielmehr bestimmt der 3D-Deviator $A^D = A - \operatorname{spur} A/3I$ bereits die Matrix A durch

$$A = A^D + (\operatorname{spur} A^D)I. \tag{5.72}$$

Die Hauptdiagonalelemente des Deviators sind linear unabhängig, wir schreiben daher einfach q anstelle von q^D. Auch eliminieren wir den Deviator im Materialgesetz (5.70) und gelangen mit $\lambda = K - 2/3G$ quasi wieder zur Laméschen Formulierung

$$\boldsymbol{\sigma} = 2G\varepsilon - 2G_M q^D + \lambda(\operatorname{spur}\varepsilon)I \tag{5.73}$$

$$\eta q' = 2G_M(\varepsilon^D - q) \tag{5.74}$$

Schwache Formulierung

Die Bewegung des Systems wird bei Kriechprozessen als sehr langsam angesehen, so daß wir die auftretenden Trägheitskräfte vernachlässigen können. Die Evolutionsgleichungen für die viskosen Verzerrungen behalten wir als Gleichungen bei, während wir für das Kräfte- und Momentengleichgewicht eine schwache Formulierung verwenden.

In der schwachen Formulierung können nun Verschiebungen, Verzerrungen und auch die Spannungen auftreten, man spricht in diesem Fall von gemischten Methoden, siehe [6]. Wir verwenden den reinen Verschiebungsansatz, wo sowohl die Spannungen $\boldsymbol{\sigma}$ als auch die Verzerrungen $\boldsymbol{\varepsilon}$ eliminiert sind.

Das System besteht jetzt aus Evolutionsgleichungen und der schwachen Formulierung zur Bestimmung der Verschiebungen u:

$$q' = \frac{2G_M}{\eta}((Du)^D - q) \tag{5.75}$$

$$\begin{aligned} 0 = &\int_\Omega (2GDv : Du + \lambda \operatorname{div} u \operatorname{div} v)\, dx \\ &- \int_\Omega (2G_M Dv : q + v \cdot f)\ dx - \int_{\Gamma_N} g \cdot v\ ds \end{aligned} \tag{5.76}$$

Die Variationsgleichung (5.76) und auch ihre Diskretisierung haben (bei geeigneter Wahl der Lösungsräume) eine eindeutig bestimmte Lösung. Somit liegt ein differential-algebraisches System vom Index 1 vor. Probleme der Viskoelastizität wurden unter diesem Gesichtspunkt von verschiedenen Autoren modelliert und gelöst, wir verweisen auf [39], [29], [30], [48] und [49].

Bemerkung 5.5.3. *Der Fall der Plastizität – also das Auftreten von Hystereseeffekten – führt auf Systeme vom Index 2, man vergleiche hierzu die Arbeiten von Simeon, Scherf und Büttner [89], [90] und [9].*

5.5.3 Ein Scherexperiment

Konfiguration und Materialparameter

Wir betrachten eine Scheibe im ebenen Verzerrungszustand. Als Längen geben wir (dimensionslos) $l_x = 1$ und $l_y = 0.1$ vor. An der linken Seite sei die Scheibe fest eingespannt, an der rechten Seite wirke eine vertikale Scherkraft.

Wir haben die für Stahl typischen Werte $G = 200000$ und $\nu = 0.3$ vorgegeben. Für das Maxwell-Glied geben wir $G_M = G/2$ und $\eta = 1000$ vor. Wir lösen das Problem für konstante Scherkraft im Zeitintervall $[0, 100]$.

Ortsdiskretisierung

Die schwache Formulierung (5.76) legt eine Diskretisierung mit finiten Elementen nahe. Bei der Berechnung der Steifigkeitsmatrix für die finiten Elemente verwendet man häufig Gauß-Quadratur-Formeln, obwohl die auftretenden Skalarprodukte auch exakt aus den Ansatzfunktionen berechenbar sind. Dies vereinfacht die Implementierung, und ist oft auch weniger aufwendig. Hier legt dieser Ansatz nahe, die internen Variablen q gerade in den Knotenpunkten der Gauß-Quadratur-Formeln zu definieren. Wir haben uns dafür entscheiden, nur den Mittelpunkt der Elemente zur Bestimmung der internen Variablen q heranzuziehen.

Für hinreichend glatte äußere Kräfte kann zur schwachen Formulierung mit der reinen Verschiebungsmethode der Raum $H^1(\Omega)$ genutzt werden. Wir verwenden zur Diskretisierung vierknotige Viereckselemente mit bilinearen Ansatzfunktionen [55]. Im Falle reiner Elastizität ergibt sich das Resultat des Scherversuchs mit 10×4 Elementen wie in Abbildung 5.4 dargestellt.

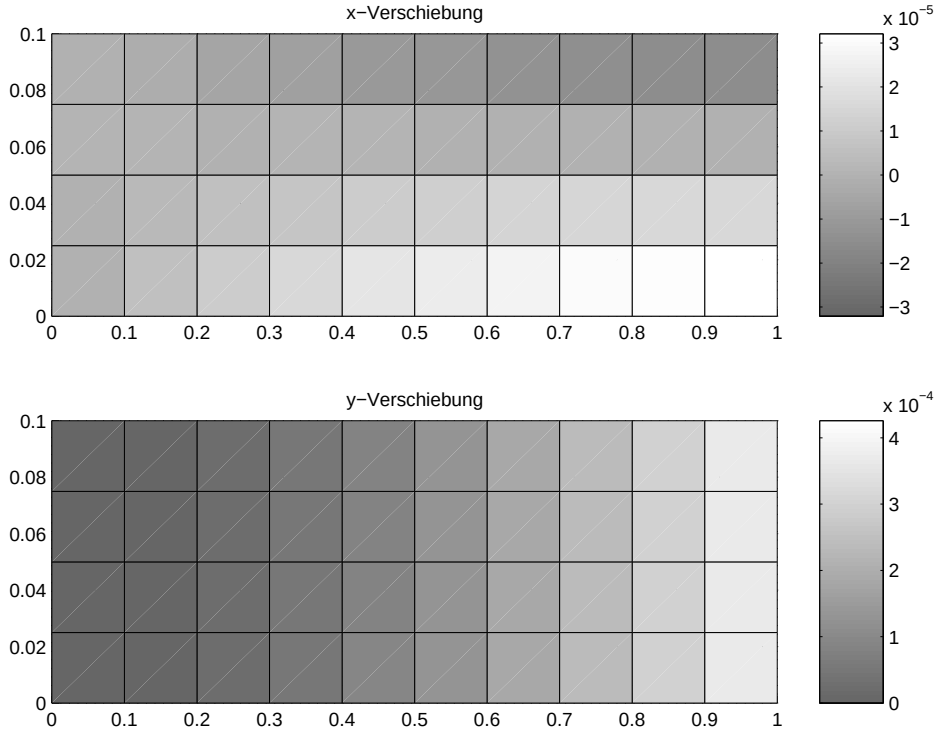


Abbildung 5.4: Verschiebung in x- und y-Richtung beim Scherversuch.

Zeitintegration

Wendet man klassische Rosenbrock-Methoden auf DAEs vom Index 1 an, so sind zusätzliche Ordnungsbedingungen zu erfüllen. Wir haben uns für die Methode RODAS von Hairer und Wanner [46] entschieden. Sie erfüllt die zusätzlichen Ordnungsbedingungen für Index-1-Systeme. Mit $s = 5$ Stufen erhält man eine Lösung der Ordnung 4 sowie eine eingebettete Lösung der Ordnung 3.

Wir haben die Rosenbrock-Methode RODAS mit direkter Lösung der linearen Gleichungssysteme mit der entsprechenden Krylovraum-Methode nach Variante **KDAE** (5.4.2, Seite 113) verglichen. Zur iterativen Lösung der Gleichungssysteme haben wir GMRES herangezogen. Das lineare Gleichungssystem wird bis zu einer vorgegebenen Genauigkeit gelöst, wir haben diese Genauigkeit TOL mit der relativen Toleranz $RTOL$ für die Schrittweitensteuerung gekoppelt. Dabei haben wir folgende 3 Varianten betrachtet:

$$TOL = 0.1RTOL, \quad TOL = h * RTOL, \quad TOL = 0.01 * RTOL/h. \tag{5.77}$$

Die Genauigkeit im Ergebnis zeigen wir in Abbildung 5.5. Es zeigt sich, daß der Einsatz iterativer Löser keinen Einfluß auf die Genauigkeit im Ergebnis hat. Wir haben die Fehler, die durchschnittliche Anzahl der Krylov-Schritte sowie die Schrittzahlen in den folgenden Tabellen dargestellt.

Die Tabellen 5.1 und 5.3 verdeutlichen noch einmal, daß die Einführung von Krylovtechniken das Ergebnis nicht signifikant beeinflußt, die Schrittzahlen gleichen sich weitgehend, und auch die globalen Fehler sind nahezu identisch, wie man ja bereits in Abbildung 5.5 feststellen konnte. Allerdings zahlt man dafür einen hohen Preis – es sind im Durchschnitt mehr als 100 Iterationsschritte durchzuführen. Dies bestätigt eindrucksvoll die im theoretischen Teil des Kapitels gewonnenen Erkenntnisse, daß nämlich ohne Vorkonditionierung wenigstens so viele Krylovschritte durchgeführt werden müs-

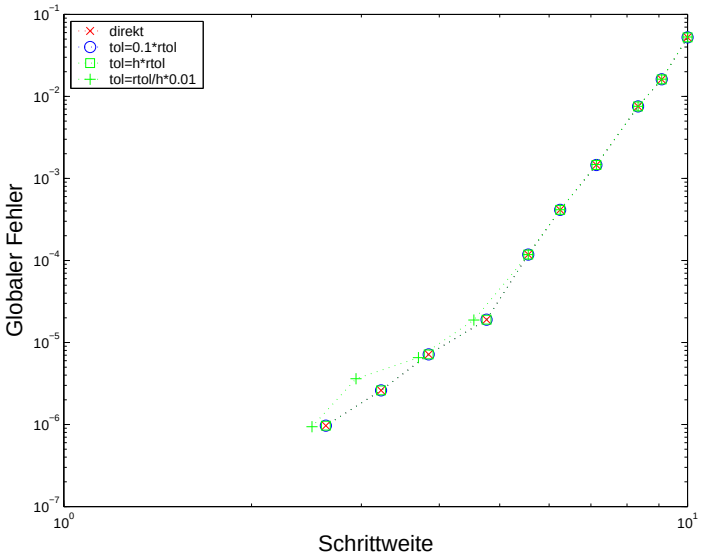


Abbildung 5.5: Resultate für RODAS mit Krylovraum-Techniken.

sen, wie algebraische Variablen vorhanden sind. Wir haben hier nämlich 120 differentielle und 110 algebraische Variable. Der typische Verlauf einer Krylovraum-Iteration wird in Abbildung 5.6 gezeigt. Erst nach ungefähr 100 Schritten fällt das Residuum deutlich, die Iteration bricht dann auch sehr schnell ab.

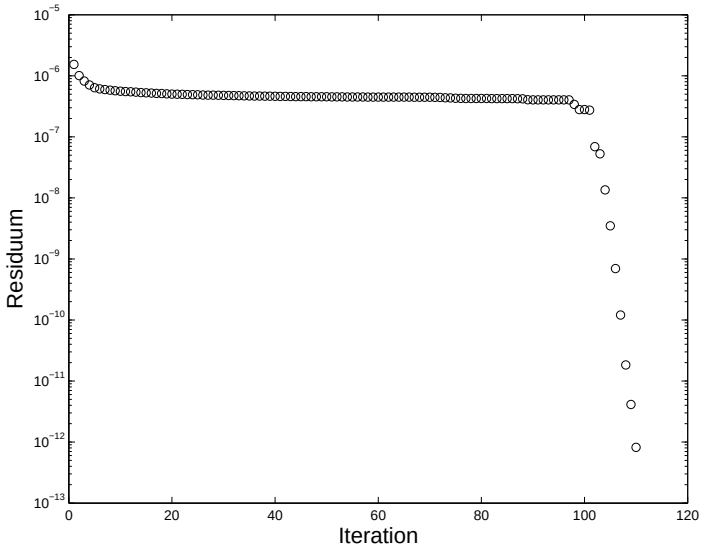


Abbildung 5.6: Typischer Verlauf des Residuums bei einer Iteration mit GMRES.

Abbildung 5.6 erklärt auch, daß der Versuch, mit deutlich weniger Iterationen auszukommen, scheitert. Wir haben die Maximalzahl von Iterationen pro GMRES-Aufruf auf 30 begrenzt, und mußten feststellen, daß die Ergebnisse unbrauchbar sind.

Ein Ausweg stellt, wie auch eingangs des Kapitels erwähnt, die Verwendung von Vorkonditionierern, oder aber die direkte Auflösung der algebraischen Gleichungen, dar.

	$TOL(GMRES)$			
$RTOL$	0	$0.1RTOL$	$h * RTOL$	$0.01 * RTOL/h$
$1.00E-02$	$5.248E-02$	$5.251E-02$	$5.249E-02$	$5.288E-02$
$3.59E-03$	$1.613E-02$	$1.613E-02$	$1.613E-02$	$1.620E-02$
$1.29E-03$	$7.551E-03$	$7.545E-03$	$7.550E-03$	$7.579E-03$
$4.64E-04$	$1.458E-03$	$1.458E-03$	$1.458E-03$	$1.475E-03$
$1.67E-04$	$4.137E-04$	$4.134E-04$	$4.137E-04$	$4.152E-04$
$5.99E-05$	$1.178E-04$	$1.178E-04$	$1.178E-04$	$1.178E-04$
$2.15E-05$	$1.896E-05$	$1.899E-05$	$1.896E-05$	$1.879E-05$
$7.74E-06$	$7.169E-06$	$7.171E-06$	$7.170E-06$	$6.537E-06$
$2.78E-06$	$2.607E-06$	$2.608E-06$	$2.607E-06$	$3.618E-06$
$1.00E-06$	$9.677E-07$	$9.679E-07$	$9.677E-07$	$9.400E-07$

Tabelle 5.1: Der maximale globale Fehler im Verlauf der Integration für die verschiedenen Krylov-Implementierungen einschließlich RODAS selbst.

	$TOL(GMRES)$			
$RTOL$	0	$0.1RTOL$	$h * RTOL$	$0.01 * RTOL/h$
$1.00E-02$	0.00	105.85	104.46	106.54
$3.59E-03$	0.00	106.00	105.71	106.86
$1.29E-03$	0.00	106.50	106.07	107.14
$4.64E-04$	0.00	106.62	106.25	107.31
$1.67E-04$	0.00	106.82	106.53	107.35
$5.99E-05$	0.00	107.11	107.00	107.50
$2.15E-05$	0.00	107.25	107.40	107.57
$7.74E-06$	0.00	107.44	107.52	116.31
$2.78E-06$	0.00	107.47	107.80	117.61
$1.00E-06$	0.00	107.46	107.81	118.69

Tabelle 5.2: Die durchschnittliche Anzahl der Iterationen für die verschiedenen Krylov-Implementierungen einschließlich RODAS selbst.

5.6 Zusammenfassung

Wir haben uns in diesem Kapitel mit einem modernen Zeitintegrationsverfahren befaßt. Steife Systeme erfordern (zumeist, siehe Kapitel 6) implizite Verfahren zur effizienten Zeitintegration, klassische explizite Verfahren sind aufgrund von Schrittweitenbeschränkungen aus Stabilitätsgründen weniger geeignet.

Mit Krylov-Methoden versucht man den Grad an Implizität so klein wie möglich zu halten, lediglich die näherungsweise Lösung linearer Gleichungssysteme ist erforderlich. Wir betrachten die Erweiterung der erfolgreichen ODE-Methode auf differential-algebraische Systeme vom Index 1. Es bieten sich zwei Zugänge an, wir zeigen, daß beide in gewissem Sinne äquivalent sind. Zugleich folgern wir, daß die typischen Restriktionen an die Dimension der Krylov-Räume bei differential-algebraischen Systemen nicht gestellt werden sollten.

	$TOL(GMRES)$			
$RTOL$	0	$0.1RTOL$	$h * RTOL$	$0.01 * RTOL/h$
$1.00E-02$	10.00	10.00	10.00	10.00
$3.59E-03$	11.00	11.00	11.00	11.00
$1.29E-03$	12.00	12.00	12.00	12.00
$4.64E-04$	14.00	14.00	14.00	14.00
$1.67E-04$	16.00	16.00	16.00	16.00
$5.99E-05$	18.00	18.00	18.00	18.00
$2.15E-05$	21.00	21.00	21.00	22.00
$7.74E-06$	26.00	26.00	26.00	27.00
$2.78E-06$	31.00	31.00	31.00	34.00
$1.00E-06$	38.00	38.00	38.00	40.00

Tabelle 5.3: Die Schrittzahl für die verschiedenen Krylov-Implementierungen einschließlich RODAS selbst.

Als Anwendungsbeispiel untersuchen wir eine Problemstellung aus dem Bereich der Viskoelastizität – die Scherung einer Scheibe. Das entstehende große differential-algebraische System wird mit der vorgeschlagenen Krylov-Technik gelöst. Die theoretischen Resultate für Systeme mit einer großen Anzahl von algebraischen Gleichungen finden dabei eindrucksvoll Bestätigung.

Kapitel 6

Numerische Simulation eines Modells aus der Neurobiologie auf paralleler Rechnerarchitektur

Wir beschäftigen uns in diesem Kapitel mit einer Themenstellung aus der Neurobiologie, in der das Wachstum von Nervenzellen im menschlichen Gehirn, nachdem sie durch einen Unfall durchtrennt wurden, modelliert wird. Das Wachstum der Nervenzellen wird durch verschiedene chemische Substanzen stimuliert. Dies führt auf ein gekoppeltes System von gewöhnlichen Differentialgleichungen für die Nervenzellen und partiellen Differentialgleichungen vom parabolischen Typ für die chemischen Substanzen.

Die Diskretisierung der Ortsableitungen führt auf ein großes System steifer Differentialgleichungen. Zur Zeitintegration wird ein explizites Runge-Kutta-Verfahren herangezogen, eine Runge-Kutta-Tschebyscheff-Methode, die ein der Problemstellung angepaßtes ausgedehntes Stabilitätsgebiet entlang der reellen Achse besitzt.

Die parallele Implementierung erfolgte auf einer SGI Origin 3800 (Teras, Sara Computing Center, Amsterdam). Als paralleles Programmiermodell wurde OpenMP verwendet.

Bemerkung 6.0.1. *Diese Thematik wurde während eines Aufenthalts am CWI, Amsterdam, im Herbst 2001 zusammen mit Ben P. Sommeijer bearbeitet, siehe [118], [120].*

6.1 Einleitung

Beim Menschen vollzieht sich die Entwicklung des Nervensystems durch den Aufbau immer neuer Nervenleitungen im Großhirn. Dies geschieht durch Wachstum einer Nervenbahn – dem Axon – von einer Quell-Zelle zu einer Ziel-Zelle. Kontrolliert wird dieses Wachstum durch chemische Substanzen. Die Zielzelle stößt chemische Substanzen aus, um die Bewegung des Axons in die geeignete Richtung zu stimulieren. Der Wachstumskopf des Axons reagiert auf Konzentrationsunterschiede in diesen Substanzen - er wächst in Richtung des Gradienten.

Insbesondere nach Schädigungen des Gehirns steht die Aufgabe an, Regionen wieder neu zu verbinden. Beobachtet wird dabei eine Bündelung der Axonen, die bei Erreichen der Zielregionen dann wieder aufgehoben wird. Auch dieses Verhalten wird durch chemische Substanzen hervorgerufen, die von den Wachstumsköpfen der Axonen ausgestoßen werden. Zu Beginn des Wachstumsprozes-

ses stoßen die Wachstumsköpfe anziehende Chemikalien aus, in der letzten Phase jedoch abstoßende Chemikalien, so daß es gerade zur Bündelung bzw. Entflechtung kommt.

Die neurologische Grundlage dieses Kapitels bildet die Arbeit [51], wo Hentschel & Van Ooyen ein mathematisches Modell für obigen Prozeß entwickeln. Wir greifen mit geringfügigen Modifikationen auf dieses Modell zurück. Die räumliche und zeitliche Verteilung der Konzentrationen der anziehenden und abstoßenden Chemikalien unterliegt im wesentlichen den Gesetzen der Diffusion und wird durch parabolische partielle Differentialgleichungen beschrieben. Die Bewegung der Axonen wird durch gewöhnliche Differentialgleichungen beschrieben. Gekoppelt sind diese Gleichungen zum einen durch Quellterme in den parabolischen Gleichungen, die von der Position der Axonen abhängen, zum anderen in der rechten Seite der gewöhnlichen Differentialgleichungen, wo Gradienten der Konzentrationen auftreten. Diese Gleichungen werden daher auch als Gradientengleichungen bezeichnet.

Zur numerischen Lösung greifen wir auf Techniken zurück, die auf den in [106] (oder auch [63]) entwickelten beruhen. Zur Ortsdiskretisierung der parabolischen Gleichungen nutzen wir finite Differenzen 2. Ordnung. Die Gradienten approximieren wir durch bilineare Interpolation. Das Resultat ist ein System gewöhnlicher Differentialgleichungen. Zur Zeitintegration nutzen wir eine Runge-Kutta-Tschebyscheff-Methode [92]. Dieses explizite Verfahren wurde speziell für mäßig steife Differentialgleichungen entwickelt, wo die Jacobi-Matrix nahezu normal ist und die Eigenwerte in einem schmalen Streifen entlang der negativen reellen Achse liegen. Unser neurologisches Problem erfüllt diese Anforderungen.

Der so vorgegebene numerische Algorithmus wurde auf einem Parallelrechner – Teras, aus der SGI Origin 3800er Serie – implementiert. Die dabei gewonnen Erkenntnisse über die effiziente Nutzung dieser Maschine für Aufgabenstellungen des Hochleistungsrechnens stellen ein wichtiges Resultat dieses Kapitels dar.

Das Kapitel gliedert sich wie folgt: In Abschnitt 2 beschreiben wir die Problemstellung, in Abschnitt 3 die Diskretisierung und das Zeitintegrationsverfahren. In den Abschnitten 4, 5 und 6 diskutieren wir nacheinander verschiedene Parallelisierungs-Strategien, die Resultate der Simulation sowie die Effizienz der Parallelisierung auf Teras. Im letzten Abschnitt erfolgt eine abschließende Diskussion.

6.2 Modellierung

Wir beschreiben hier das zugrundeliegende mathematische Modell, eine detaillierte Darstellung findet sich in [51], [106].

Die Konzentrationen der 3 Chemikalien werden mit ρ_t (von den Zielzellen, engl. **t**arget, ausgestoßene anziehende Substanz), ρ_a (von den Axonen abgesonderte anziehende Substanz, engl. **a**ttractant) und ρ_r (von den Axonen abgesonderte abstoßende Substanz, engl. **r**epellant), bezeichnet. Diese Konzentrationen hängen von Zeit (t) und Ort ($\mathbf{x} = (x, y) \in \mathbb{R}^2$) ab und genügen den parabolischen partiellen Differentialgleichungen

$$\begin{aligned} \frac{\partial \rho_t}{\partial t} &= D_t \Delta \rho_t - \kappa_t \rho_t + S_t, \\ \frac{\partial \rho_a}{\partial t} &= D_a \Delta \rho_a - \kappa_a \rho_a + S_a, \\ \frac{\partial \rho_r}{\partial t} &= D_r \Delta \rho_r - \kappa_r \rho_r + S_r. \end{aligned} \tag{6.1}$$

Die Diffusionskoeffizienten D_* und die κ_* in den linearen Zerfallstermen sind als räumlich und zeitlich konstant angenommen. Die Quellterme S_* hängen sowohl von der Lösung als auch vom Ort ab. Wir stützen uns zu ihrer Modellierung auf den Zugang von [51] und verwenden Diracsche δ-Funktionen im Ansatz. Die Zielzellen und die Wachsstumsköpfe sind also Punktquellen.

Für die von den Zielzellen emittierte Substanz (ρ_t) verwenden wir den Quellterm

$$S_t = \sum_{j=1}^{N_t} \sigma_t \delta(\mathbf{x} - \mathbf{x}_j^t), \tag{6.2}$$

wobei N_t die Anzahl der Zielzellen bezeichnet, die räumlich fest in den Punkten $\mathbf{x}_j^t$ liegen. Die Intensität σ_t kann prinzipiell von allen 3 Substanzen abhängen. Hier wählen wir σ_t als konstant.

Für die Quellterme S_a und S_r verwenden wir ähnliche Ausdrücke:

$$S_a = \sum_{j=1}^{N_A} \sigma_a \delta(\mathbf{x} - \mathbf{x}_j^A), \tag{6.3}$$

$$S_r = \sum_{j=1}^{N_A} \sigma_r \delta(\mathbf{x} - \mathbf{x}_j^A). \tag{6.4}$$

Die Summation läuft hier bis N_A, der Anzahl der Axonen. Die räumliche Lage des j-ten Axons ist durch $\mathbf{x}_j^A$ gegeben. Zu bemerken ist, daß die Lage der Quellen $\mathbf{x}_j^A$ aufgrund der Bewegung der Wachstumsköpfe variiert, während die Quellen $\mathbf{x}_j^t$ fest sind. Wir wählen die Intensität σ_a konstant, während σ_r von σ_t abhängt. Damit berücksichtigen wir, daß der Ausstoß der abstoßenden Substanz erst in der Nähe der Zielzellen beginnt. Die Details findet man in Abschnitt 6.5 über die Simulationsergebnisse.

Die Bewegung der Axonen wird durch 3 Effekte beeinflußt: Den Hauptteil bildet eine Bewegung in Richtung der Zielzellen, die durch die Substanz ρ_t bewirkt wird. Außerden beeinflussen sich die Axonen gegenseitig: Der Ausstoß der anziehenden Substanz ρ_a bewirkt eine Bündelung der Axonen. Erreichen diese die Zielzone, so wird dieser Effekt durch den vermehrten Ausstoß der abstoßenden Chemikalie ρ_r mehr als aufgehoben, und die Axone driften auseinander. Beschrieben wird dies durch die Gradientengleichung

$$\frac{d\mathbf{x}_j^A(t)}{dt} = \lambda_t \nabla\rho_t(\mathbf{x}_j^A(t), t) + \lambda_a \nabla\rho_a(\mathbf{x}_j^A(t), t) - \lambda_r \nabla\rho_r(\mathbf{x}_j^A(t), t), \quad j = 1, \ldots, N_A, \tag{6.5}$$

wobei λ_t, λ_a und λ_r als konstant und positiv vorausgesetzt sind. In der rechten Seite treten dabei die Gradienten von ρ_t, ρ_a und ρ_r auf, ausgewertet an der Position der Axonen. Da wir die parabolischen Gleichungen mit einem Finite-Differenzen-Verfahren lösen, kennen wir die Konzentrationen ρ_* nur in den Gitterpunkten. Im allgemeinen werden wir daher unter Verwendung der Gitterwerte interpolieren müssen. Diese Interpolationsprozedur wird in Abschnitt 6.3.2 beschrieben.

Implementiert wurde das Problem in 2 Raumdimensionen. Im Raum wurde ein quadratisches Gebiet gewählt, das groß genug ist, um homogene Dirichletbedingungen auf dem Rand als realistische Wahl für die Konzentrationen anzusehen.

6.3 Numerische Methoden

Wir stellen hier die numerischen Techniken, die wir zur Lösung des Problems heranziehen, dar. Im einzelnen geht es dabei um die Diskretisierung des Laplace-Operators, die Interpolation der Gradienten und die Approximation der δ-Funktionen in den Quelltermen.

Eine besondere Schwierigkeit stellt der sogenannte "self-boost" einzelner Zellen dar, wir werden später auf die Behandlung dieses numerischen Phänomens zurückkommen. Details findet man in [63] und [106].

6.3.1 Ortsdiskretisierung des Laplace-Operators

Das Problem wird auf dem quadratischen Gebiet $\Omega = [0, L]^2 \subset \mathbb{R}^2$ der Länge L in $\mathbb{R}^2$ gelöst. Zur räumlichen Diskretisierung nutzen wir die Linienmethode (MOL - method of lines). Die Lösungen $\rho_* \in \{\rho_t, \rho_a, \rho_r\}$ der parabolischen Differentialgleichungen werden auf einem äquidistanten Gitter mit der Gitterweite $h = L/N$ approximiert.

$$x_i = hi, \quad y_j = hj, \quad i = 0, \ldots, N, \quad j = 0, \ldots, N \tag{6.6}$$

$$\rho_{*,i,j} \approx \rho_*(x_i, y_j) \tag{6.7}$$

$$\rho_{*,i,j} = 0 \quad \text{für } i = 0 \text{ oder } i = N \text{ oder } j = 0 \text{ oder } j = N. \tag{6.8}$$

Somit kann der Laplace-Operator durch den symmetrischen Differenzenquotienten 2. Ordnung in beiden Raumrichtungen angenähert werden, womit der Standard 5-Punkt-Stern erhalten wird.

6.3.2 Interpolation

Die Berechnung der Gradienten in (6.5) erfolgt in 2 Stufen. Zunächst approximieren wir aus den Konzentrationen $\rho_{*,i,j}$ in den Gitterpunkten die Gradienten in den Gitterpunkten mit Hilfe des symmetrischen Differenzenquotienten

$$\nabla \rho_*(x_i, y_j) \approx \frac{1}{2h} \begin{pmatrix} \rho_{*,i+1,j} - \rho_{*,i-1,j} \\ \rho_{*,i,j+1} - \rho_{*,i,j-1} \end{pmatrix}. \tag{6.9}$$

Im zweiten Schritt wird aus den diskreten Gradienten der Gradient im Innern der Gitterquadrate durch eine geeignete Interpolationsvorschrift bestimmt.

Wir bemerken, daß der diskrete Gradient auf diese Weise nur in inneren Gitterpunkten approximiert werden kann. In den direkt an den Rand grenzenden Gitterquadraten müßte daher eine andere Interpolationsvorschrift verwendet werden. Da das Gebiet Ω jedoch in ausreichender Größe gewählt wurde, so daß kein Axon ein Randquadrat erreicht, können wir unsere Betrachtung auf die inneren Gitterquadrate beschränken.

Der Gradient im Punkt $\mathbf{x}_j^A$ ergibt sich formal durch Auswertung einer geeigneten Interpolationsfunktion I_h für die diskreten Gradienten. Die diskretisierte Gleichung (6.5) lautet somit

$$\frac{d\mathbf{x}_j^A(t)}{dt} = \lambda_t \cdot I_h \left[\nabla_h \rho_{h,t}\right](\mathbf{x}_j^A) + \lambda_a \cdot I_h \left[\nabla_h \rho_{h,a}\right](\mathbf{x}_j^A) - \lambda_r \cdot I_h \left[\nabla_h \rho_{h,r}\right](\mathbf{x}_j^A), \quad j = 1, \ldots, N_A. \tag{6.10}$$

Aus Gründen der Effizienz und der besseren Parallelisierbarkeit interpolieren wir nur lokal, d.h., wir nutzen nur die Werte auf den Ecken des $\mathbf{x}_j^A$ einschließenden Quadrats. Wechselt das Axon von einem

Gitterquadrat ins nächste, so ändern sich die Stützpunkte der Interpolation. Daraus resultiert eine verminderte Glattheit der rechten Seiten, die unter Umständen die Effizienz der Schrittweitensteuerung in der Zeitintegration ungünstig beeinflussen kann.

Wir verwenden einen stückweise bilinearen Ansatz zur Interpolation, in jedem Gitterquadrat wird ein Ansatz

$$I_h f(x,y) = a + bx + cy + dxy \tag{6.11}$$

verwendet, wobei die Koeffizienten a, b, c, d vom Gitterquadrat abhängen. Das Interpolationspolynom (6.11) ergibt sich eindeutig aus den Funktionswerten in den Eckpunkten des Gitterquadrates. Auf diese Weise garantieren wir auch die Stetigkeit der Interpolierenden in den Kanten der Gitterquadrate, denn dort ist die Interpolierende ein lineares Polynom in einer Variablen und somit eindeutig durch die Vorgabe von Funktionswerten in den zwei Ecken (der Kante) bestimmt.

Das Resultat ist eine 12-Punkte-Formel für den Gradienten an beliebiger Position. Neben den 4 Eckpunkten der Gitterzelle kommen jeweils 4 Gitterpunkte für $\frac{\partial}{\partial x}$ und $\frac{\partial}{\partial y}$ hinzu. Abbildung 6.1 illustriert die Vorgehensweise.

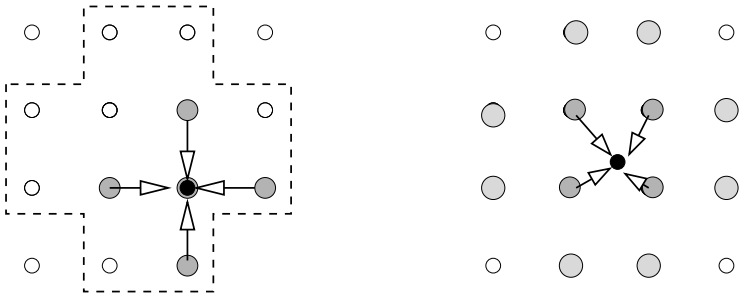


Abbildung 6.1: Der 12-Punkte-Stern zur Interpolation der Gradienten.

6.3.3 Implementierung der δ-Funktion

Die Diracsche δ-Funktion wird durch eine geeignete stetige Funktion ersetzt, die wir dann in den Gitterpunkten auswerten. Um die Gesamtquellstärke zu erhalten, skalieren wir diese Funktion so, daß ihr Integral eins beträgt (siehe [23]).

Wir haben verschiedene Ansätze betrachtet, unter anderem Exponentialfunktionen und lineare B-Splines. In beiden Fällen sind die Ansatzfunktionen vom Gitter abhängig.

Die Exponentialfunktion

Ausgangspunkt ist

$$\delta_h(\mathbf{x}) = \frac{\gamma}{\pi} \exp(-\gamma \|\mathbf{x}\|^2), \tag{6.12}$$

wobei $\|\mathbf{x}\|$ die Euklidische Norm von $\mathbf{x}$ bezeichnet und $\gamma = h^{-2}$. Diese Funktion hat allerdings unbeschränkten Träger. Um den rechnerischen Aufwand zu verringern, schränken wir den Träger auf ein Teilgitter mit 8×8 Gitterpunkten ein, wobei $\mathbf{x}_j^A$ gerade in der zentralen Gitterzelle liegen möge. Die Funktionswerte außerhalb des 8×8-Teilgitters sind vom Absolutbetrage höchstens das $\exp(-16) \approx 10^{-7}$-fache des Maximalwertes von δ und somit vernachlässigbar.

Lineare B-Splines

Die δ-Funktion wird durch ein Tensorprodukt linearer B-Splines mit dem Träger $[-h, h]$ ersetzt:

$$\delta_h(x,y) = B_h(x)B_h(y), \text{ wobei} \tag{6.13}$$

$$B_h(x) := \begin{cases} \frac{h-|x|}{h^2} & \text{für } |x| \leq h \\ 0 & \text{sonst} \end{cases} . \tag{6.14}$$

Möge das Axon $\mathbf{x}_k^A$ im Gitterquadrat $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$ mit $\mathbf{x}_k^A = (x_i + \theta_x h, y_j + \theta_y h)$ liegen, wobei $0 \leq \theta_x, \theta_y < 1$. Dann sind die Beiträge der approximierten δ-Funktion zu den Gitterpunkten gerade die Funktionswerte $\delta_h(x - x_k^A, y - y_k^A)$ in den 4 Ecken des Gitterquadrats:

$$\begin{aligned} \delta_{h,i,j} &= \frac{(1-\theta_x)(1-\theta_y)}{h^2} \\ \delta_{h,i+1,j} &= \frac{\theta_x(1-\theta_y)}{h^2} \\ \delta_{h,i,j+1} &= \frac{(1-\theta_x)\theta_y}{h^2} \\ \delta_{h,i+1,j+1} &= \frac{\theta_x\theta_y}{h^2}. \end{aligned} \tag{6.15}$$

Dieser Zugang ermöglicht eine geometrische Interpretation: Betrachtet man eine Aufteilung des Gitterquadrats in 4 Rechtecke durch zwei Geraden, die parallel zur x- bzw. y-Achse durch den Punkt $\mathbf{x^A}$ verlaufen, so ist der Beitrag zu einem Eckpunkt gerade gleich der Fläche des gegenüberliegenden Rechtecks. Dies stimmt mit der von Dallon (siehe [23]) vorgeschlagenen Approximation überein.

6.3.4 Zeitintegration

Nach der Diskretisierung im Ort liegt ein System gewöhnlicher Differentialgleichungen vor. Die Lösung partieller Differentialgleichungen mittels einer solchen Semidiskretisierung bietet den Vorteil, auf hocheffiziente Zeitintegrationsverfahren zurückgreifen zu können. Diese lösen Anfangswertprobleme in der Form

$$\frac{dU}{dt} = F(t,U), \quad t > 0, \quad U(0) = U_0, \tag{6.16}$$

wobei wir in der folgenden Darstellung das Argument t in F weglassen, da dies zum einen keine Einschränkung darstellt (man vergleiche [98], S. 14) und in unserem Fall ohnehin ein autonomes System vorliegt.

Typisch ist die Verwendung expliziter Verfahren für nichtsteife Probleme und die Verwendung impliziter Verfahren für steife Systeme. Wir wählen hier einen etwas untypischen Ansatz, wir lösen ein (mild) steifes Problem mit einem expliziten Verfahren. Begründet ist dies in der speziellen Eigenwertstruktur der Jacobi-Matrix der rechten Seite unserer Differentialgleichung, der dominierende Diffusionsterm bewirkt, daß die Eigenwerte in einem schmalen Streifen um die negative reelle Achse liegen.

Wir nutzen zur Lösung von (6.16) eine Runge-Kutta-Tschebyscheff-Methode (RKC), im Code RKC [92] von Sommeijer, Shampine und Verwer implementiert. Das C in RKC rührt im übrigen von der englischen Transkription Chebyshev. Diese Methode besitzt ein ausgedehntes Stabilitätsgebiet entlang der reellen Achse. Genau genommen handelt es sich um eine Klasse von Methoden der Ordnung 2 mit variabler Stufenzahl. Das Einfügen zusätzlicher Stufen wird hier nicht zur Erhöhung der Ordnung genutzt, sondern zur Erweiterung des Stabilitätsgebiets entlang der reellen Achse.

Bemerkung 6.3.1. *Ein weiterer Grund für die Auswahl eines expliziten Verfahrens ist die einfache Parallelisierbarkeit. Ein anderer Ansatz ist die Forderung nach weitergehender Parallelität in der Methode – hier sind eine ganze Reihe von Verfahren konstruiert worden, die die parallele Nutzung einer kleinen bis mittleren Anzahl von Prozessoren durch parallele Stufen zulassen, man siehe [11], [103], [79], [80].*

Die s-stufige RKC-Methode ist gegeben durch [92]

$$\begin{aligned} Y_0 &= U_m, \\ Y_1 &= Y_0 + \tilde{\mu}_1 \tau F_0, \\ Y_j &= (1-\mu_j-\nu_j)Y_0 + \mu_j Y_{j-1} + \nu_j Y_{j-2} + \tilde{\mu}_j \tau F_{j-1} + \tilde{\gamma}_j \tau F_0, \quad j = 2, \ldots, s, \\ U_{m+1} &= Y_s, \end{aligned} \tag{6.17}$$

wobei $F_j = F(Y_j)$. In einem Schritt mit der Zeitschrittweite τ wird, ausgehend von einer Näherung U_m an die Lösung in $t = t_m$ eine neue Näherung U_{m+1} an die Lösung in $t_{m+1} = t_m + \tau$ berechnet.

Die Koeffizienten des Verfahrens werden nach [92] so bestimmt, daß zum einen Ordnung 2 gewährleistet ist, und zum anderen die Stabilitätsfunktion in einem möglichst ausgedehnten Intervall entlang der negativen reellen Achse vom Betrage kleiner (gleich) eins ist. Diese Problemstellung wird für Ordnung 1 bekanntlich von den Tschebyscheff-Polynomen gelöst – unter allen Polynomen p vom Grad s mit $p(0) = 1, p'(0) = 1$ besitzt das transformierte Tschebyscheff-Polynom $p(x) = T_s(x/s^2 + 1)$ das längste Stabilitätsgebiet (Länge: $2s^2$) entlang der reellen Achse. Die Transformation des Argumentes überführt gerade das Intervall $[-1, 1]$ (hier oszillieren die Tschebyscheff-Polynome zwischen -1 und 1) in das Intervall $[-2s^2, 0]$

Das Tschebyscheff-Polynom T_k vom Grad k ist dabei durch die Drei-Term-Rekursion $T_0(x) = 1, T_1(x) = x, T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), 2 \leq k \leq j$ gegeben.

Um Ordnung 2 zu erreichen, unterwerfen wir zusätzlich zum Argument auch den Funktionswert des Tschebyscheff-Polynoms einer affinen Transformation. Wir verwenden den Ansatz

$$p(x) = aT_s(bx + c) + d \tag{6.18}$$

mit $s \geq 2$. Zu erfüllen ist für Ordnung 2

$$1 = p(0) = aT_s(c) + d \tag{6.19}$$

$$1 = p'(0) = abT_s'(c) \tag{6.20}$$

$$1 = p''(0) = ab^2 T_s''(c). \tag{6.21}$$

Mit der Wahl von $c = 1$, $d = 0$ ergibt sich mit $T_s(1) = 1$, $T_s'(1) = s^2$ und $T_s''(1) = s^2(s^2-1)/3$ aus den Gleichungen (6.20) und (6.21) gerade $b \approx 3/s^2$, womit sich die Länge des Stabilitätsgebietes auf die Größenordnung $2/3s^2$ verringert. Da wir nur einen zusätzlichen freien Parameter benötigen, verwenden wir den verbleibenden freien Parameter, um $|p(z)| < 1 - \mathcal{O}(\varepsilon)$ (anstelle von $|p(z)| < 1$) in einem möglichst großen Intervall entlang der negativen reellen Achse (natürlich eine Umgebung der Null ausschließend) zu fordern. Dazu setzen wir $c = 1 + \varepsilon/s^2$. Die restlichen Parameter ergeben sich dann wie folgt:

$$a = \frac{T_s''(c)}{T_s'(c)^2} \tag{6.22}$$

$$b = \frac{T_s'(c)}{T_s''(c)} \tag{6.23}$$

$$d = 1 - \frac{T_s(c)T_s''(c)}{T_s'(c)^2}. \tag{6.24}$$

Die Länge des Stabilitätsgebietes beträgt $2/b + \mathcal{O}(\varepsilon) \approx 2/3s^2$. In diesem Intervall (wieder eine Umgebung der Null ausschließend) liegt das Stabilitätspolynom zwischen $d - a$ und $d + a$ mit

$$d + a = 1 - \frac{1}{3}\varepsilon + \mathcal{O}\left(\varepsilon/s^2\right) + \mathcal{O}\left(\varepsilon^2\right) \tag{6.25}$$

$$d - a = \frac{1}{3} + \frac{2}{3s^2} - \frac{1}{45}\varepsilon + \mathcal{O}\left(\varepsilon/s^2\right) + \mathcal{O}\left(\varepsilon^2\right). \tag{6.26}$$

Wir nutzen hier eine Implementierung mit $\varepsilon = 2/13$, dies resultiert im Koeffizientensatz für (6.17)

$$\begin{aligned} &\tilde{\mu}_1 = a_2 b_s, \;\; \mu_j = \frac{2a_j c_s}{a_{j-1}}, \;\; \nu_j = \frac{-a_j}{a_{j-2}}, \;\; \tilde{\mu}_j = \frac{2a_j b_s}{a_{j-1}}, \\ &\tilde{\gamma}_j = -\left(1 - a_{j-1} T_{j-1}(w c_s)\right)\tilde{\mu}_j \quad \text{für } 2 \le j \le s, \;\; \text{wobei } a_0 := a_1 := a_2. \end{aligned} \tag{6.27}$$

Diese Koeffizienten stehen in analytischer Form für $s \geq 2$ zur Verfügung. Die Länge des Stabilitätsintervalls hängt quadratisch von der Zahl der Stufen ab. Interpretiert man die k-fache Hintereinanderausführung eines s-stufigen Einschrittverfahrens mit der Schrittweite h/k als ein Verfahren mit ks Stufen für die Schrittweite h, so hängt in diesem Fall die Länge des Stabilitätsgebietes nur linear von der Anzahl der Stufen ab, RKC weist hingegen ein quadratisches Wachstum auf.

Der lokale Fehler variiert nur gering mit der Anzahl der Stufen, was folgende Implementierung nahelegt: Nach Maßgaben der Fehlerschätzung wird eine Schrittweite bestimmt, und dann die Stufenzahl des Verfahrens so gewählt, daß das Verfahren innerhalb des Stabilitätsgebietes arbeitet.

Für mild-steife Probleme arbeitet man im Bereich von 20 bis 100 Stufen, so daß die Drei-Term-Rekursion für die Stufen von großer Bedeutung ist, da andernfalls bei einer vollbesetzten Koeffizientenmatrix ein unvertretbar hoher Speicheraufwand für die internen Stufen entstünde.

Der FORTRAN-Quellcode für RKC kann unter der Adresse ftp://ftp.cwi.nl/pub/bsom/rkc heruntergeladen werden.

6.3.5 Das Phänomen “self-boost”

Mit den Techniken der vergangenen Abschnitte kann eine sequentielle Implementierung erstellt werden. Erste Experimente zeigen jedoch für bestimmte Konfigurationen unbefriedigende Ergebnisse: Einige Axonen erreichen stationäre Punkte bevor sie die Zielzellen erreicht haben, teilweise bereits nach wenigen Integrationsschritten.

Zum Verständnis der zugrundeliegenden Effekte betrachten wir ein einzelnes Axon unter dem Einfluß einer einzelnen Chemikalie (man siehe auch [63] für eine detaillierte Diskussion dieses Sachverhaltes).

$$\begin{aligned} \dot{\rho} &= L\rho + S(\mathbf{x}^A), \\ \dot{\mathbf{x}}^A &= \nabla|_{\mathbf{x}^A} \rho. \end{aligned} \tag{6.28}$$

Dabei sei L ein Differentialoperator (hier: Diffusion und linearer Zerfall) und S der Quellterm. Wird diese Gleichung diskretisiert, so werden ρ, L, S und ∇ durch diskrete Versionen ρ_h, L_h, S_h, und ∇_h ersetzt.

Betrachten wir noch einmal die Modellannahmen: Die Axonen sollten einander durch den Ausstoß von Chemikalien derart beeinflussen, daß es zur Bündelung bzw. zum Auseinanderdriften kommt. Unser Ansatz führt allerdings dazu, daß ein Axon auch durch die von ihm selbst ausgestoßenen Chemikalien beeinflußt wird. Wir bezeichnen diesen Effekt als “self-boost”. Hebt nun der “self-boost”

gerade die anziehende Wirkung der Zielzellen auf, so kommt es zu den oben beschriebenen Phänomenen.

Um "self-boost" zu vermeiden, genügt es, wenn die Diskretisierung und der auf die Gitterpunkte eingeschränkte Quellterm S_h durch folgende Bedingung gekoppelt sind:

$$I_h[\nabla_h S_h(\mathbf{x}^A)](\mathbf{x}^A) \;=\; 0. \tag{6.29}$$

Man fordert im Prinzip, daß die Beiträge des eigenen Quellterms zum Gradienten verschwinden.

Lemma 6.3.1. *Für die Kombination von linearen B-Splines für den Gradienten, symmetrischem Differenzenquotient für die Gradienten in den Gitterpunkten und bilinearer Interpolation ist die Bedingung* (6.29) *erfüllt.*

Beweis. Wir führen den Beweis für die Ableitungen in x-Richtung, in y-Richtung ergibt sich das Ergebnis analog. Sei $\mathbf{x}^A = (x^A, y^A)$ im Gitterquadrat $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$ gelegen mit

$$\begin{aligned} x^A =& x_i + \theta_x h \\ y^A =& y_i + \theta_y h. \end{aligned} \tag{6.30}$$

Wir drücken die Ableitung nach x von S_h durch die Ableitungen nach x in den Gitterpunkten aus:

$$\begin{aligned} \partial_x|_{\mathbf{x}^A} S_h(\mathbf{x}^A) \;=\; & \{(1-\theta_x)[(1-\theta_y)\partial_{x,i,j} + \theta_y \partial_{x,i,j+1}] \\ & + \theta_x[(1-\theta_y)\partial_{x,i+1,j} + \theta_y \partial_{x,i+1,j+1}]\} S_h(\mathbf{x}^A). \end{aligned}$$

Wir ersetzen nun ∂_x durch den diskretisierten Differentialoperator ((6.9)) und erhalten für die rechte Seite

$$\frac{1}{2h}\left\{(1-\theta_x)[(1-\theta_y)\delta_{h,i+1,j} + \theta_y \delta_{h,i+1,j+1}] - \theta_x[(1-\theta_y)\delta_{h,i,j} + \theta_y \delta_{h,i,j+1}]\right\}.$$

Letztendlich ersetzen wir die δ_h-Werte in den Gitterpunkten wie in (6.15) definiert und erhalten

$$\frac{1}{2h}\frac{1}{h^2}\left\{(1-\theta_x)[(1-\theta_y)\theta_x(1-\theta_y) + \theta_y\theta_x\theta_y] - \theta_x[(1-\theta_y)(1-\theta_x)(1-\theta_y) + \theta_y(1-\theta_x)\theta_y]\right\},$$

was wir leicht als verschwindend identifizieren. □

6.4 Parallelisierung

Bei einer Verfeinerung der Ortsdiskretisierung zugleich in x- und y-Richtung verhält sich der Rechenaufwand wie $\mathcal{O}\left(h^{-3}\right)$. Zwei Potenzen von h^{-1} werden durch die erhöhte Zahl von Gitterpunkten hervorgerufen, während die dritte aus der erhöhten Steifheit des Systems resultiert. Zwar erhöht sich die Norm der Jacobi-Matrix der rechten Seite der Differentialgleichung auch um den Faktor h^{-2}, aber RKC kann dies durch eine um den Faktor h^{-1} höhere Stufenzahl ausgleichen, da die Größe des Stabilitätsgebietes quadratisch von der Stufenzahl abhängt. Für sehr feine Gitter ist der Einsatz eines Parallelrechners daher unumgänglich.

Mit der Vorgabe der Ortsdiskretisierung und der Zeitintegration ist der gesamte numerische Algorithmus beschrieben. Die aufwendigsten Teile für ein feines Gitter sind die Auswertung des Laplace-Operators und die lineare Algebra in RKC. Beide sind prinzipiell unabhängig voneinander parallelisierbar, zur Vermeidung von zusätzlichem Kommunikationsaufwand sollten jedoch beide Teile im Zusammenhang parallelisiert werden. Wir beschreiben im folgenden die von uns gewählte Vorgehensweise auf Teras, einer SGI Origin 3800.

6.4.1 Die Teras-Architektur

Teras besitzt 1024 Prozessoren, jeder mit einer theoretischen Maximalleistung von 1 Gflop/s. Diese sind in zwei Clustern mit je 512 Prozessoren angeordnet, die Cluster sind als “Hypercubes” organisiert. Kleine Einheiten von je 4 Prozessoren sind über ein HUB mit einem gemeinsamen Hauptspeicher verbunden. Nutzt man mehr als 4 Prozessoren, so präsentiert sich das System zwar immer noch als “shared memory” Maschine, physikalisch ist sie jedoch “distributed memory” Modell.

Die Architektur wird unter der Bezeichnung “Cache coherent, non uniform memory access” (CC-NUMA) geführt. Dies steht für einen gemischten Ansatz aus einem physikalischen “distributed memory” Modell (was zum nicht gleichmäßigen Speicherzugriff – NUMA – führt), und einem logischen “shared memory”-Modell, was sich im Begriff “Cache coherent” niederschlägt. Für weitere Details verweisen wir auf [88]. Es wird sich zeigen, daß die physikalischen Eigenschaften des Systems bei der Implementierung einer parallelen Applikation nicht ignoriert werden können.

6.4.2 Parametrisierung

Ein wichtiger Aspekt bei der Parallelisierung eines Codes stellt die Analyse der parallelisierungs-relevanten Problem- und Maschinenparameter dar – zusammengefaßt unter dem Namen Parametrisierung. Dies erlaubt Vorhersagen über die Skalierbarkeit des Problems, was insbesondere für zukünftige 3D-Implementierungen von Bedeutung ist. Auf der anderen Seite kann mit Hilfe einer Parametrisierung für eine spezielle Klasse von Anwendungen eine maßgeschneiderte Computerkonfiguration ermittelt werden.

Wir folgen der Herangehensweise in [41] und diskutieren hier zwei wichtige Parameter: das problemabhängige (**a**pplication) γ_a und das **m**aschinenabhängige γ_m. Je kleiner der Koeffizient γ_m/γ_a, umso besser paßt die Anwendung zur Maschine. Dies bedeutet, daß die durch Kommunikation verursachte Minderung der “Peak-Performance” möglichst gering ist.

Der Parameter γ_a ist als Quotient aus der Anzahl von Operationen (in Mflops), die auf einem einzelnen Prozessor auszuführen sind, und der Menge von Daten (in Mwords), die zu diesem Zwecke von Prozessor zu Prozessor zu transportieren sind, definiert. Dieser Parameter hängt von der konkreten Anwendung ab. Bei der von uns gewählten Raum- und Zeitdiskretisierung kann dieser Parameter in

erster Näherung mit

$$\gamma_a = \frac{25.5N}{p} \tag{6.31}$$

angegeben werden, wobei N die Anzahl der Gitterpunkte in jeder der beiden Raumdimensionen bezeichnet, und p die Anzahl von Prozessoren. Dabei haben wir nur den Aufwand durch den Laplace-Operator auf den drei Gittern berücksichtigt, nicht aber die Berechnung der Quellterme und der Gradienten. Da diese Operationen für N_A Axonen und N_t Zielzellen bei feinen Gittern eine um Größenordnungen kleinere Rechenleistung erfordern, haben wir sie vernachlässigt in (6.31).

Der Parameter γ_m beschreibt im wesentlichen, wieviele Operationen auf einem Prozessor in der Zeit ausgeführt werden können, die notwendig ist, ein Wort an Daten dorthin zu transportieren. Eine genaue Definition von γ_m, sowie weitere Details zur Parametrisierung, können in [41] gefunden werden.

Die Teras verfügt über eine CC-NUMA-Architektur, daher ist γ_m keine Konstante. Der Kommunikationsaufwand hängt von der relativen Lage der sendenden und empfangenden (wenn man denn dieses Bild verwenden möchte) Prozessoren ab. So fällt kein Kommunikationsaufwand an, wenn die Prozessoren sich in einem 4er-Cluster mit gemeinsamem HUB befinden. Für andere Paarungen hängt der Kommunikationsaufwand von der Betragssummennorm des Abstands im Hypercube ab. Im typischen Multi-User-Betrieb ist dieser Abstand a priori nicht bekannt, einzig bei Anforderung von 4 Prozessoren bekommt man für gewöhnlich solche 4, die über ein HUB auf einen gemeinsamen Hauptspeicher zugreifen. Auch bei Anforderung von 2^n Prozessoren scheint man einen kompletten Hypercube zugewiesen zu bekommen. Dies ist zumindest unsere Erfahrung nach mehr-monatiger Nutzung der Maschine. Im allgemeinen wird also γ_m auch für identische Testläufe variieren.

Im Prinzip kann γ_m als Quotient zwischen der effektiven Leistung des Prozessors und der Netzwerk-Übertragungsrate bestimmt werden. Für Teras erhalten wir so einen Wert von 20. Unter Berücksichtigung von eventuell ungünstigen Plazierungen im Hypercube erscheint 25 eine realistische Wahl. Wie wir in Abschnitt 6 sehen werden, ist der Effekt eines nicht-konstanten γ_m durchaus wahrnehmbar, hier speziell für die Gebietszerlegung auf dem gröbsten Gitter. Wir konnten hier für wiederholte Rechnungen wesentlich verschiedene Rechenzeiten messen. Der Kommunikationsaufwand war in keiner Weise vorhersagbar. Für alle anderen Rechnungen (also entweder auf feinerem Gitter oder mit anderen Parallelisierungen des Laplace-Operators) konnten keine größeren Abweichungen bei Wiederholungen festgestellt werden. Insbesondere scheint die Anforderung von genau 2^n Prozessoren einen positiven Effekt zu haben.

6.4.3 Das “shared memory”-Modell auf Teras

Wir haben den OpenMP-Standard zur Implementierung der parallelen Anwendung gewählt. Die Anweisungen zur Parallelisierung sind in Kommentaren versteckt, so daß der Code ebenso auf Ein-Prozessor-Rechnern läuft.

Diese Anweisungen beinhalten die parallele Abarbeitung einzelner Programmteile, die Synchronisierung der Prozessoren sowie die Spezifikation der Lage und Zugriffsrechte für Daten, siehe [77]

Zur parallelen Verarbeitung wird ein Fork-Join-Modell verwendet. Die Abarbeitung startet sequentiell, und sobald ein paralleler Abschnitt erreicht wird, werden weitere Prozessoren aktiviert, die die folgenden Anweisungen parallel abarbeiten. Die Verteilung der Aufgaben auf die Prozessoren erfolgt dabei durch drei mögliche Modelle:

- PARALLEL DO zur Parallelisierung von Schleifen. Zur Erhaltung der Load-Balance werden verschiedene Modi unterstützt:

  - STATIC: Für Schleifen mit konstanten Kosten pro Durchlauf. Die Prozessoren starten zur gleichen Zeit (d.h., alle Prozessoren sind verfügbar), die Zuweisung der Aufgaben erfolgt zu Beginn (statisch).
  - DYNAMIC: Für Schleifen mit variablen Kosten pro Prozessor.
  - GUIDED: Bei konstanten Kosten pro Prozessor, aber unsynchronisiertem Start (d.h., die Prozessoren beginnen zu unterschiedlichen, nicht vorhersagbaren Zeiten mit der Abarbeitung der Schleife).

- PARALLEL SECTIONS – unterschiedlicher Code wird auf unterschiedlichen Prozessoren abgearbeitet – sogenannte multiple code parallelization.
- PARALLEL – gleicher Code wird auf unterschiedlichen Prozessoren abgearbeitet – sogenannte single code parallelization. Jeder Prozessor benötigt dabei natürlich unterschiedliche Initialisierungen. Dies wird durch die THREADPRIVATE-Klausel unterstützt, die private Kopien von einem Common-Block erzeugt.

Außerdem kann die Verteilung auf die Prozessoren auch innerhalb eines Unterprogramms geändert werden, welches separat übersetzt wurde (“orphane parallelism”).

Wir haben zur Parallelisierung der rechten Seite im wesentlichen Schleifen-Parallelisierung verwendet. Die von uns erstellte parallele Version von RKC nutzt den “single-code”-Ansatz und “orphane”-Parallelisierung.

Im Gegensatz zu MPI gibt es keine expliziten Kommunikationsanweisungen. Der Transfer von Daten erfolgt über SHARED-Variablen, die von jedem Prozessor gelesen und beschrieben werden können. Der prinzipiell andere Datentyp (es gibt noch weitere Mischformen) ist PRIVATE. Für Variablen dieses Typs legt jeder Prozessor seine eigene Kopie an. OpenMP unterstützt die parallele Auswertung von Summen, zu diesem Zweck gibt es die REDUCTION-Klausel.

Zur Synchronisierung der Prozessoren können die Klauseln BARRIER, CRITICAL und ATOMIC verwendet werden, wobei bei letzteren zwei Klauseln Vorsicht geboten ist: Sie können bei unsachgemäßem Einsatz zu wesentlich verringerter Leistung führen. Ein gewichtiger Vorteil von OpenMP besteht darin, daß ein lauffähiger serieller Code in einen parallelen Code in kleinen Zwischenschritten überführt werden kann, wobei der Code in den einzelnen Stufen der Portierungsphase lauffähig bleibt.

6.4.4 Die Funktionsauswertung der rechten Seite

Die rechte Seite setzt sich aus drei wesentlichen Bestandteilen zusammen:

- der diskretisierte Laplace-Operator
- die Interpolation der Gradienten
- die diskretisierte δ-Funktion.

Parallelisierung des Laplace-Operators

Wir approximieren Ableitungen zweiter Ordnung der drei Konzentrationen $\rho_j, j \in \{t, a, r\}$ in jedem Gitterpunkt $(x_k, y_l), \quad k, l = 1, \ldots, N-1$.

Bei der Parallelisierung ist jedem Prozessor eine Teilmenge der Menge aller Tripel

$$I := \{(j,k,l)\} = \{t,a,r\} \times \{1,\ldots,N-1\}^2 \tag{6.32}$$

zuzuordnen, wobei die einzelnen Teilmengen aus Effizienzgründen annähernd gleich groß sind. Bei der Parallelisierung von ineinander geschachtelten Schleifen ist es zumeist am effizientesten, nur die äußere Schleife (bzw. mehrere äußere Schleifen) zu parallelisieren ("Coarse grain parallelization"), und jeden Durchlauf der inneren Schleifen komplett auf Einzel-Prozessoren zu realisieren. Damit wird insbesondere beim hier genutzten "Fork-Join"-Modell die Anzahl der Verzweigungen und Synchronisationen minimiert. Enthält die äußere Schleife weniger Durchläufe als Prozessoren, so nimmt man weitere Schleifenebenen zur Parallelisierung hinzu. Die der Parallelisierung unterworfenen äußeren Schleifen wollen wir als Master-Schleifen bezeichnen.

Die Master-Schleifen werden im Prinzip als eine Schleife interpretiert, und die s Durchläufe dieser Schleife werden in zusammenhängenden Blöcken auf p Prozessoren aufgeteilt: Die Durchläufe $0, \ldots, \lfloor s/p \rfloor - 1$ werden dem Prozessor mit Nummer 0 zugewiesen, die Durchläufe $\lfloor s/p \rfloor, \ldots, \lfloor 2s/p \rfloor - 1$ dem Prozessor mit Nummer 1, und so fort. Wir setzen dabei voraus, daß die Kosten der einzelnen Durchläufe gleich groß sind.

Die einzigen Entscheidungen bei der Parallelisierung sind somit die Reihenfolge der Schleifen und die Wahl der Master-Schleifen. Wir haben 3 Zugänge gewählt, die in Abbildung 6.2 illustriert werden. Beim Element-Ansatz laufen die Master-Schleifen über die Chemikalien (ganz außen) und über

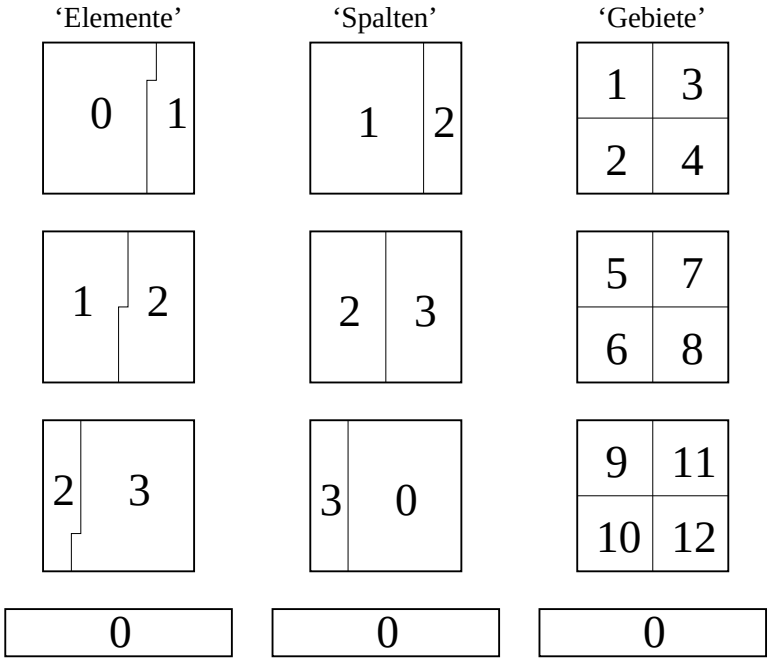


Abbildung 6.2: Verschiedene Parallelisierungs-Strategien für die Gitterpunkte

die Gitterpunkte. Soweit möglich, wird also auf jedem Prozessor die gleiche Zahl von Gitterpunkten ausgewertet. OpenMP parallelisiert genau so im Modus STATIC, wenn die Gitterpunkte für alle drei Chemikalien in einem zusammenhängenden Vektor (in passender Reihenfolge) abgelegt werden und eine Schleife zur Auswertung des Laplace-Operators über alle Einträge des Vektors läuft.

Beim Spalten-Ansatz laufen die Master-Schleifen über die Chemikalien (ganz außen) und über die Spalten des Gitters. Ein weiterer Unterschied besteht noch zum Element-Ansatz - wir haben dem Prozessor mit Nummer 0 (Master-Prozessor), der auch die Axonendaten speichert, den letzten Satz von Gitterpunkten zugeordnet.

Der dritte Ansatz steht im Geist des Gebiets-Splitting. Die quadratischen Gebiete werden in Teil-

quadrate zerlegt. Außerdem wird der Master-Prozessor nicht zur Auswertung des Laplace-Operators eingesetzt. Somit benötigen wir zur Aufteilung des Gebietes in M^2 Teilquadrate genau $3M^2 + 1$ Prozessoren.

Parallelisierung der Gradienten-Interpolation

Eine äußere Schleife über die Axonen ist hier die Master-Schleife. Jeder Prozessor arbeitet auf einer Teilmenge der Axonen, wobei im Falle $p > N_A$ einige Teilmengen leer sein können.

Hier wie auch im vergangenen Abschnitt gilt, daß verschiedene Prozessoren disjunkte Komponenten der Lösung eigenständig ausrechnen, d.h., es finden keine Schreib-Operationen in gemeinsame Speicherbereiche statt (z.B. die gemeinsame Aufdatierung einer Variablen), und es benötigt auch kein Prozessor Zwischenergebnisse eines anderen Prozessors. Der Zugriff auf sogenannte "SHARED"-Variablen geschieht nur im Rahmen von Lese-Operationen. Die Parallelisierung einer Schleife ist bei diesen Voraussetzungen eine Sache des Einfügens von wenigen OpenMP-Direktiven, der Code selbst bleibt unverändert sequentiell lauffähig.

Im umgekehrten Fall spricht man von "Race-Conditions": Mehrere Prozessoren schreiben an die gleiche Stelle. Dabei kann einfach nur die Reihenfolge der Operationen bedeutsam sein, was gewissen Synchronisationsaufwand erfordert. Typisch für "Race-Conditions" ist jedoch folgendes Szenario: Der Ausdruck $f(g(x))$ ist durch Prozessor F (wertet f aus) und Prozessor G (wertet g aus) zu berechnen. Nun liest Prozessor G den Wert x und beginnt mit der Auswertung von g. Wenn nun Prozessor F den Wert x liest, bevor Prozessor G das Ergebnis $g(x)$ zurückgeschrieben hat, so schreibt Prozessor F nicht $f(g(x))$ zurück, sondern überschreibt das (inzwischen von G berechnete) Zwischenergebnis $g(x)$ mit $f(x)$. Es findet im Prinzip ein "Wettrennen" zwischen der Leseaufforderung von F und dem Schreiben von G statt. Offensichtlich führt dies selbst dann zu Problemen, wenn die Operationen f, g vertauschbar sind. Race-Conditions erfordern zusätzlichen Synchronisationsaufwand und sind nach Möglichkeit bereits auf algorithmischer Ebene zu vermeiden.

Parallelisierung der Quellterme

Auch in den Quelltermen findet man einige Ansatzpunkte zur Parallelisierung. Allerdings stellt sich heraus, daß die meisten Ansätze zu Race-Conditions führen.

Ein natürlicher Zugang ist, die Schleife über Axonen und Zielzellen zur Master-Schleife zu erklären. Für die Zielzellen führt dies auch zu keinen Komplikationen, für die von uns betrachteten Gitterweiten liegen die Zielzellen in verschiedenen Gitterzellen, und selbst im entgegengesetzten Fall ist eine Synchronisation aufgrund der festen Lage einfach. Aber die Lage der Axonen ist im voraus nicht bekannt, hier können mehrere Axonen sehr nahe beieinander und somit in der gleichen Gitterzelle liegen. Sie liefern damit Beiträge zu den gleichen Gitterpunkten – und dies resultiert in Race-Conditions.

OpenMP bietet eine einfache Möglichkeit, die notwendige Synchronisation durchzuführen – die Klauseln ATOMIC und CRITICAL. Sie bewirken, daß eine Anweisung bzw. eine Sequenz von Anweisungen nur von einem Prozessor gleichzeitig ausgeführt werden kann. Man fordert eben dies für die Schreiboperationen.

Beim Ansatz mit Exponentialfunktionen für die Quellterme erscheint eine Parallelisierung durchaus lohnenswert, denn insbesondere die Berechnung der Exponentialfunktionen ist sehr kostenintensiv. Mit 40 Axonen und einem 8×8-Gitter als Träger des Quellterms sind 2560 Gitterpunkte aufzudatieren. Die ATOMIC/CRITICAL-Klauseln bewirken, daß der erste Prozessor, der die kritische Anweisung erreicht, ein Flag setzt ("LOCK"), das jeder andere Prozessor vor Ausführung testet. Nur

wenn das Flag nicht gesetzt ist, darf die Anweisung ausgeführt werden. Dies führt natürlich zu erhöhtem Kommunikationsaufwand. Unsere Erfahrungen mit diesem Konzept sind durchweg negativ, die parallele Version war zum Teil langsamer als die sequentielle. Wir empfehlen die Verwendung der Klauseln ATOMIC/CRITICAL nur unter Vorbehalt.

Wir wollen noch kurz einige kompliziertere Ansätze kommentieren. Ein Zugang nutzt Parallelisierung über die Gitterpunkte für bis zu 64 Prozessoren. Wir berechnen die Beiträge der Quellterme im 8×8-Träger mit 64 Prozessoren parallel. Nachteil ist zusätzlicher Kommunikationsaufwand, da der Beitrag nicht auf dem Prozessor berechnet wird, der auch die jeweiligen Konzentrationswerte speichert. Ein zweiter Zugang teilt die Gitterpunkte in 64 Untergitter modulo 8 auf und arbeitet mit bis zu N_A Prozessoren. Im Schritt k berechnen wir auf Prozessor l (dem Axon l zugeordnet) den Beitrag zum Gitterpunkt (i, j) mit

$$8i + j \equiv k + l \mod 64. \tag{6.33}$$

Damit werden Race-Conditions vermieden, allerdings muß nach jedem solchen Schritt synchronisiert werden, d.h., es muß auf die Vollzugsmeldung aller Prozessoren gewartet werden.

Sehr kurz ist die Diskussion zur Parallelisierung der Tensorprodukt-B-Splines – wir parallelisieren sie überhaupt nicht. Der rechnerische Aufwand ist, verglichen mit den anderen Teilen des Algorithmus, sehr gering. Unsere Versuche zeigten keinerlei Speedup bei einer Verteilung der Aufgabe auf verschiedene Prozessoren.

6.4.5 Lineare Algebra in RKC

Ein Vorteil der Verwendung einer expliziten Methode gegenüber einer impliziten Methode besteht darin, daß nur Level-1-BLAS-Routinen benötigt werden. Diese sind zumeist problemlos zu parallelisieren. Der Code RKC benötigt 5 Vektoren der Größe n um interne Größen zu speichern, wobei n die Dimension des Problems sei. Mit diesen Vektoren werden die folgenden Operationen ausgeführt

$$u = v, \tag{6.34}$$

$$u = v + \alpha w, \tag{6.35}$$

$$u = \alpha u + \beta v + \gamma_1 w_1 + \gamma_2 w_2 + \gamma_3 w_3, \tag{6.36}$$

$$\alpha = \|v - w\|_u = \max_i \frac{|v_i - w_i|}{a + r * |u_i|}. \tag{6.37}$$

Bei den ersten drei Ausdrücken werden einfach die Schleifen parallelisiert. Für den vierten Ausdruck nutzen wir die REDUCTION-Klausel. Diese sorgt bei der parallelen Auswertung von Summen, Produkten oder Maxima (Voraussetzung ist die Assoziativität der Operation) für das “Einsammeln” der Teilergebnisse von den einzelnen Prozessoren.

Der Kommunikationsaufwand bei den Vektoroperationen wird minimiert, wenn die Daten auf jedem der 5 Vektoren in gleicher Weise den verschiedenen Prozessoren zugeordnet werden. Wir haben dies durch folgende Implementierung gesichert: Jeder Prozessor verfügt über einen mit THREADPRIVATE privatisierten Common-Block, in dem Information über die von ihm zu bearbeitenden Komponenten der Vektoren abgelegt wird. Alle 5 Arbeitsvektoren von RKC und der Vektor der Startwerte (der von RKC ebenfalls genutzt wird) sind so in gleicher Weise unterteilt. Da die Zuordnung der Variablen zu den Prozessoren bei der ersten Initialisierung erfolgt, werden zu Beginn alle Arbeitsvektoren einmal parallel mit der gleichen Prozedur initialisiert, um die korrekte Zuordnung zu fixieren.

Die Auslastung der Prozessoren bei den Vektoroperationen ist sicher optimal, wenn jeder Prozessor die gleiche Anzahl von Komponenten verarbeitet. Auf der anderen Seite müssen wir auch bei der

Berechnung der rechten Seite auf gleichmäßige Verteilung der Arbeit achten. Nach unseren Annahmen wird die i-te Komponente der Lösung U auf dem gleichen Prozessor gespeichert wie die i-te Komponente der rechten Seite $F(U)$ (vergleiche (6.16)). Zur Berechnung des Laplace-Operators ist dies ideal. Bei der Berechnung der Gradienten und der Quellterme ist die Situation allerdings anders. Die Gradienten hängen von den Konzentrationen der Chemikalien ab und treten in der rechten Seite der Gradientengleichungen auf. Die Quellterme wiederum hängen von der Lage der Axonen ab und gehen in die rechte Seite der Diffusionsgleichungen ein, wobei die beeinflußten Gitterpunkte eben mit der Lage des Axons variieren. Würden wir die Zuordnung zu einem Prozessor von der Lage des Axons abhängig machen, so könnte unter Umständen (wenn alle Axonen dicht beieinander liegen, zum Beispiel beim Bündeln gewisse Zeit nach dem Start) ein Prozessor die gesamte Arbeit aufgebürdet bekommen.

Da für unsere Konfiguration jedoch der Rechenaufwand für die Gradienten und Quellterme einen relativ geringen Anteil am Gesamtaufwand darstellt, und eben wegen oben geschilderter Komplikationen, haben wir uns entschieden, die Quellterme nicht zu parallelisieren. Für die Gradienten wählen wir einfache Schleifen-Parallelisierung ohne Optimierung des Kommunikationsaufwandes.

Bei Nutzung der von uns erstellten parallelen Version von RKC hat man die Entscheidung über die Parallelisierung der rechten Seite und der Vektoroperationen zu treffen. Die rechte Seite ist dabei unter kompletter Kontrolle durch den Anwender, während zur Parallelisierung der Vektoroperationen das Modul ‘parallel’ exportiert wird. Dort müssen im wesentlichen eine Initialisierungs-Prozedur und zwei Schleifen geschrieben werden. Eine Schleife führt die Vektoroperationen aus, die für jede einzelne Komponente separat ausgewertet werden können (zum Beispiel “daxpy” BLAS1). Die zweite Schleife dient der parallelen Berechnung von Vektornormen unter Nutzung der REDUCTION-Klausel. Die Aufteilung der Komponenten auf die Prozessoren ist dabei in beliebiger Komplexität möglich, die notwendigen Daten werden im THREADPRIVATE-Common-Block abgelegt. Für den einfachen Fall, daß jeweils gleich große zusammenhängende Blöcke auf die Prozessoren verteilt werden, haben wir das zu erstellende Code-Segment in Abbildung 6.4 aufgeführt.

```
MODULE parallel
      common/pdata/iproc,nprocs,ind0,ind1
      !$OMP THREADPRIVATE(/pdata/)
contains
      subroutine initproc()
          use problem
          nprocs=OMP_GET_NUM_THREADS()
          iproc=OMP_GET_THREAD_NUM()
          call getrange(iproc,nprocs,nvar,ind0,ind1)
          ...
```

Abbildung 6.3: Typische Init-Routine im Modul “parallel”.

Die entsprechende Init-Routine ist in Abbildung 6.3 angegeben. Sie wird im parallelen Modus von allen Prozessoren im Hauptprogramm aufgerufen. Für jeden einzelnen Prozessor wird die parallele Umgebung analysiert, die Gesamtzahl der Prozessoren sowie die individuelle Nummer im Kollektiv bestimmt, und dann die Start- und Endpunkte ind0, ind1 für den zugeordneten Abschnitt im Lösungsvektor bestimmt. Diese individuellen Daten werden im Common-Block /pdata/ gespeichert,

```
subroutine genlinalg(task,y1,y2,a1,a2,a3,a4,a5)
    !$OMP PARALLEL PRIVATE (i)
    do i=ind0,ind1
        call genlinalg_i(task,i,y1,y2,a1,a2,a3,a4,a5)
    end do
    !$OMP END PARALLEL
    ...

subroutine redlinalg(task,err,y1,a1,a2,i1)
    !$OMP PARALLEL REDUCTION(MAX:err) PRIVATE(i)
    do i = ind0,ind1
        call redlinalg_i(task,i,err,y1,a1,a2,i1)
    end do
    !$OMP END PARALLEL
    ...
```

Abbildung 6.4: Beispiel-Code aus dem Modul “parallel”

der für alle Programmteile über das Modul parallel verfügbar ist. Dort kann jeder Prozessor zusätzliche private Daten für die Berechnung der rechten Seite ablegen. Sind die Daten in komplizierterer Weise als oben beschrieben auf die Prozessoren verteilt, so müssen die Schleifen in Abbildung 6.4 modifiziert werden, sowie die notwendigen privaten Daten (Inkremente für Schleifen, weitere Start- und Endpunkte, etc.) in /pdata/ abgelegt werden. Weitere Eingriffe in den Code sind nicht notwendig, alle Modifikationen beschränken sich auf die Parallelisierung der rechten Seite und die Änderungen im Modul “parallel”.

6.5 Simulationsergebnisse

In diesem Abschnitt stellen wir die Ergebnisse der Simulation dar. Unser Test-Beispiel sieht wie folgt aus:

Die partielle Differentialgleichung wird auf einem quadratischen Gebiet der Größe $1mm \times 1mm$ gelöst. Dieses Gebiet wird mit 4 verschiedenen Gittern der Auflösung $N = 64, 128, 256, 512$ diskretisiert, wobei N die Anzahl der Teilintervalle in jeder Ortskoordinate ist. Da unser Interesse auch in der Untersuchung des Einflusses der Gittergröße auf die numerische Lösbarkeit als auch auf die Parallelisierbarkeit liegt, haben wir für alle Gittergrößen Berechnungen durchgeführt.

Das Test-Modell ist durch die Gleichungen (6.1), (6.2), (6.3), (6.4) und (6.5) definiert, wobei die Parameter folgende Werte besitzen:

$$D_t = D_a = D_r = 10^{-4};\ \ \kappa_t = \kappa_a = \kappa_r = 10^{-4};$$

$$\lambda_t = 10^{-5},\ \lambda_a = 5 \cdot 10^{-6},\ \lambda_r = 3.75 \cdot 10^{-5};$$

$$\sigma_t = 6 \cdot 10^{-5},\ \sigma_a = 7.5 \cdot 10^{-5},\ \sigma_r = \frac{7.5 \cdot 10^{-5} \cdot r}{2.42 + r},\ r = 1 - e^{-\rho_t}.$$

Fast alle Parameter sind in Zeit und Ort konstant, nur σ_r ist sowohl von der Zeit als auch vom Ort abhängig.

In unserer Versuchsanordnung starten 40 Axonen ($N_A = 40$) in Richtung von 50 Zielzellen ($N_t = 50$). Dabei haben wir den Ausstoß an anziehender Substanz ρ_t durch die Zielzellen gestoppt, sobald die Zielzelle von einem Axon erreicht wurde. Dies bestimmt auch die zeitliche Skale unserer Simulation: Sobald alle Axonen ein Ziel gefunden haben, wird die Integration beendet.

In Abbildung 6.5 zeigen wir die Lösung auf dem feinsten Gitter (512×512 Gitterpunkte). Man erkennt das typische Verhalten der Axonen: Bündelung, Wachstum zu den Zielzellen, Auseinanderdriften, und letztendlich Verbindung mit den Zielzellen. Die Lösungen auf gröberen Gittern sind von qualitativ gleicher Gestalt. In Abbildung 6.5 sind die Positionen der Axonen zu drei Zeitpunkten in t=1200, 5600, und 7880 markiert (gemessen in Sekunden). Zur Zeit $t = 1200$ hat sich das Axonenbündel fast formiert. Die Axonen erhöhen nun den Ausstoß an abstoßender Chemikalie, womit ein Gleichgewicht zwischen abstoßenden und anziehenden Chemikalien entsteht. Kurz nach dem zweiten Zeitpunkt, $t = 5600$, dominiert der Ausstoß an abstoßender Chemikalie und die Axonen driften auseinander. Zum letzten Zeitpunkt, $t = 7880$, haben fast alle Axonen eine Zielzelle erreicht. Man beachte, daß sich zunächst zu viele Axonen in Richtung des Zentrums bewegt haben, wo nicht genug Zielzellen für alle Axonen vorhanden waren. Überschüssige Axonen laufen daraufhin in die äußeren Regionen, wo noch genügend freie Zielzellen zum Andocken bereitstehen.

In den Abbildungen 6.6, 6.7, 6.8 zeigen wir für die drei Zeitpunkte die zugehörigen Konzentrationen der drei Chemikalien, sowie die Resultierende $\rho_{resultant} := \lambda_t\rho_t + \lambda_a\rho_a - \lambda_r\rho_r$. Der Gradient dieser Linearkombination definiert die rechte Seite der Gradientengleichung für die Axonen.

In allen drei Abbildungen sind Extrema in der Konzentration $\rho_{resultant}$ in der Nähe der Lage der Axonen auszumachen. Die Extrema wechseln von Maxima bei $t = 1200$ zu Minima bei $t = 5600$. Die Ursache ist das Dominieren der abstoßenden Chemikalie bei $t = 5600$. Man erkennt auch, daß die von den Zielzellen ausgestoßene Chemikalie wesentlich die Bewegung in Richtung der Zielzellen steuert, während die von den Axonen ausgestoßenen Chemikalien als Korrektiv für das Bündeln und Auseinanderdriften wirken.

Zum Zeitpunkt $t = 7880$ (Abbildung 6.8), wenn fast alle Axonen an einer Zielzelle angedockt haben, fällt dann auch die Konzentration ρ_t wesentlich, da die verbundenen Zielzellen die Produktion einstellen. Nur wenige Zielzellen am Rand sind aktiv.

Die Lösung des entstehenden ODE-Systems stellt sich als sehr sensitiv gegenüber kleinen Änderungen der Parameter und der Anfangswerte heraus. Dies gilt insbesondere für den Pfad, dem die Axonen folgen. Kleine Änderungen in den Positionen, insbesondere im Moment des Auseinanderdriftens, haben wesentliche Änderungen im weiteren Pfad oder sogar in der erreichten Zielzelle zur Folge. Auch eine Änderung des Gitters kann solche Effekte nach sich ziehen.

Um solche Effekte auszuschließen, ist die Verwendung eines feinen Gitters notwendig. Wir bemerken, daß eine Halbierung der Gitterweite den Rechenaufwand um den Faktor 8 steigert. Dies wird neben der erhöhten Zahl von Gitterpunkten außerdem durch die erhöhte Steifheit des Systems hervorgerufen.

6.6 Parallele Effizienz

Wir haben eine große Anzahl von Simulationsläufen für verschiedene Gitterweiten, Laplace-Parallelisierungen (Element, Spalten oder Gebiet, siehe Abbildung 6.2) und Prozessorzahlen (von 1 bis maximal 128) durchgeführt. Für jede Kombination haben wir die absolute Rechenzeit (in Sekunden) für den Gesamt-Algorithmus, den Laplace-Operator, die Gradienten-Berechnung und die lineare Al-

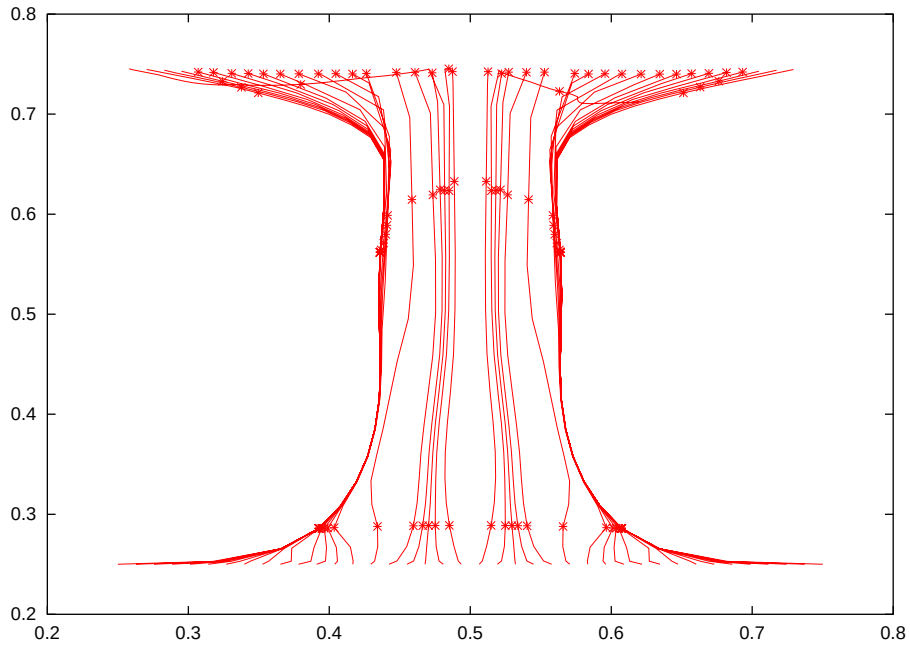


Abbildung 6.5: Der Pfad der Axonen und ihre Position in $t = 1200, 5600, 7880$. Simulation auf dem feinsten (512 x 512) Gitter. Längeneinheit: mm.

gebra in RKC gemessen. Daraus errechnen sich Speedup (Quotient aus gemessener Rechenzeit und der Rechenzeit bei sequentieller Ausführung) und die parallele Effizienz (Quotient aus Speedup und Zahl der Prozessoren). Diese Daten sind in den Diagrammen in den Abbildungen 6.9, 6.10, 6.11 und 6.12 graphisch aufbereitet.

Rekapitulieren wir noch einmal die charakteristischen Parameter von Problem und Maschine (siehe Abschnitt 6.4.2): Der Quotient γ_a/γ_m berechnet sich näherungsweise zu N/p. Zumindest für die feineren Gitter ($N = 256, N = 512$) können wir konstatieren, daß eine Effizienz von 100% für Werte $\gamma_a/\gamma_m \geq 8$ erreicht wird. Dies bestätigt die Resultate in [41], wo ein Wert von 4 oder höher empfohlen wurde. Unsere Resultate liegen in der gleichen Größenordnung.

Von den Ergebnissen der Abbildungen 6.9, 6.10, 6.11 und 6.12 können folgende Folgerungen gezogen werden:

- Mit einer kleinen Anzahl parallel arbeitender Prozessoren (bis 8) ist es relativ einfach, effizient zu parallelisieren. Auf dem gröbsten Gitter (64×64) erreichen wir noch eine Effizienz von 30%, auf dem nächst feineren (128×128) eine Effizienz von 50%-60%, und auf den zwei feinsten Gittern können wir sogar überlinearen Speedup beobachten.

- Auf dem feinsten Gitter können bis zu 64 Prozessoren effektiv ausgenutzt werden, die Effizienz liegt bei 75 % oder höher.

- Beim Vergleich der 3 Parallelisierungs-Strategien für den Laplace-Operator zeigen sich der Element- und der Spalten-Ansatz effizienter als der Gebiets-Ansatz.

- Auf dem gröbsten Gitter zeigt der Gebiets-Ansatz ein irreguläres Verhalten: Mit $13 = 3 \cdot 2^2 + 1$ Prozessoren wird eine sehr geringe Effizienz erreicht. Außerdem variieren die gemessenen

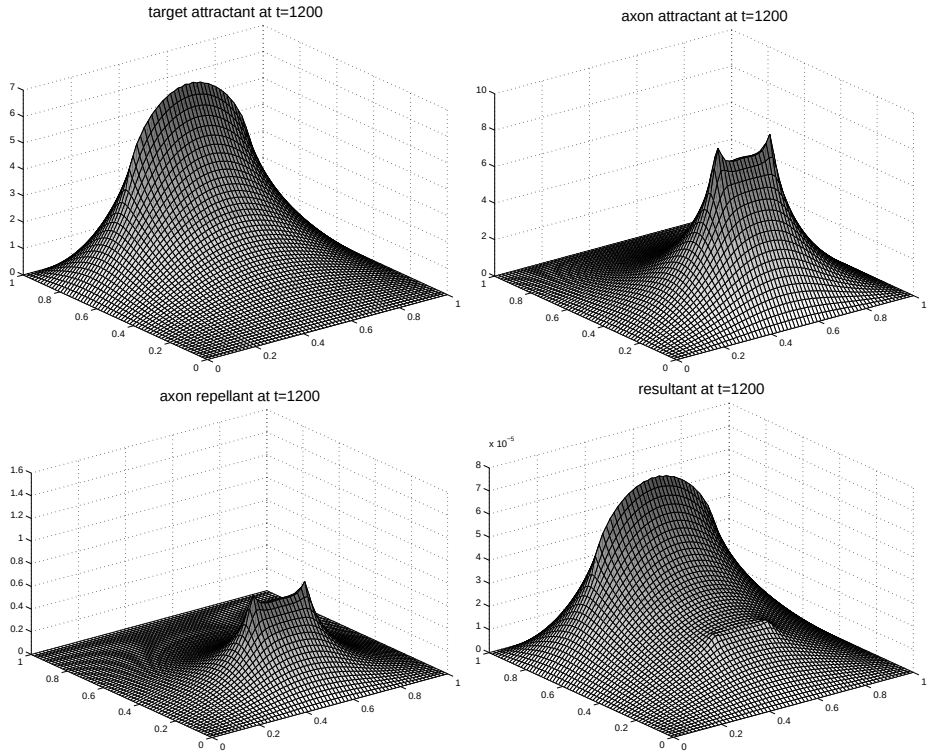


Abbildung 6.6: Die dimensionslosen Konzentrationen ρ_t, ρ_a, ρ_r und die Resultante $\rho_{resultant}$ kurz nach dem Start ($t = 1200$).

Laufzeiten erheblich. Eine mögliche Erklärung hierfür ist, daß bei Anforderung einer Prozessorenzahl, die keine Zweierpotenz ist, die zugewiesenen Prozessoren nicht auf dem kleinstmöglichen Hypercube liegen, sondern je nach Verfügbarkeit teilweise in weiter Distanz voneinander. Die CC-NUMA Architektur der Teras und die zufällige Verteilung der Prozessoren könnten die Ursache für die zufälligen Größen der Laufzeiten sein.

Eine weitere Besonderheit bei dem Gebiets-Ansatz ist, daß die Vektoren bei der Parallelisierung der linearen Algebra in RKC nicht in zusammenhängende Blöcke zerlegt werden. Wir bemerken außerdem, daß wir für höhere Prozessorenzahlen wie z.B. $28 = 3 \cdot 3^2 + 1$ keine randomisierten Laufzeiten beobachten konnten. Auch bei feineren Gittern und 13 Prozessoren traten diese Effekte nicht auf.

- Bei Zweierpotenzen $p = 2^n$ als Prozessorenzahl können die Laufzeiten problemlos reproduziert werden. Wir haben für alle Messungen mehrere Durchläufe vorgenommen, und haben dabei Differenzen von höchstens 10% feststellen können.

- Bei Betrachtung der verschiedenen Teilaufgaben im Algorithmus können wir die Vektoroperationen als am effizientesten parallelisierbar einschätzen. Auch die Parallelisierung des Laplace-Operators ist effizient möglich. Die Parallelisierbarkeit der Gradienten-Auswertung nimmt mit steigender Prozessorzahl deutlich ab.

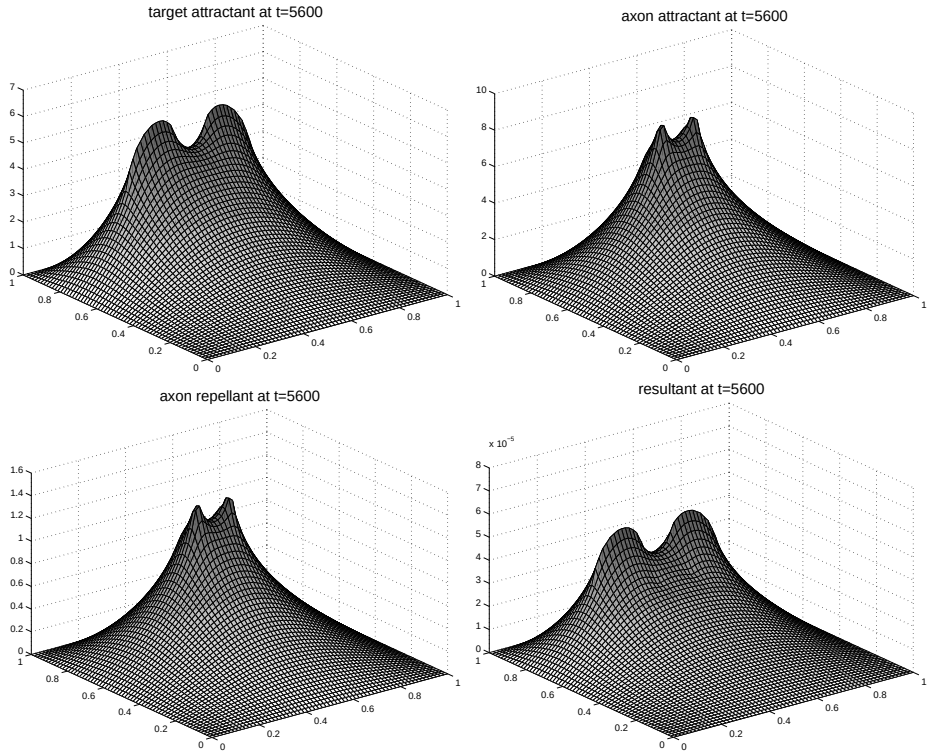


Abbildung 6.7: Die dimensionslosen Konzentrationen ρ_t, ρ_a, ρ_r und die Resultante $\rho_{resultant}$ beim Auseinanderdriften ($t = 5600$).

6.7 Zusammenfassung

Wir haben die parallele Implementierung einer Anwendung aus der Neurobiologie auf einer SGI Origin 3800 (Teras) erörtert. Das Modell besteht aus einem gekoppelten System aus drei parabolischen Gleichungen in zwei Raumdimensionen und gewöhnlichen Differentialgleichungen, die die Lage der Axonen in diesen zwei Dimensionen beschreiben.

Nach der Diskussion numerischer Techniken zur Diskretisierung in Zeit und Raum werden verschiedene Ansätze zur Parallelisierung vorgestellt. Der Gebiets-Ansatz stellt sich als am wenigsten effizient heraus, aussichtsreicher sind die Ansätze bei denen der Resultatsvektor in zusammenhängende Teile zerlegt wird.

Die Implementierung wurde für verschiedene Gittergrößen getestet: 64×64, 128×128, 256×256 und 512×512. Die effizienteste Parallelisierung konnte auf dem feinsten Gitter erzielt werden: eine Effizienz von 75% für 64 Prozessoren. Auf dem nächst gröberen Gitter konnte diese Effizienz immerhin noch für bis zu 16 Prozessoren erreicht werden. Auf dem gröbsten Gitter konnten immerhin noch bis zu 8 Prozessoren effektiv ausgenutzt werden. Für die eigentliche Simulation ist allerdings aus Gründen der Genauigkeit ohnehin das feinste Gitter zu verwenden, so daß man das Parallelisierungs-Experiment als erfolgreich abgeschlossen werten kann. Auf der Maschine Teras (Sommer 2003 auf Rang 285 in der “Top500-Supercomputing-Liste”) konnte mit moderatem Programmieraufwand ein Speedup von 50 mit 64 Prozessoren erzielt werden.

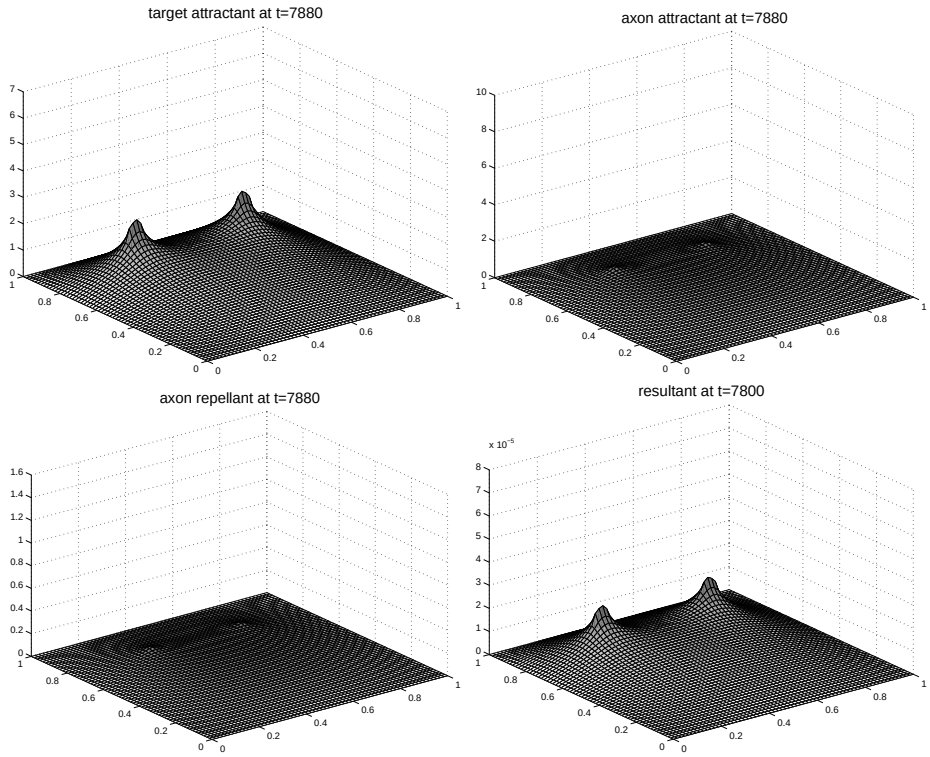


Abbildung 6.8: Die dimensionslosen Konzentrationen ρ_t, ρ_a, ρ_r und die Resultante $\rho_{resultant}$ zum Ende der Simulation ($t = 7880$).

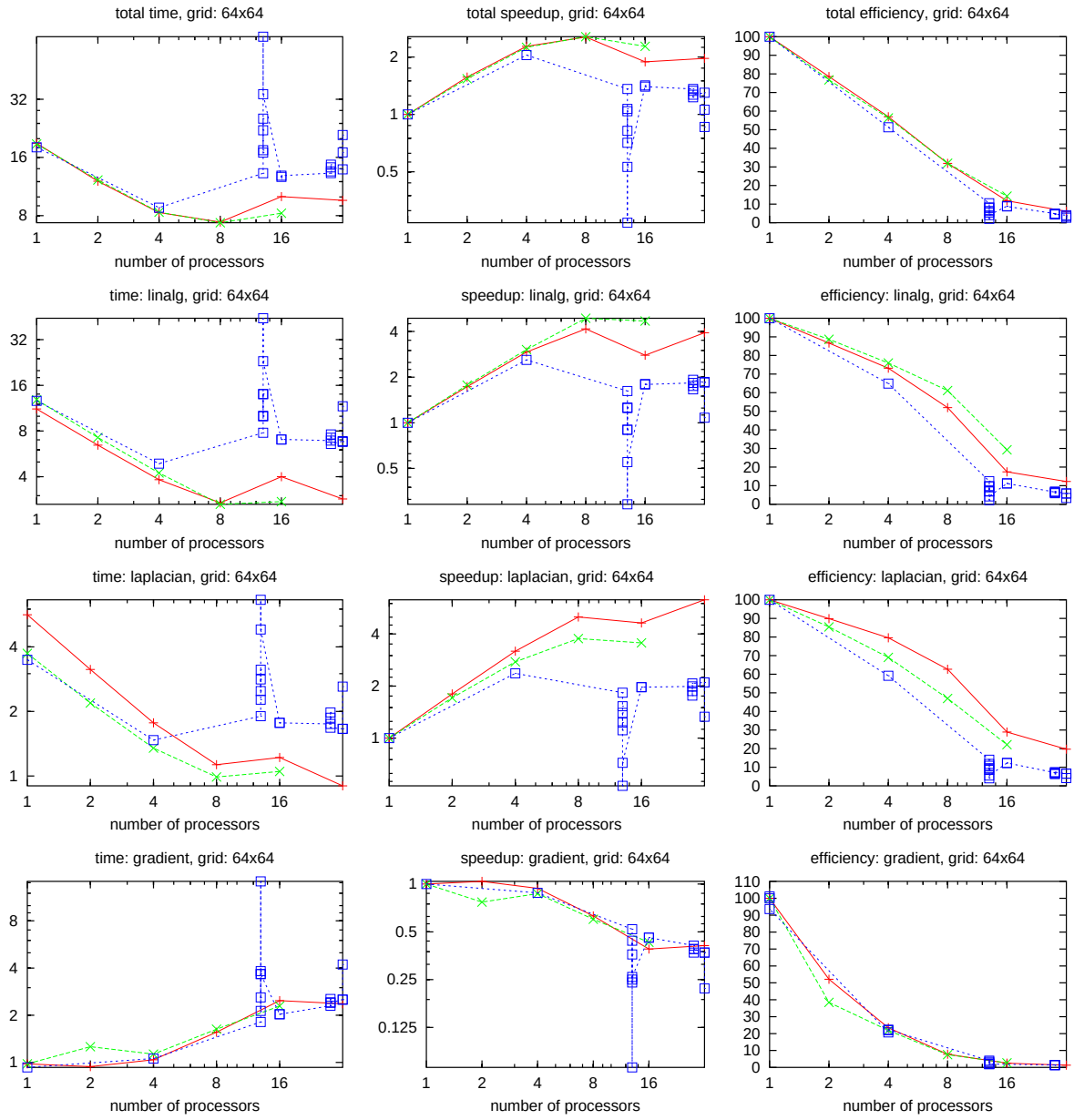


Abbildung 6.9: Resultate auf dem 64×64 Gitter für die 3 Ansätze (Abb. 6.2): ‘Element’ (+ + +), ‘Spalten’ ($\times \times \times$) und ‘Gebiet’ (□□□). Erste Zeile: Gesamtalgorithmus. Zweite Zeile: Vektoroperationen in RKC. Dritte Zeile: Laplace-Operator. Vierte Zeile: Gradienten. Erste Spalte: Rechenzeit in Sekunden. Zweite Spalte: Speedup. Dritte Spalte: Effizienz.

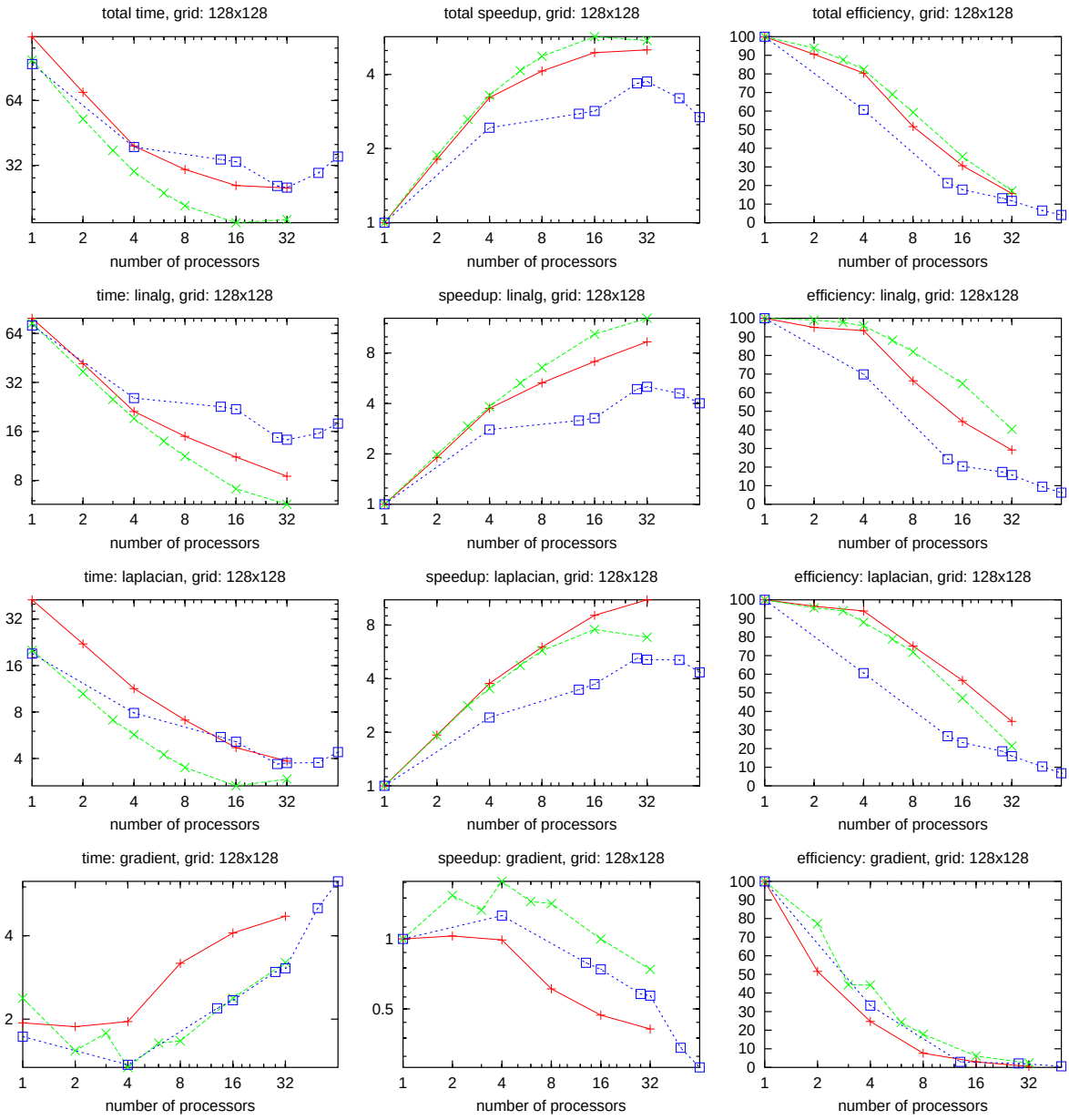


Abbildung 6.10: Resultate auf dem 128×128 Gitter für die 3 Ansätze (Abb. 6.2): 'Element' (+ + +), 'Spalten' ($\times \times \times$) und 'Gebiet' (□□□). Erste Zeile: Gesamtalgorithmus. Zweite Zeile: Vektoroperationen in RKC. Dritte Zeile: Laplace-Operator. Vierte Zeile: Gradienten. Erste Spalte: Rechenzeit in Sekunden. Zweite Spalte: Speedup. Dritte Spalte: Effizienz.

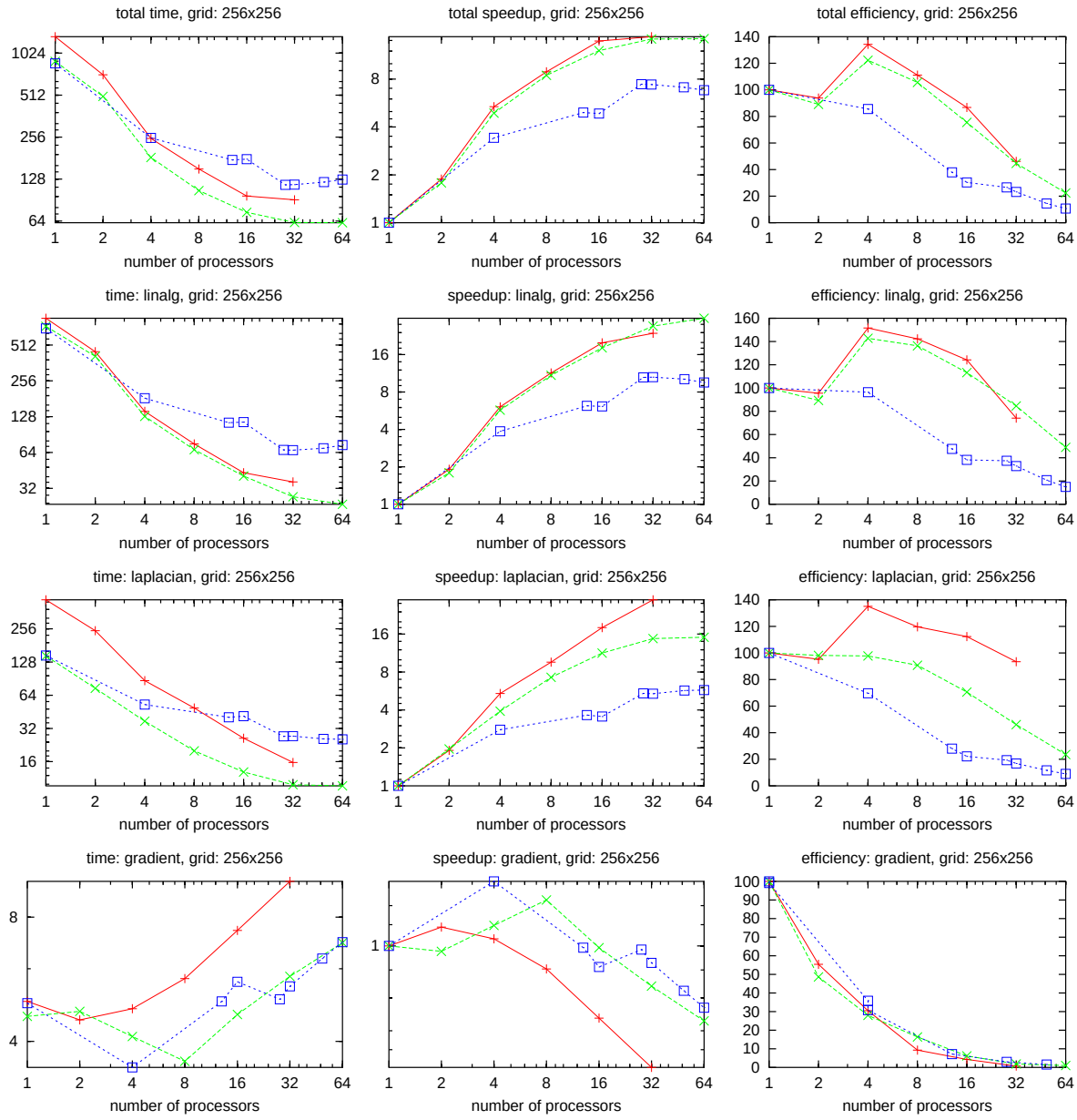


Abbildung 6.11: Resultate auf dem 256×256 Gitter für die 3 Ansätze (Abb. 6.2): 'Element' (+ + +), 'Spalten' ($\times \times \times$) und 'Gebiet' (□□□). Erste Zeile: Gesamtalgorithmus. Zweite Zeile: Vektoroperationen in RKC. Dritte Zeile: Laplace-Operator. Vierte Zeile: Gradienten. Erste Spalte: Rechenzeit in Sekunden. Zweite Spalte: Speedup. Dritte Spalte: Effizienz.

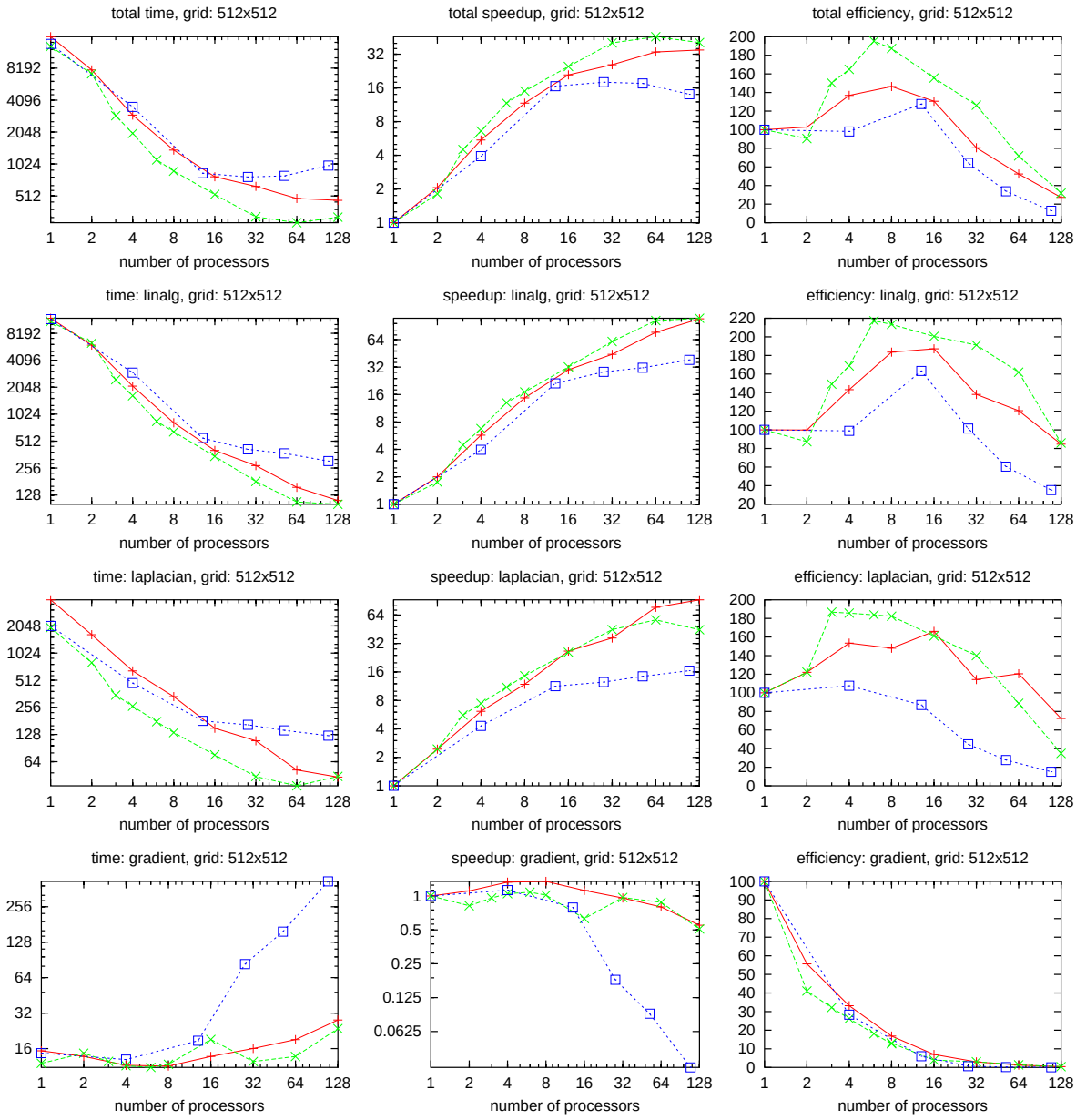


Abbildung 6.12: Resultate auf dem 512×512 Gitter für die 3 Ansätze (Abb. 6.2): 'Element' (+ + +), 'Spalten' ($\times \times \times$) und 'Gebiet' (□□□). Erste Zeile: Gesamtalgorithmus. Zweite Zeile: Vektoroperationen in RKC. Dritte Zeile: Laplace-Operator. Vierte Zeile: Gradienten. Erste Spalte: Rechenzeit in Sekunden. Zweite Spalte: Speedup. Dritte Spalte: Effizienz.

Symbol- und Abkürzungsverzeichnis

Lie-Gruppen

$GL(n)$	Gruppe der regulären Matrizen mit Dimension n
$\mathfrak{gl}(n)$	Lie-Algebra zu $GL(n)$
$O(n)$	Gruppe der orthogonalen Matrizen von Dimension n
$SO(n)$	Gruppe der orthogonalen Matrizen von Dimension n mit Determinante 1
$\mathfrak{so}$	Lie-Algebra der schiefsymmetrischen Matrizen von Dimension n
$\mathfrak{g}, \mathfrak{h}$	Lie-Algebren
$[\cdot, \cdot]$	Lie-Klammer
$[\cdot, \ldots, \cdot]$	iterierte Lie-Klammer
ad	Kommutator $\mathrm{ad}_A(B) := [A, B]$
$\exp$	Exponentialabbildung
$d\exp$	Differential der Exponentialabbildung
dexp	Die Abbildung dexp
cay	Cayley-Abbildung
$d\,\mathrm{cay}$	Differential der Abbildung cay
dcay	Die Abbildung dcay

Differenzierbare Mannigfaltigkeiten

$TM\|_x$	Tangentialraum an die Mannigfaltigkeit M im Punkt x
$v\|_y$	Vektorfeld v im Punkt y
$\Psi_v(\varepsilon, x)$	Fluß des Vektorfeldes v
dF	Differential der Abbildung F
$v(f)$	Richtungsableitung der Funktion f

Einschrittverfahren

$\Phi(t, y, h)$	Inkrementfunktion
le_h	lokaler Fehler bei Schrittweite h
e_h	globaler Fehler bei Schrittweite h
RKC	Runge-Kutta-Tschebyscheff
RK-MK	Runge-Kutta-Munthe-Kaas (Methoden)

Tensoren in der Mechanik

$A : B$	doppelt-skalares Produkt $\mathrm{spur}\, AB$
$\boldsymbol{\sigma}$	Spannungstensor
$\boldsymbol{\varepsilon}$	linearer Verzerrungstensor

Wurzelbäume

$[\,]$, $\{\,\}$	Basis-Baum, 1 Knoten (Wurzel)
$[\tau_1, \ldots, \tau_k]$	abgeleiteter Baum
$\{\tau_1, \ldots, \tau_k\}$	abgeleiteter Baum
ρ	Ordnung, Zahl der Knoten
F	zugeordnetes elementares Differential
α, γ	rekursiv definierte Funktionen
ϕ_i	Koeffizientenfunktion in Ordnungsbedingung

Quantenmechanik

h, $\hbar$	Plancksches Wirkungsquantum h, $\hbar = h/(2\pi)$
Ψ	Wellenfunktion
$\mathbf{r}$	Ortsvektor
r	Betrag des Ortsvektors, radiale Komponente
e	Elektronenladung
E, ϵ	Energie
$\hat{H}$	Hamilton-Operator

Verschiedenes

Re	Realteil einer komplexen Zahl
Im	Imaginärteil einer komplexen Zahl
$\mathbb{1}$	$(1, \ldots, 1)^T$
$\mathcal{O}(\,)$	Landau-Symbol
spec	Spektrum eines linearen Operators
δ_{ij}	Kronecker-Delta
B_k	Bernoulli-Zahlen
S_n	Symmetrische Gruppe (Permutationsgruppe)
n_x, n_y, ...	Dimension von x, y, ...
$\cong$	isomorph
CC-NUMA	Cache-coherent, non-uniform memory access

Literaturverzeichnis

[1] M. Abramowitz und I. Stegun. *Handbook of mathematical functions.* New York (1972).

[2] R. Alexander. *Diagonally implicit Runge-Kutta methods for stiff o.d.e.'s*. SIAM J. Numer. Anal., Vol. 14, 1006–1021 (1977).

[3] M. Arnold, K. Strehmel, R. Weiner. *Half-explicit Runge-Kutta methods for semi-explicit differential- algebraic equations of index 1*. Numer. Math. 64, No.4, 409-431 (1993).

[4] V. I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer-Verlag (1989).

[5] H. F. Baker. *Alternants and continuous groups*. Proc. London Math. Soc. 3, 24–47 (1905).

[6] D. Braess. *Finite Elemente.* Springer-Verlag, Berlin (1997).

[7] P.N. Brown, A.C. Hindmarsh und L.R. Petzold. *Using Krylov methods in the solution of large-scale differential- algebraic systems.* SIAM J. Sci. Comput. 15, No.6, 1467-1488 (1994).

[8] J. Bruder, K. Strehmel und R. Weiner. *Partitioned adaptive Runge-Kutta methods for the solution of nonstiff and stiff systems*. Numer. Math. 52, no. 6, 621–638 (1988).

[9] J. Büttner und B. Simeon. Runge-Kutta methods in elastoplasticity. Appl. Numer. Math. 41, No.4, 443-458 (2002).

[10] R. Burlisch und J. Stoer. *Numerical treatment of ordinary differential equations by extrapolation methods.* Numer. Math. 8, 1-13 (1966).

[11] K. Burrage. *Parallel and sequential methods for ordinary differential equations.* Oxford, Clarendon Press (1995).

[12] J. Butcher. *Coefficients for the study of Runge-Kutta integration processes.* J. Austr. Math. Soc. 3, 185–201 (1963).

[13] J. Butcher. *The numerical analysis of ordinary differential equations : Runge-Kutta and general linear methods*. Wiley (1987).

[14] M. P. Calvo, A. Iserles und A. Zanna. *Runge-Kutta methods for orthogonal and isospectral flows*. Appl. Numer. Math. 22, 153–163 (1996).

[15] M. P. Calvo, A. Iserles und A. Zanna. *Numerical solution of isospectral flows*. Math. Comp., 66(220), 1461–1486 (1997).

[16] J. E. Campbell. *On a law of combination of operators bearing on the theory of continuous transformation groups*. Proc. London Math. Soc., 28, 381.-390 (1897).

[17] S. Blanes, F. Casas, J. Ros. *Symplectic integration with processing: A general study.* SIAM J. Sci. Comput. 21, No.2, 711–727 (1999).

[18] S. Blanes, F. Casas und J. Ros. *Improved high order integrators based on the Magnus expansion*. BIT 40, 434-450 (2000).

[19] S. Blanes, F. Casas und J. Ros, High order optimized geometric integrators for linear differential equations, *preprint GIPS 2001-004* (2001).

[20] Cash, J.R. *Diagonally implicit Runge-Kutta formulae with error estimates.* J. Inst. Math. Appl. 24, 293-301 (1979).

[21] E. Celledoni und A. Iserles. *Methods for the approximation of the matrix exponential in a Lie-algebraic setting*. 21(2), 463-488 (2001).

[22] Crouch, P.E.; Grossman, R. *Numerical integration of ordinary differential equations on manifolds.* J. Nonlinear Sci. 3, No.1, 1–33 (1993).

[23] J.C. Dallon, *Numerical aspects of discrete and continuum hybrid models in cell biology*, APNUM 32, 137–159 (2000).

[24] K. Decker und J. Verwer. *Stability of Runge-Kutta methods for stiff nonlinear differential equations.* North Holland (1984).

[25] J.W. Demmel. *Applied Numerical Linear Algebra*, SIAM, Philadelphia (1997).

[26] F. Diele, L. Lopez und R. Peluso. *The Cayley transform in the numerical solution of unitary differential systems.* Adv. Comput. Math., 8(4), 317–334 (1998).

[27] F. Diele, L. Lopez, und T. Politi. *One step semi-explicit methods based on the Cayley transform for solving isospectral flows.* J. Comput. Appl. Math, 89, 219–223 (1998).

[28] J.R. Dormand und P.J. Prince. *New Runge-Kutta algorithms for numerical simulation in astronomy.* Celestial Mechanics 18, 223-232 (1878).

[29] P. Ellsiepen. *Zeit- und ortsadaptive Verfahren angewandt auf Mehrphasenprobleme poröser Medien.* Dissertation, Institut für Mechanik II, Universität Stuttgart (1999).

[30] P. Ellsiepen und S. Hartmann. *Remarks on the interpretation of current non-linear finite-element-analysis as differential-algebraic equations.* International Journal for Numerical Methods in Engineering, 51:679-707 (2001).

[31] K. Engo und A. Marthinsen. *Modeling and solution of some mechanical problems on Lie groups*. Multibody System Dynamics, 2, 71–88 (1998).

[32] K. Engo. *On the construction of geometric integrators in the RKMK class*. BIT, 40(1), 41–61 (2000).

[33] K. Engo. *On the BCH-formula in* $\mathfrak{so}(3)$. BIT 41, No.3, 629-632 (2001).

[34] Fa‘a di Bruno. *Note sur une novelle formule de calcul differentiel*. Quart. J. Math. 1 (1857).

[35] S. Faltinsen, A. Marthinsen und H. Munthe-Kaas. *Multistep methods integrating ordinary differential equations on manifolds*. Appl. Numer. Math. 39, no. 3-4, 349–365 (2001).

[36] K. Feng. *Difference schemes for Hamiltonian formalism and symplectic geometry.* J. Comp. Math. 4, 279-289 (1986).

[37] R. Feynman. *Vorlesungen über Physik. III. Quantenmechanik.* Oldenburg-Verlag (1988).

[38] H. Flaschka. *The Toda-Lattice. II. Existence of Integrals*. Phys. Rev. B9, 1924-1925 (1974).

[39] P. Fritze. *Numerische Behandlung nichtlinearer Probleme der Elastizitäts- und Plastizitätstheorie.* Dissertation, Fachbereich Mathematik, Technische Universität Darmstadt (1997).

[40] W. B. Gragg. *On extrapolation algorithms for ordinary initial value problems.* SIAM J. Numer. Anal., 2, 384–403 (1965).

[41] R. Gruber, P. Volgers, A. De Vito, M. Stengel und R.-M. Tran, *Parameterisation to tailor commodity clusters to applications.* J. Future Generation Computer Systems, 19, 111–120 (2003).

[42] Hairer, E. *Highest possible order of algebraically stable diagonally implicit Runge-Kutta methods.* BIT, 20, 254-256 (1980).

[43] E. Hairer, C. Lubich und M. Roche. *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*. Springer-Verlag (1989).

[44] E. Hairer, S. P. Nørsett und G. Wanner. *Solving ordinary differential equations. I. Nonstiff problems*. Springer-Verlag, Berlin (1993).

[45] E. Hairer. *Backward analysis of numerical integrators and symplectic methods.* Ann. Numer. Math. 1, No.1-4, 107–132 (1994).

[46] E. Hairer und G. Wanner. *Solving ordinary differential equations II. Stiff and differential-algebraic problems.* Springer-Verlag, Berlin (1996).

[47] E. Hairer, C. Lubich und G. Wanner. *Geometric numerical integration.* Springer-Verlag, Berlin (2002).

[48] S. Hartmann. *Computation in finite strain viscoelasticity: finite elements based on the interpretation as differential-algebraic equations.* Computer Methods in applied Mechanics and Engineering, 191(13-14), p. 1439-1470 (2002).

[49] S. Hartmann. *Finite-Elemente Berechnung inelastischer Kontinua.* Habilitationsschrift, Universität Kassel (2003).

[50] F. Hausdorff. *Die symbolische Exponentialformel in der Gruppentheorie*. Leipziger Ber., 58, 19–48 (1906).

[51] H.G.E. Hentschel und A. van Ooyen, *Models of axon guidance and bundling during development*. Proc. R. Soc. Lond. B. 266, 2231–2238 (1999).

[52] M. Hochbruck und Ch. Lubich. *On Krylov subspace approximations to the matrix exponential operator*. SIAM J. Numer. Anal., 34(5), 1911-1925 (1997).

[53] M. Hochbruck, C. Lubich und H. Selhofer. *Exponential integrators for large systems of differential equations.* SIAM J. Sci. Comput. 19, No.5, 1552–1574 (1998).

[54] M. Hochbruck und C. Lubich. *Error analysis of Krylov methods in a nutshell.* SIAM J. Sci. Comput. 19, no. 2, 695–701 (1998).

[55] T.J.R. Hughes. *The Finite Element Method.* Prentice-Hall (1987).

[56] A. Iserles, A. Marthinsen und S.P. Nørsett, *On the implementation of the method of Magnus series for linear differential equations.* BIT 39, no. 2, 281–304 (1999).

[57] A. Iserles und S. P. Nørsett. *On the solution of linear differential equations in Lie groups*. Phil. Trans. Royal Soc. A, 357:983-1020, 1999.

[58] A. Iserles, H. Munthe-Kaas, S.P. Nørsett, A. Zanna. *Lie-group methods*, Acta Numerica, 215-365 (2000).

[59] P. Kaps und P. Rentrop. *Generalized Runge-Kutta methods of order four with stepsize control for stiff ordinary differential equations*. Numer. Math., Vol. 33, 55-68 (1979).

[60] H.B. Keller. *Numerical Solution of Two Point Boundary Value Problems.* Society for Industrial and Applied Mathematics, Philadelphia (1976).

[61] A. Kiełbasiński und H. Schwetlick. *Numerische Lineare Algebra.* Berlin (1988).

[62] S. Klarsfeld und J.A. Oteo. *Recursive generation of high-order terms in the Magnus expansion*, Phys. Rev. A, 39 (1989).

[63] B. Lastdrager, *Numerical solution of mixed gradient-diffusion equations modelling axon growth*, CWI, Report MAS-R0203 (2002).

[64] S. Lie und F. Engel. *Theorie der Transformationsgruppen*. B. G. Teubner, Leipzig (1893).

[65] M. Hochbruck und C. Lubich. *On Magnus integrators for time-dependent Schrödinger equations*. Technical report, Universität Tübingen (2002).

[66] W. Magnus. *On the exponential solution of differential equations for a linear operator*. Comm. Pure and Appl. Math., VII, 649–673 (1954).

[67] J. E. Marsden und T. S. Ratiu. *Introduction to Mechanics and Symmetry*. Springer-Verlag (1994).

[68] A. Marthinsen, H. Munthe-Kaas und B. Owren. *Simulation of ordinary differential equations on manifolds — Some numerical experiments and verifications,* Applied Modelling and Simulation - Proceedings of the 38th SIMS Simulation Conference, Tapir trykkeri, Norway (1996).

[69] A. Marthinsen. *Interpolation in Lie groups*. SIAM J. Numer. Anal., 37(1), 269–285 (1999).

[70] A. Marthinsen und B. Owren. *Quadrature methods based on the Cayley transform*. Technical Report Numerics No. 1/1999, The Norwegian University of Science and Technology, Trondheim, Norway (1999).

[71] C. Moler, C.F. Van Loan. *Nineteen dubious ways to compute the exponential of a matrix.* SIAM Rev. 20, 801-836 (1978).

[72] H. Munthe-Kaas. *Lie-Butcher Theory for Runge-Kutta Methods.* BIT, 35, 572–587 (1995).

[73] H. Munthe-Kaas. *Runge-Kutta methods on Lie groups.* BIT, 38, 92–111 (1998).

[74] H. Munthe-Kaas. *High Order Runge-Kutta Methods on Manifolds.* Applied Numerical Mathematics, 29, 115–127 (1999).

[75] H. Munthe-Kaas und B. Owren. *Computations in a Free Lie Algebra.* Phil. Trans. Royal Soc. A. 357, 957–981 (1999).

[76] P.J. Olver. *Applications of Lie Groups to Differential Equations*, Springer GTM no. 107 (1986).

[77] OpenMP Architecture Review Board, *OpenMP Fortran Application Program Interface.* http://www.openmp.org (2001).

[78] B. Owren und B. Welfert. *The Newton iteration on Lie groups*. BIT, 40(1), 121–145 (2000).

[79] H. Podhaisky und R. Weiner. *A class of explicit two-step Runge-Kutta methods with enlarged stability regions for parallel computers.* Parallel computation. 4th international ACPC conference including special tracks on Parallel numerics (ParNum '99) and parallel computing in image processing, video processing, and multimedia. Salzburg, Austria, February 16–18, 1999. Proceedings. Berlin: Springer. Lect. Notes Comput. Sci. 1557, 68-77 (1999).

[80] H. Podhaisky, R. Weiner und J. Wensch. *High Order Explicit Two-Step Runge-Kutta Methods for Parallel Computers.* Journal of Computing and Information Technology - CIT 8, 1, 13-18 (2000).

[81] W. C. Rheinboldt. *Differential-algebraic systems as differential equations on manifolds*. Math. Comp., 43, 473–482, (1984).

[82] P. Rentrop, M. Roche und G. Steinebach. *The application of Rosenbrock-Wanner type methods with stepsize control in differential-algebraic equations*. Numer. Math., Vol. 55, 545-563 (1989).

[83] M. Roche. *Runge-Kutta and Rosenbrock methods for differential-algebraic equations*. Dissertation, Universität Genf (1988).

[84] M. Roche. *Rosenbrock methods for differential algebraic equations*. Numer. Math., Vol. 52, 45-63 (1988).

[85] H.H. Rosenbrock. *Some general implicit proxesses for the numerical solution of differential equations.* Comp. J. 5, 329-331 (1963).

[86] Y. Saad. *Iterative methods for Sparse linear systems.* PWS, Boston (1996).

[87] J.M.Sanz-Serna und M.P. Calvo. *Numerical Hamiltonian Problems*, Chapman & Hall, London (1994).

[88] *SGI Origin3000 Teras manual*, SARA, Academic Computer Centre Amsterdam (2001).

[89] O. Scherf und B. Simeon. *Viscoplastic deformation from the DAE perspective – a benchmark problem.* ZAMM, Z. Angew. Math. Mech. 79, Suppl. 1, S17-S20 (1999).

[90] O. Scherf. *Numerische Simulation inelastischer Körper.* Dissertation, Fortschritt-Berichte VDI, Reihe 20, Nr. 321. VDI-Verlag, Düsseldorf (2000).

[91] B.A. Schmitt und R. Weiner. *Matrix-free W-methods using a multiple Arnoldi iteration*. Appl. Numer. Math., 18, 307–320 (1995).

[92] B.P. Sommeijer, L.F. Shampine und J.G. Verwer, *RKC: An explicit solver for parabolic PDEs*. J. Comput. Appl. Math. 88, 315-326 (1997).

[93] T. Steihaug und A. Wolfbrand. *An attempt to avoid exact Jacobian and nonlinear equations in the numerical solution of stiff ordinary differential equations.* Math. Comp. 33, 521-534 (1979).

[94] H.J. Stetter. *Symmetric two-step algorithms for ordinary differential equations*, Computing, Vol. 5 (1970).

[95] K. Strehmel und R. Weiner, *Behandlung steifer Anfangswertprobleme gewöhnlicher Differentialgleichungen mit adaptiven Runge-Kutta-Methoden*. Computing 29, no. 2, 153–165 (1982).

[96] K. Strehmel und R. Weiner, *B-Convergence results for linearly implicit one step methods.* BIT, vol. 27, 264-281 (1987).

[97] K. Strehmel und R. Weiner, *Linearly-implicit Runge-Kutta methods for singularly perturbed problems and index-1-DAEs*. Numerical treatment of differential equations (Halle 1989), Teubner-Texte Math., 121, Teubner, Stuttgart, 168–177 (1991).

[98] K. Strehmel und R. Weiner. *Linear-implizite Runge-Kutta-Methoden und ihre Anwendung*. B. G. Teubner, Stuttgart (1992).

[99] K. Strehmel und R. Weiner. *Numerik gewöhnlicher Differentialgleichungen*. B. G. Teubner, Stuttgart (1995).

[100] M. Toda. *Theory of Nonlinear Lattices*, Springer-Verlag, Berlin (1978).

[101] L.N. Trefethen und D. Bau. *Numerical linear algebra*. Philadelphia (1997).

[102] I. Turek, V. Drchal, J. Kudrowský und M. Ŝob. *Electronic structure of disordered alloys, surfaces and interfaces.* Boston, Kluwer (1997).

[103] P.J. van der Houwen und B.P. Sommeijer. CWI contributions to the development of parallel Runge-Kutta methods. Appl. Numer. Math. 22, No.1-3, 327-344 (1996).

[104] H.A. van der Vorst. *An iterative solution method for solving $f(A)x = b$, using Krylov subspace information obtained for the symmetric positive definite matrix A*. J. Comput. Appl. Math., 18, 249–263 (1987).

[105] V.S. Varadarajan. *Lie Groups, Lie Algebras and their Representations*, Springer-Verlag (1984).

[106] J.G. Verwer und B.P. Sommeijer. *A numerical study of mixed parabolic-gradient systems*. J. Comput. Appl. Math. 132, 191-210 (2001).

[107] G. Wanner. *On the integration of stiff differential equations.* Numerical Analysis (J. Descloux und J. Marti, eds.), ISNM, Vol. 37, 209-226, Birkhäuser, Basel (1977).

[108] F. W. Warner. *Foundations of Differentiable Manifolds and Lie Groups*. Springer-Verlag (1983).

[109] R. Weiner, M. Arnold, P. Rentrop und K. Strehmel. *Partitioning strategies in Runge-Kutta type methods.* IMA Joutnal of Numerical Analysis 13, 303-319 (1993).

[110] R. Weiner, B. A. Schmitt und H. Podhaisky, *ROWMAP – a ROW-code with Krylov techniques for large stiff ODEs.* Appl. Numer. Math. 25, No.2-3, 303-319 (1997).

[111] J. Wensch, R. Weiner und K. Strehmel. *Stability Investigations for Index-2 Systems.* Report 94-01, Reports on Numerical Mathematics, Martin-Luther-Universität Halle-Wittenberg (1994).

[112] J. Wensch, K. Strehmel und R.Weiner. *A class of linearly-implicit Runge-Kutta methods for multibody systems*. Appl. Numer. Math. 22, no. 1-3, 381–398 (1996).

[113] Wensch, J. *Partitioned W-Methods for Descriptor Systems.* Tagungsband, 3. Workshop über Deskriptorsysteme, 357-364, Paderborn (1996).

[114] J. Wensch. *Zur numerischen Integration differential-algebraischer Systeme - partitionierte ROW-Methoden für Mehrkörpersysteme und Stabilität von Diskretisierungsverfahren für Index-2-Systeme.* Dissertation, Martin-Luther-Universität Halle-Wittenberg, 1997. Tectum-Verlag, Marburg (1997).

[115] J. Wensch. *An eight stage fourth order partitioned Rosenbrock method for multibody systems in index*-3 *formulation*. Appl. Numer. Math. 27, no. 2, 171–183 (1998).

[116] J. Wensch. *Extrapolation methods in Lie groups*. Numer. Math. 89, no. 3, 591–604 (2001).

[117] J. Wensch und F. Casas. *Extrapolation in Lie groups with approximated BCH-formula*. Appl. Numer. Math. 42, no. 1-3, 465–472 (2002).

[118] J. Wensch und B.P. Sommeijer. *Parallel simulation of axon growth in the nervous system.* CWI report MAS-R0208 (2002).

[119] J. Wensch. *Das Paket GLIEALG. In: Mathematica Information Center, GradedFreeLieAlgebra.* http://library.wolfram.com/infocenter/MathSource/4955 (September 2003).

[120] J. Wensch und B.P. Sommeijer. *Parallel simulation of axon growth in the nervous system.* Angenommen zur Publication in Parallel Computing (2003).

[121] J. Wensch, H. Podhaisky und S. Hartmann. *Time integration of index 1 DAEs with Rosenbrock methods using Krylov subspace techniques.* Eingereicht bei Proceedings in Applied Mathematics and Mechanics (2003).

[122] J. Wensch, M. Däne, W. Hergert und A. Ernst. *The solution of stationary problems in quantum mechanics by Magnus methods with stepsize control.* Report 03-09, Reports on Numerical Mathematics, Martin-Luther-Universität Halle-Wittenberg (2003).

[123] S. Wolfram. *Mathematica: a system for doing mathematics by computer.* Addison-Wesley (1991).

[124] H. Yoshida. *Construction of higher order symplectic integrators.* Phys. Lett. A150, 262-268 (1990).

[125] A. Zanna. *Collocation and Relaxed Collocation for the Fer and the Magnus Expansion.* SIAM J. Numer. Anal., Vol. 36, No. 4, 1145-1182 (1999).

Erklärung

Hiermit erkläre ich an Eides statt, die Habilitationsschrift selbständig und ohne fremde Hilfe verfaßt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht zu haben.

Jörg Wensch

Lebenslauf

Persönliche Daten

Name:	Jörg Wensch.
Adresse:	Philipp-Müller-Str. 11, 06110 Halle, BRD.
Geburtsdatum:	29.9.1968 in Köthen/Anhalt.
Familienstand:	verheiratet, 2 Kinder.

Schulausbildung / Wehrdienst

1975-1985:	Besuch der Polytechnischen Oberschule in Köthen
1985-1987:	Abitur, Spezialklassen für Mathematik und Physik, Martin-Luther-Universität Halle-Wittenberg.
1987-1988:	Wehrdienst.

Wissenschaftlicher Werdegang

Sep. 1988 - Sep. 1992:	Studium der Mathematik an der Martin-Luther-Universität Halle-Wittenberg.
Sep. 1992:	Diplommathematiker, MLU Halle.
Okt. 1992 - Sep. 1997:	Wissenschaftlicher Mitarbeiter, MLU Halle, FB Mathematik und Informatik, Institut für Numerische Mathematik
April 1997:	Dr. rer. nat., MLU Halle, „summa cum laude“.
Okt. 1997 - Sep. 2003:	Wissenschaftlicher Assistent, MLU Halle, FB Mathematik und Informatik, Institut für Numerische Mathematik
Ab Okt. 2003:	Wissenschaftlicher Mitarbeiter, MLU Halle, FB Mathematik und Informatik, Institut für Informatik