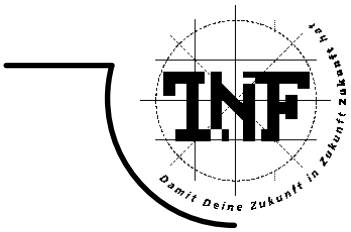
JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis



Numerische Lösung von Algebro-Differentialgleichungen mit Anwendung in der Simulation von Kühlsystemen

MASTERARBEIT

zur Erlangung des akademischen Grades

DIPLOMINGENIEURIN

in der Studienrichtung

INDUSTRIEMATHEMATIK

Angefertigt am *Institut für Numerische Mathematik*

Betreuung:

A.Univ.-Prof. Dipl.-Ing. Dr. Walter Zulehner

Eingereicht von:

Marion Lackner

Mitbetreuung:

Dipl.-Ing. Christian Rathberger

Linz, Jänner 2007

Johannes Kepler Universität
A-4040 Linz · Altenbergerstraße 69 · Internet: http://www.uni-linz.ac.at · DVR 0093696

Zusammenfassung

Diese Diplomarbeit ergab sich aus einer Problemstellung bei der Simulation von Kühlsystemen durch das Softwarepacket KULI. Die der transienten Simulation zugrundeliegende Modellierung führt auf eine semi-explizite Algebro-Differentialgleichung vom Index 1, deren effiziente numerische Lösung Ziel der Arbeit ist.
Aufgrund der Reduzierbarkeit semi-expliziter Algebro-Differentialgleichungen vom Index 1 auf explizite gewöhnliche Differentialgleichungen können die üblichen numerische Verfahren zur Lösung gewöhnlicher Differentialgleichungen herangezogen werden. In der Diplomarbeit werden zwei Klassen von numerischen Verfahren behandelt, die expliziten Runge-Kutta-Verfahren und die linearen Mehrschrittverfahren, speziell die Adams-Verfahren.
Neben der Beschreibung und Analyse der Verfahren und ihrer Anwendung auf semi-explizite Algebro-Differentialgleichungen vom Index 1, wird ebenso auf die praktische Umsetzung der Verfahren, insbesondere auf die Schrittweitensteuerung und auf die Datenausgabe, eingegangen.
Für die Durchführung der numerischen Experimente wurden bereits implementierte Verfahren in KULI integriert und mit der ursprünglichen Berechnungsvorschrift verglichen. Für ein repräsentatives Modell konnte eine Beschleunigung um mindestens den Faktor zehn erreicht werden.

Abstract

This thesis is based on a problem that arised in the simulation of cooling systems with the software package KULI. The underlying model of the transient simulation is described by a system of semi-explicit differential-algebraic equations of index 1. The aim of the work is the efficient numerical solution of this system.
This system of semi-explicit differential-algebraic equations of index 1 can be reduced to a system of ordinary differential equations, hence for solving them, it is possible to use conventional numerical methods. In the thesis two classes of numerical methods are considered, explicit Runge-Kutta methods and linear multistep methods, especially Adams methods.
In addition to the description and analysis of the numerical methods and their application to semi-explicit differential-algebraic equations of index 1, also the implementation of the methods is considered, in particular step size control and dense output.
The numerical experiments were done by implementing already exisiting codes of the methods into KULI. For a representative model the comparison of the original algorithm and the new one shows a performance increase of at least a factor of ten.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben.

Vor allem bei Prof. Dr. Walter Zulehner für die engagierte und lehrreiche Betreuung. Seine Ideen und Vorschläge waren mir eine große Hilfe bei der Anfertigung dieser Arbeit.

Mein besonderer Dank gilt dem Engineering Center Steyr für die Bereitstellung des Themas sowie die finanzielle Unterstützung. Vielen Dank den Mitarbeitern der Abteilung Technische Berechnung, insbesondere Dipl.Ing. Christian Rathberger, für das angenehme Arbeitsklima und die Beantwortung vieler meiner Fragen.

Mein größter Dank gebührt meinen Eltern, die mich in allem, was ich tue, unterstützen und nie an mir zweifeln.

Zu guter Letzt möchte ich mich noch bei all jenen bedanken, die mich durch meine Studienjahre begleitet und mein Leben bereichert haben.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die Diplomarbeit entstand im Rahmen der Mitarbeit bei einem Industrieprojekt der Firma **Magna Powertrain** *Engineering Center Steyr*. Ziel dieses Projekts war die Verbesserung der transienten Rechnung des Sofwarepackets KULI, welches der Simulation von Kühlsystemen bei Fahrzeugen dient. Anhand einer graphischen Oberfläche lassen sich verschiedene Komponenten eines solchen Kühlsystems zu Kreisläufen zusammenschließen und die Zustände Temperatur, Massenstrom und Druck simulieren. KULI wurde ursprünglich zur stationären Berechnung von Kühlsystemen ausgelegt, doch aufgrund der Nachfrage nach transienter Simulation wurde das Programm erweitert. Bei dem Projekt sollten die dabei aufgetretenen Probleme in der transienten Rechnung, wie beispielsweise Schrittweitenabhängigkeit der Lösung, möglichst ohne großen Mehraufwand in der Implementierung und Modellierung beseitigt werden. Durch mathematische Analyse der transienten Rechnung wurde festgestellt, dass das zu lösende Gleichungssystem einem semi-expliziten Algebro-Differentialgleichungssytem (DAEs) vom Index 1 entspricht. Solche DAEs lassen sich auf ein System expliziter gewöhnlicher Differentialgleichungen (ODEs) reduzieren. Dadurch ist es möglich semi-explizite DAEs vom Index 1 mit den bekannten numerischen Methoden für explizite ODEs zu lösen.

In der Diplomarbeit wird auf zwei Klassen von Verfahren zur Lösung von semi-expliziten DAEs vom Index 1 eingegangen, den *Runge-Kutta-Verfahren* und den *linearen Mehrschrittverfahren*, genauer gesagt den *Adams-Verfahren*. Diese beiden Verfahrensklassen wurden ausgwählt, da Runge-Kutta-Verfahren die üblichsten Verfahren zur Lösung gewöhnlicher Differentialgleichungen darstellen und ein 8-stufiges Runge-Kutta-Verfahren der Ordnung 5(6) aus der *imsl*-Bibliothek zur Verfügung stand. Zur Berechnung der nächsten Näherung be-

nötigen Runge-Kutta-Verfahren jedoch die Auswertung der Funktion an Zwischenschritten. Bei semi-expliziten DAEs vom Index 1 bedeutet dies indirekt die Auswertung des algebraischen Teils und in unserem Fall eine zeitintensive Berechnung. Mehrschrittverfahren hingegen benötigen keine Zwischenschritte zur Berechnung der nächsten Näherung und sind somit für diese Fälle möglicherweise von Vorteil.

1.2 Kapitelübersicht

In Kapitel 2 wird anhand eines einführenden Beispiels das Programm KULI beschrieben. Es wird sowohl die stationäre wie auch die transiente Modellierung grob erläutert und die daraus resultierende Problemstellung mathematisch formuliert.

Kapitel 3 und 4 dienen der Beschreibung und Analyse der beiden Klassen von numerischen Methoden. (Dabei wird stets auf die Erweiterung der Themen auf semi-explizite Algebro-Differentialgleichungen vom Index 1 eingegangen.)

Bei der praktischen Durchführung der beiden Klassen von numerischen Methoden waren zwei Themen von besonderem Interesse, jenes der Schrittweitensteuerung und jenes der kontinuierlichen Datenausgabe (engl. *dense output*), welche in Kapitel 5 behandelt werden. Aufgrund der Schrittweitensteuerung wählt das Verfahren die Gitterpunkte, an denen die Funktion ausgewertet wird, selbst, die kontinuierliche Datenausgabe bietet, mit nur geringem Mehraufwand im Laufe der Berechnung, die Möglichkeit, die Näherungslösung auch zwischen den Gitterpunkten zu erhalten (z.B. zur graphischen Ausgabe) .

Direkt in KULI implementiert und getestet wurden drei Runge-Kutta-Verfahren (`imsl_d_ode_runge_kutta`, MATLAB-Funktion `ode45` und `DOPRI5`) und ein Mehrschrittverfahren (MATLAB-Funktion `ode113`). Die Ergebnisse und Beschreibung der numerischen Experimente und die daraus resultierenden Schlussfolgerungen sind in Kapitel 6 erläutert.

Kapitel 2

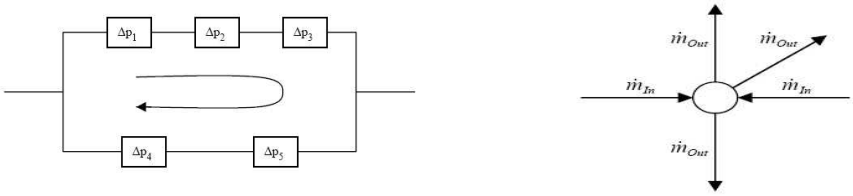
Einführendes Beispiel und Problemstellung

In diesem Kapitel wird die Modellierung und Simulation in KULI anhand eines konkreten Beispiels grob erläutert. Dabei wird insbesondere auf die transient modellierten Komponenten, den sogenannten Punktmassen, eingegangen. Die sich aus der Modellierung ergebende Problemstellung ist eine semi-explizite Algebro-Differentialgleichung, die in Abschnitt 2.3 diskutiert wird.

2.1 Das Programm KULI

KULI ist ein Software-Programm zur Simulation von Kühlsystemen in Fahrzeugen. Solch ein Kühlsystem ist einerseits durch Wärmeerzeuger, wie Motor und Wandler, und andererseits durch Komponenten, wie Kühler und Lüfter bestimmt. Über eine graphische Oberfläche lassen sich einzelne Komponenten (z.B. Kühler, Rohre, Lüfter, usw.) zu verschiedenen Kreisläufen vernetzen. Ziel der Berechnung ist, die Temperaturen der an der Kühlung beteiligten Medien im gesamten System zu ermitteln. Durch Vorgabe der Simulationsparameter (Motordrehzahl, effektiver Mitteldruck, Fahrgeschwindigkeit, Umgebungsdruck, -temperatur, Luftfeuchte, usw.) beschreibt man einen Betriebspunkt des Fahrzeugs. Massenstrom, Druck und Eintrittstemperatur bzw. zugeführte Wärme in einem Kreislauf können als fixe Werte oder abhängig von den Simulationsparametern, z.B. in Form von Kennlinien, vorgegeben werden.

Jeder Kreislauf ist als (gerichteter) Graph modelliert, in dem jede Komponente einer Kante entspricht, die mit beliebig vielen anderen Komponenten über Knoten verknüpft werden kann. Die Zustandsgrößen des Modells sind *Massenstrom, Temperatur* und *Druck*. Jedem Knoten werden dabei Temperatur und



(a) $\sum \Delta p_i = 0$ (b) $\sum \dot{m}_{In} = \sum \dot{m}_{Out}$

Abbildung 2.1: Maschenregel und Knotenregel

Druck zugewiesen und jeder Kante ein Massenstrom.

Die Zustandsgleichungen, erhält man einerseits über die Komponenten, in denen vorgegebene Funktionen oder Kennlinien einen Druckverlust und Wärmeübergang beschreiben, und andererseits aus Maschenregeln, Knotenregeln und der Wärmebilanz. Die Maschenregel ist in Abb. 2.1(a) dargestellt und besagt, dass die Summe der Druckverluste in einer Masche gleich Null ist. Als Masche bezeichnet man einen geschlossenenen Weg in einem Netzwerk. Aus der Massenerhaltung folgt die Knotenregel (Abb. 2.1(b)): die Summe der eingehenden Massenströme ist gleich jener der ausströmenden Massenströme. Die Wärmebilanz ergibt $\sum \dot{Q} = 0$ für einen Kreislauf, es wird gleich viel Wärme zu- wie abgeführt.

Daraus ergibt sich ein Gleichungssystem, das man nach Temperatur und Druck am Ein- und Ausgang jeder Komponente und nach den dazugehörigen Massenströmen auflösen kann. Die Lösung dieses Gleichungssystems ist Ziel der stationären Rechnung in Kuli und wird als *stationärer Abgleich* des Kühlsystems bezeichnet. Beim stationären Abgleich wird das Gleichungssystem mithilfe eines Iterationsverfahrens gelöst.

Die Modellierung der einzelnen Komponenten in Kuli beruht auf der stationären Lösung des Systems, daher sind Zustandsgrößen und -gleichungen zeitunabhängig modelliert. Ein interessanter Aspekt bei der Simulation von Kühlsystemen ist allerdings auch, wie sich die Kreisläufe, bzw. ihre Temperatur, im Laufe der Zeit und der damit verbundenen Veränderung der Simulationsparameter des Fahrzeugs verhalten, z.B. bei einer Bergauffahrt oder einem Aufheizvorgang. Um Dynamik im System zu erreichen, wurden daher neue Komponenten, die sogenannten *Punktmassen*, entworfen. Eingefügt in einem Kreislauf dienen Punktmassen der Simulation der thermischen Trägheit des jeweiligen im Kreislauf befindlichen Mediums.

Eine Punktmasse steht in zwei Versionen zur Verfügung: entweder als *reguläre Punktmasse*, die Teil eines Kreislaufs ist, oder als *isolierte Punktmasse*, die mit anderen Punktmassen durch Wärmeleitung verbunden werden kann. Das Verhalten der Punktmasse lässt sich durch Vorgabe ihrer Masse m, Wärmekapazi-

tät c_p, maximale Wärmeübergangsfläche A und Wärmeübergangskoeffizienten k beschreiben.

Eine Punktmasse kann thermische Energie aus verschiedenen Quellen erhalten.

- Befindet sich die Punktmasse in einem Kreislauf, so findet ein Wärmeübergang zwischen Punktmasse und dem umströmenden Medium statt:

$$\dot{Q} = k \cdot A \cdot (T_m - T_{pm}). \tag{2.1}$$

- Zwischen zwei Punktmassen kann es über eine Wärmeleitkomponente zu einem Wärmeübergang kommen:

$$\dot{Q} = \frac{\lambda \cdot A}{l} \cdot (T_{pm_1} - T_{pm_2}), \tag{2.2}$$

  wobei l die Länge und λ die Wärmeleitzahl der Wärmeleitkomponente sind.

- Eine Punktmasse kann thermische Energie aus externen Quellen erhalten:

$$\dot{Q} = \dot{Q}_{ext}. \tag{2.3}$$

Insgesamt erhält man damit eine thermische Energie von:

$$\dot{Q} = k \cdot A \cdot (T_m - T_{pm}) + \sum_i \frac{\lambda_i \cdot A}{l_i}(T_{pm_i} - T_{pm}) + \dot{Q}_{ext}. \tag{2.4}$$

Diese thermische Energie lässt sich aber auch als Temperaturänderung der Punktmasse beschreiben:

$$\dot{Q} = m \cdot c_p \cdot \frac{dT_{pm}}{dt}. \tag{2.5}$$

Durch Gleichsetzen der beiden Gleichungen erhält man eine Differentialgleichung für die Temperatur der Punktmasse:

$$\frac{dT_{pm}}{dt} = \frac{1}{m \cdot c_p}\left(k \cdot A \cdot (T_m - T_{pm}) + \sum_i \frac{\lambda_i \cdot A}{l_i}(T_{pm_i} - T_{pm}) + \dot{Q}_{ext}\right). \tag{2.6}$$

2.2 Ein Beispiel eines Kühlsystems

Ein Kühlsystemmodell in Kuli besteht aus inneren Kreisläufen (z.B. Kühlflüssigkeitkreislauf) und der Luftseite. In Abbildung 2.2 sind inneren Kreisläufe in

Kuli dargestellt, die zusammen mit der Luftseite in Abbildung 2.3 ein Kühlsystem modellieren. Die beiden inneren Kreisläufe in Abbildung 2.2 sind ein Öl- (rot) und ein Kühlflüssigkeitkreislauf (blau).

Ziel dieses Modells ist die transiente Simulation eines Aufheizvorgangs. Wie man aus den Simulationsparametern in Abbildung 2.5 erkennen kann, beträgt die Umgebungstemperatur lediglich $-10°C$. Das Fahrzeug fährt langsam weg und erreicht nach 200 Sekunden eine Geschwindigkeit von 50 km/h, die es dann bis zum Ende der Simulation (1800 Sekunden) hält.

Der Motor ist mithilfe eines Wärmeleitnetzwerkes modelliert. Er befindet sich auf der rechten Seite von Abbildung 2.2 und besteht aus fünf Wärmeleitkomponenten, die die Punktmassen in den beiden inneren Kreisläufen miteinander, mit jeweils einer isolierten Punktmasse und mit einer weiteren Punktmasse, deren Temperatur konstant ist, verbinden.

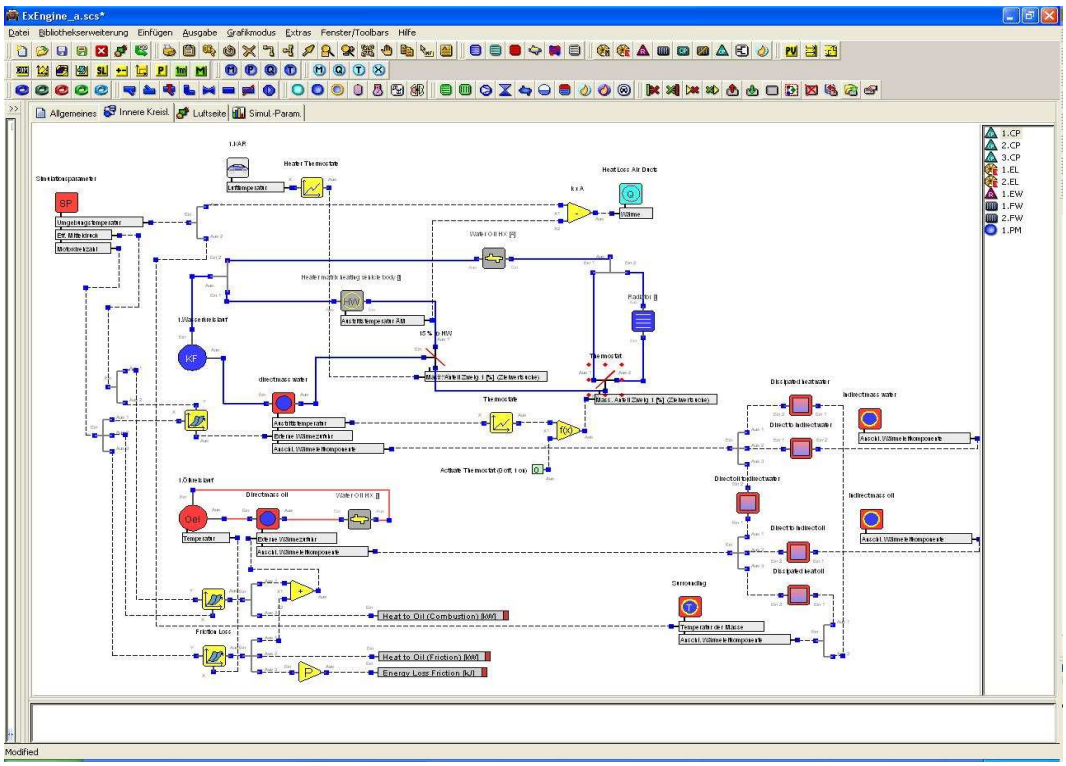


Abbildung 2.2: Innere Kreisläufe

So gilt beispielsweise für die thermische Energie, die sich für die Punktmasse des Ölkreislaufes (rot), aus dem Wärmeleitnetzwerkes ergibt:

$$\dot{Q} = \sum_{i=1}^{3} \frac{\lambda_i \cdot A_i}{l_i} \cdot (T_{pm_i} - T_{pm_{oel}}),$$

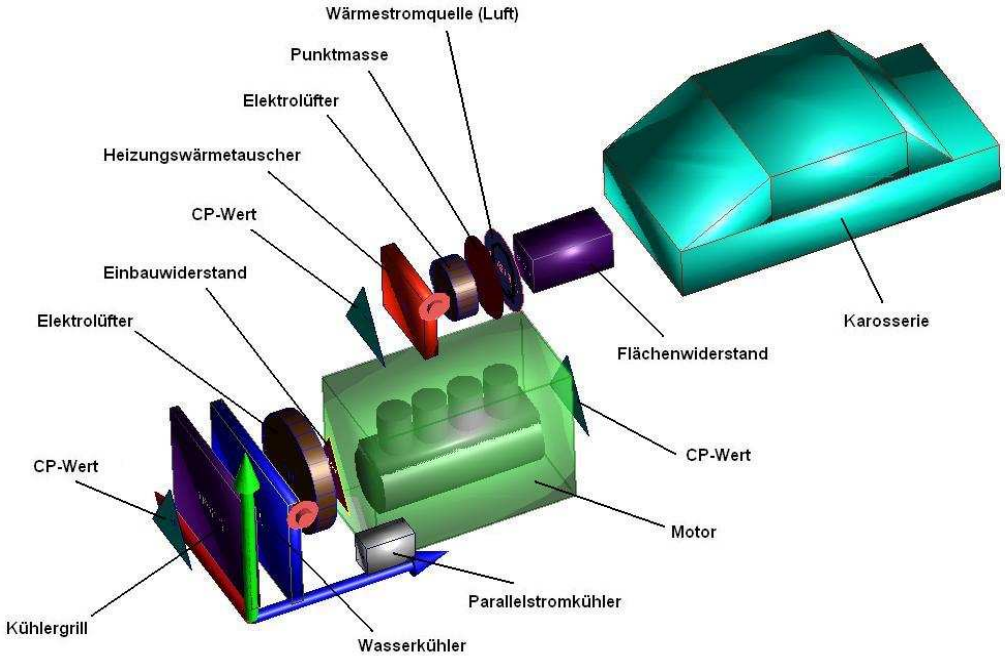


Abbildung 2.3: Luftseite

wobei $T_{pm_{oel}}$ die Temperatur der Punkmasse im Ölkreislauf ist, T_{pm_1} die Temperatur der Punktmasse im Kühlflüssigkeitkreislauf, und damit λ_1, A_1, l_1 die Parameter der verbindenden Wärmeleitkomponente, T_{pm_2} die Temperatur der verbundenen isolierten Punktmasse und damit λ_2, A_2, l_2 die Parameter der entsprechenden Wärmeleitkomponente und T_{pm_3} die Temperatur der isolierten Punktmasse mit fixer Temperatur und λ_3, A_3, l_3 wiederum die Parameter der Wärmeleitkomponente.

In den beiden Kreislaufkomponenten der inneren Kreisläufe (Oel (roter Punkt) und KF (blauer Punkt)) sind Massenstrom, Druckniveau und zugeführte Wärme des jeweiligen Kreislaufs vorgegeben. Über einen Parallelstromkühler (Water Oil HX), der sich in beiden Kreisläufen befindet, findet ein Wärmeübergang zwischen Öl und Kühlflüssigkeit statt. Während der Ölkreislauf nur aus der Kreislaufkomponente, einer Punktmasse und dem Parallelstromkühler besteht, ist der Kühlflüssigkeitkreislauf etwas komplizierter.

Betrachten wir den Kühlflüssigkeitkreislauf anhand von Abbildung 2.4 etwas genauer. In ihm befinden sich die Kreislaufkomponente (KF), eine Punktmasse, zwei Ventile, zwei Sammlungen, ein Heizungswärmetauscher (HW), ein Wasserkühler (Radiator) und ein Parallelstromkühler (Water Oil HX).
Ausgehend von der Kreislaufkomponente geht es nach rechts zur Punktmasse

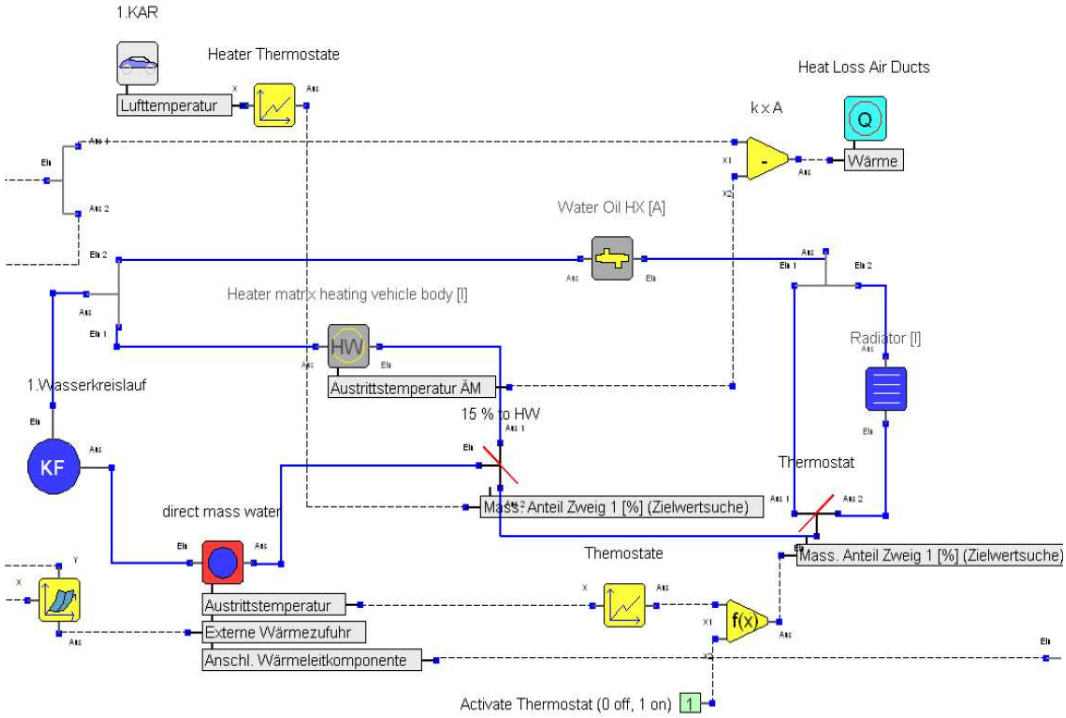


Abbildung 2.4: Kühlflüssigkeitskreislauf

und weiter zum ersten Ventil. Es regelt abhängig von der Innenraumtemperatur des Fahrzeugs die Aufteilung des Massenstroms einerseits zum Heizungswärmetauscher, der für die Temperaturregelung des Innenraums zuständig ist, und andererseits zum Wasserkühler.
Vor dem Wasserkühler befindet sich ein weiteres Ventil, welches abhängig von der Austrittstemperatur des Kühlmittels bei der Punktmasse, einen Teil des Kühlmittels durch den Wasserkühler und den Rest daran vorbei schickt. Der Grund hierfür ist, dass man das Kühlmittel und damit verbunden (Parallelstromkühler) das Öl auf eine gewisse Temperatur bringen möchte, nämlich auf die optimale Betriebstemperatur des Fahrzeugs. Man lässt also immer nur soviel Kühlmittel durch den Wasserkühler, wie nötig ist, um diese Temperatur zu halten.

Bei der Darstellung der Luftseite in Abbildung 2.3 strömt die Luft von links in zwei verschiedene Zweige ein. Im unteren befindet sich der Wasserkühler und im oberen der Heizungswärmetauscher. Der Wasserkühler dient der Kühlung des Kühlmittels im Kühlflüssigkeitskreislauf und der Heizungswärmetauscher dient der Erwärmung der Luft im Innenraum des Fahrzeugs.

Wie schon oben erwähnt ist jeder Kreislauf als (gerichteter) Graph modelliert, in dem jede Komponente einer Kante entspricht, die mit beliebig vielen anderen

Zeit [s]	Motordrz [U/min]	pme [bar]	Fahrg.	Aufw.temp [K]	Umg.temp.	A/C ein	1.EL St.Nr:	2.EL St.Nr:
0	2000	3	0	0	-10	Aus	3	3
10	2500	4	20	0	-10	Aus	3	3
100	3000	5	40	0	-10	Aus	3	3
200	2500	4	50	0	-10	Aus	3	3
500	2000	2	50	0	-10	Aus	3	3
800	1500	1.5	50	0	-10	Aus	3	3
1800	1500	1.5	50	0	-10	Aus	3	3

Abbildung 2.5: Simulationsparameter

Komponenten über Knoten verknüpft werden kann. Die Zustandsgrößen des Modells sind *Massenstrom, Temperatur* und *Druck*. Jedem Knoten werden dabei Temperatur und Druck zugewiesen und jeder Kante ein Massenstrom.

Der Massenstrom ändert sich nur bei Verzweigungen. Er bleibt also für einen bestimmten Zweig des Graphen aufgrund der Massenerhaltung konstant. Für das Beispiel ergeben sich somit zwei unterschiedliche Massenströme für die Luftseite, einer für den Ölkreislauf und sechs für den Kühlflüssigkeitkreislauf.

Der Druck ändert sich nur in bestimmten Komponenten, durch Vorgabe einer Druckverlustkennlinie oder -funktion, die von Massenstrom, Eingangstemperatur und Eingangsdruck der Komponente abhängt. Für den Druckverlust wiederum ergeben sich Bedingungen aus der Maschenregel. Im Beispiel besteht der Kühlflüssigkeitkreislauf aus drei Maschen. Unterschiedlicher Druck ergibt sich zwischen zwei Komponenten, wobei Punktmassen und Sammlungen den Druck nicht verändern. Für unser Beispiel ergeben sich somit zumindest zwei Drücke im Ölkreislauf, acht im Kühlflüssigkeitkreislauf und zwölf auf der Luftseite.

Die Temperatur der Kreisläufe ändert sich durch Wärmeübergänge in den Komponenten, die wiederum durch Kennlinien oder Funktionen in Abhängigkeit der anderen Zustände gegeben sind. Damit ergeben sich zumindest drei unbekannte Temperaturen im Ölkreislauf des Beispiels, sieben im Kühlflüssigkeitkreislauf und fünf auf der Luftseite, da dort nur Wasserkühler, Heizungswärmetauscher, Punktmasse und die Wärmerstromquelle die Temperatur verändern.
Für die Temperaturen der Punktmasse ergeben sich fünf Differentialgleichungen. Dabei befindet sich jeweils eine Punktmasse in Öl- und Kühlflüssigkeitskreislauf, eine auf der Luftseite und zwei weitere im Wärmeleitnetzwerk. Die sechste Punktmasse des Kühlsystems besitzt konstante Temperatur.

Um nun das Kühlsystem nach Massenstrom, Temperatur und Druck aufzulösen, muss ein Gleichungssystem aufgestellt werden, das soviele Gleichungen wie Unbekannte hat. In unserem Fall ergeben sich somit neun Zustandsgleichungen

für den Massenstrom, 22 für den Druck und 15 für die Temperaturen. Für die Temperaturen der Punktmassen erhält man fünf Differentialgleichungen.

2.3 Problemstellung

Um das Gleichungssystem auf mathematische Art zu formulieren, fassen wir die algebraischen (nicht-differentiellen) Größen zu dem Vektor z und die differentiellen zu y zusammen. Im obigen Beispiel hätte z 46 Einträge und y fünf. Die 46 Einträge für z ergeben sich aus den hergeleiteten neun Massenströme, den 22 Drücken und den 15 Temperaturen im gesamten Kühlsystem. Die fünf Einträge für y beschreiben die Temperaturen der Punktmassen.
Das Ziel der transienten Rechnung ist es, Näherungen für diese Größen über einen bestimmten Zeitraum zu berechnen. Damit ergibt sich als zu lösendes Problem:

Problem 2.1 *Gesucht sind die Funktionen* $(y, z) : [0, T] \to \mathbb{R}^r \times \mathbb{R}^s$ *die folgendes Gleichungssystem erfüllen:*

$$y'(t) = f(t, y(t), z(t)), \qquad t \in [0, T] \tag{2.7a}$$

$$0 = g(t, y(t), z(t)), \qquad t \in [0, T] \tag{2.7b}$$

mit $y(0) = y_0$.

Für das obige Beispiel besteht $f : D \subseteq [0, T] \times \mathbb{R}^5 \times \mathbb{R}^{46} \to \mathbb{R}^5$ aus den fünf rechten Seiten der Differentialgleichungen der Temperaturen der Punktmassen. Diese Differentialgleichungen sind durch (2.6) beschrieben. Die restlichen Zustandsgleichungen für Massenstrom, Druck und den restlichen Temperaturen sind durch $g : D \subseteq [0, T] \times \mathbb{R}^5 \times \mathbb{R}^{46} \to \mathbb{R}^{46}$ abstrahiert. Diese Zustandsgleichungen ergeben sich aus Maschen- und Knotenregel, Energiebilanz und vorgegebenen Funktionen oder Kennlinien in den Komponenten.

Dieses Gleichungssystem beschreibt ein System von semi-expliziten Algebro-Differentialgleichungen (engl. Differential Algebraic Equations, kurz DAEs). Semi-explizit bedeutet, dass Differentialgleichung und algebraische Gleichung getrennt auftreten. Semi-explizite DAEs werden auch als gewöhnliche Differentialgleichungen (engl. Ordinary Differential Equations, kurz: ODEs) mit Nebenbedingung bezeichnet.

Für unsere konkrete Problemstellung in Kuli treffen wir die berechtigte Annahme, dass die Gleichung (2.7b) eindeutig nach z auflösbar ist:

$$z(t) = G(t, y(t)). \tag{2.8}$$

Die Lösung von (2.8) beschreibt nichts anderes als die Durchführung eines stationären Abgleichs in KULI zu einem festen Zeitpunkt mit dazugehörigen Punktmassentemperaturen. Setzt man (2.8) in (2.7a) ein, so lässt sich das Problem auf ein System expliziter gewöhnlicher Differentialgleichungen reduzieren

$$y'(t) = f\left(t, y(t), G(t, y(t))\right), \qquad t \in [0, T].$$

Somit reduziert sich Problem 2.1 auf:

Problem 2.2 *Gesucht ist die Funktion* $y : [0, T] \to \mathbb{R}^r$ *die folgendes Gleichungssystem erfüllt:*

$$y'(t) = f(t, y(t), G(t, y(t))), \qquad t \in [0, T] \tag{2.9}$$

mit $y(0) = y_0$.

Zur Klassifizierung von Algebro-Differentialgleichungen wurde der Begriff des *Index* von DAEs eingeführt. Dabei gibt es verschiedene Index-Versionen, es gilt jedoch stets, je höher der Index, desto schwieriger gestaltet sich die numerische Lösung des Problems [1].

Die hier verwendete Art des Index von Algebro-Differentialgleichungen ist allgemein folgendermaßen definiert (vgl. Definition VII.1.2. in [2]):

Definition 2.1 *Eine implizite Differentialgleichung* $F(t, u(t), u'(t)) = 0$ *hat Differentiationsindex* ν_d*, falls* $m = \nu_d$ *die kleinste Zahl ist, sodass das System*

$$\begin{aligned} F(t, u(t), u'(t)) &= 0, \\ \frac{d}{dt} F(t, u(t), u'(t)) &= 0, \\ &\vdots \\ \frac{d^m}{dt^m} F(t, u(t), u'(t)) &= 0, \end{aligned}$$

durch algebraische Manipulation in ein explizites Differentialgleichungssystem der Form

$$u'(t) = f(t, u(t))$$

transformiert werden kann.

Bemerkung: Implizite ODEs

$$F(t, u(t), u'(t)) = 0,$$

mit regulärer Ableitung $F_{u'}$ besitzen Index 0 im Sinne dieser Definition. Das schließt im Speziellen explizite ODEs ein.

Für die semi-explizite DAEs aus unserer Problemstellung

$$y'(t) = f(t, y(t), z(t)) \tag{2.7a}$$
$$0 = g(t, y(t), z(t)) \tag{2.7b}$$

gilt, dass die zweite Gleichung eindeutig nach z auflösbar ist. Durch die Regularität von $g_z(t, y(t), z(t))$ und dem Satz über implizite Funktionen lässt sich die zweite Gleichung eindeutig nach z auflösen. Durch einmaliges Differenzieren von (2.7b) erhält man:

$$0 = g_t(t, y(t), z(t)) + g_z(t, y(t), z(t))z'(t) + g_y(t, y(t), z(t))y'(t).$$

Aufgrund der Regularität von $g_z(t, y(t), z(t))$ ergibt sich somit für $z'(t)$:

$$z'(t) = -g_z(t, y(t), z(t))^{-1} \left(g_t(t, y(t), z(t)) + g_y(t, y(t), z(t))f(t, y(t), z(t))\right)$$

Somit reicht einmaliges Differenzieren aus um das Gleichungssystem in ein System expliziter ODEs gleicher Dimension überzuführen. Daher gilt $\nu_d = 1$.

Ein Anfangswertproblem für explizite ODEs definieren wir folgendermaßen:

Problem 2.3 *Sei $f : D \subseteq [0, T] \times \mathbb{R}^N \to \mathbb{R}^N$ hinreichend glatt. Gesucht ist eine stetig differenzierbare Funktion $u : [0, T] \to \mathbb{R}^N$, die das folgende Anfangswertproblem erfüllt:*

$$u'(t) = f(t, u(t)) \qquad t \in [0, T] \tag{2.10}$$
$$u(0) = u_0.$$

Semi-explizite DAEs vom Index 1 haben den Vorteil, dass sich nahezu alle Eigenschaften expliziter ODEs direkt übertragen lassen. So folgen lokale Existenz, Eindeutigkeit und Regularität der Lösung aus der Theorie gewöhnlicher Differentialgleichungen. Ebenso lassen sich die bekannten numerischen Verfahren zur Lösung expliziter ODEs auf semi-explizite DAEs vom Index 1 ohne Schwierigkeiten anwenden, was für DAEs mit höherem Index nicht der Fall ist (siehe [1]). Daher werden im Folgenden stets die Themen anhand von Problem 2.3 behandelt und danach ihre Übertragung auf Problem 2.2 bzw. Problem 2.1 (kurz) erläutert.

Für die numerische Behandlung der Probleme 2.3, 2.2 und 2.1 setzen wir im Folgenden immer die Existenz und Eindeutigkeit der Lösung voraus.

Die Existenz und Eindeutigkeit der Lösung für Problem 2.3 ergibt sich z.B. aus der Lipschitzstetigkeit von f: Es gibt eine Konstante $L > 0$, sodass für alle $t \in [0, T]$ und für alle $u, v \in \mathbb{R}^N$ die folgende Abschätzung gilt:

$$\|f(t, u) - f(t, v)\| \leq L \|u - v\| . \tag{2.11}$$

Daher setzen wir im weiteren stets voraus, dass f die Lipschitzbedingung bezüglich der zweiten Komponente erfüllt. Da bei Problem 2.2 in f implizit die Funktion G berechnet wird, ist eine weitere sinnvolle Voraussetzung die Lipschitzstetigkeit von G. Diese Bedingungen sind auch für die Analyse der numerischen Verfahren relevant.

Kapitel 3

Beschreibung der numerischen Methoden

In diesem Kapitel werden zwei Klassen von Methoden zur Berechnung der Problemstellungen aus Kapitel 2 beschrieben. Die expliziten Runge-Kutta-Verfahren und eine Klasse der linearen Mehrschrittverfahren, die Adams-Verfahren. Runge-Kutta-Verfahren zählen zu den populärsten Methoden zur numerischen Behandlung von Anfangswertproblemen, da sie in vielen Fällen Resultate mit hoher Genauigkeit mittels akzeptablen Rechenaufwands liefern. Allerdings benötigen sie zur Berechnung der nächsten Näherung die Auswertung der Funktion an Zwischenschritten. Ist die Auswertung der Funktion, wie in unserem Fall, aufwendig, können Mehrschrittverfahren von Vorteil sein, da diese nur eine zusätzliche Funktionsauswertung pro Schritt benötigen.
Die hier verwendete Beschreibung der numerischen Methoden basiert auf ihrer Darstellung in [3, 4], ihre Erweiterung auf semi-explizite DAEs vom Index 1 auf [2].

3.1 Grundlagen

Die wohl älteste Näherungsmethode zur numerischen Behandlung von Differentialgleichungssystemen stammt von Euler und dient im Folgenden der Einführung in dieses Gebiet.
Durch Integration erhält man aus der Anfangswertaufgabe (2.10) die Integraldarstellung

$$u(t) = u_0 + \int_0^t f(\tau, u(\tau))d\tau. \tag{3.1}$$

Der Intervall $[0, T]$ wird durch eine Folge von Gitterpunkten

$$0 = t_0 < t_1 < \ldots < t_m = T$$

mit (nicht notwendigerweise äquidistanter) Schrittweite

$$h_n = t_{n+1} - t_n$$

diskretisiert mit $h = \max_n h_n$. Für äquidistante Unterteilungen gilt $h_n = \frac{T}{m}$ $n = 0, \ldots, m-1$. Die Gitterpunkte ergeben sich möglicherweise erst im Laufe der Rechnung aus der automatischen Schrittweitensteuerung (siehe Abschnitt 5.1.2). In einem Teilintervall $[t_n,\, t_{n+1}]$ gilt ebenfalls die Integralbeziehung

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(\tau, u(\tau))d\tau. \tag{3.2}$$

Mithilfe der linksseitigen Rechteckregel

$$\int_t^{t+h} f(\tau, u(\tau))d\tau \approx hf(t, u(t))$$

gelangt man zu der Näherung

$$u(t_{n+1}) \approx u(t_n) + h_n\, f(t_n, u(t_n)).$$

Diese Näherung führt zur Formel des *Eulerschen Polygonzugverfahrens*

$$u_{n+1} = u_n + h_n\, f(t_n, u_n), \quad n = 0, 1, \ldots, m-1$$

zur sukzessiven Berechnung der Näherungen $u_n \approx u(t_n)$. Verbindet man die berechneten Näherungen an den Gitterpunkten linear so ergibt sich das Euler Polygon

$$u_h(t) = u_n + (t - t_n)f(t_n, u_n) \quad \text{für} \;\; t_n \le t \le t_{n+1}$$

als Näherung der gesuchten Funktion $u(t)$.

Ziel eines jeden Verfahrens ist es, durch Wahl einer entsprechend kleinen Schrittweite eine kleine Abweichung der Näherungslösung $u_h(t)$ von der exakten Lösung $u(t)$ zu erreichen:

$$e_h(t) \to 0 \;\; \text{für} \;\; h \to 0 \text{ für alle } t \in [0, T] \tag{3.3}$$

mit

$$e_h(t) = u(t) - u_h(t). \tag{3.4}$$

Die Differenz $e_h(t)$ wird als der *globale Fehler* des Verfahrens bezeichnet. Gilt (3.3) in einem geeigneten Sinn, so spricht man von einem *konvergenten Verfahren*. Üblicherweise beschränkt man sich darauf den globalen Fehler an den Gitterpunkten zu betrachten:

$$e_n = u(t_n) - u_n, \quad n = 1 \ldots, m. \tag{3.5}$$

Er setzt sich aus den Einzelfehlern der vorherigen Integrationsschritte zusammen (siehe Abbildung 3.1).
Der lokale Fehler d_n im Punkt t_n ist definiert als die Differenz der exakten Lösung $u(t_n)$ der Differentialgleichung (2.10) und der Näherungslösung $u_h(t_n)$ mit Startpunkt $(t_{n-1}, u(t_{n-1}))$ für $n = 1, \ldots, m$.
Für das Eulersche Polygonzugverfahren ergibt sich für den lokalen Fehler

$$d_n = u(t_n) - u_h(t_n) = u(t_n) - (u(t_{n-1}) + h_{n-1} f(t_{n-1}, u(t_{n-1}))\,. \tag{3.6}$$

Er gibt also den Unterschied zwischen der exakten Lösung und der Näherungslösung nach einem Schritt des Verfahrens bei gleicher Anfangsbedingung im Punkt t_{n-1} für $n = 1, \ldots, m$ an. Für $u(t)$ hinreichend glatt folgt mithilfe der Taylorentwicklung von $u(t_{n+1}) = u(t_n + h_n)$:

$$\begin{aligned} d_{n+1} &= u(t_n) + u'(t_n)\, h_n + u''(t_n)\, \frac{h_n^2}{2} + \ldots - u(t_n) - h_n f(t_n, u(t_n)) \\ &= u''(t_n)\, \frac{h_n^2}{2} + \ldots . = O(h_n^2). \end{aligned}$$

Ein Verfahren heißt *konsistent*, falls

$$d_{n+1} = o(h_n) \tag{3.7}$$

gilt, es heißt konsistent von der *Ordnung* p, wenn

$$d_{n+1} = O(h_n^{p+1}). \tag{3.8}$$

Das Eulersche Polygonzugverfahren ist in diesem Sinne von der Ordnung 1.

Die Stabilität eines Verfahrens gibt Auskunft über die Fortpflanzung von Störungen durch das Verfahren. Man nennt ein Verfahren stabil, falls für verschiedene Anfangswerte v_0 und w_0 und den durch das Verfahren erzeugten Näherungen v_j und w_j folgende Abschätzung

$$\|w_j - v_j\| \leq C \|w_0 - v_0\|\,, \tag{3.9}$$

mit einer von h unabhängigen Konstante C, gilt.
Unter der Voraussetzung, dass f die Lipschitzbedingung (2.11) erfüllt, lässt sich die Stabilität des Eulerschen Polygonzugverfahrens zeigen. Zusammen mit der Konsistenz des Verfahrens folgt dessen Konvergenz.

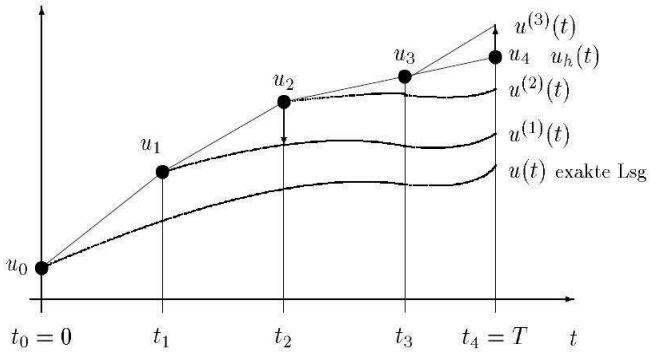


Abbildung 3.1: Fehler des Eulerverfahren

3.2 Explizite Runge-Kutta-Verfahren

Die Konvergenzordnung des Eulerschen Polygonzugverfahrens ist 1, daher sind zum Erzielen genauer Resultate sehr kleine Schrittweiten nötig, was einen hohen, möglicherweise nicht akzeptablen Rechenaufwand zur Folge hat.
Um nun eine höhere Genauigkeit mit derselben Schrittweite zu erreichen, sind Verfahren höherer Ordnung nötig. Diese lassen sich konstruieren, indem man zur Berechnung des Integrals in

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(\tau, u(\tau))d\tau. \tag{3.2}$$

eine genauere Quadraturformel verwendet.

Verbessertes Eulerverfahren

Durch Verwendung der Mittelpunktsregel ergibt sich beispielsweise

$$\int_t^{t+h} f(\tau, u(\tau))d\tau \approx hf(t + \frac{h}{2}, u(t + \frac{h}{2})),$$

wobei $u(t+\frac{h}{2})$ noch nicht bekannt ist. Die Idee von Runge war $u(t+\frac{h}{2})$ wiederum mit einem Schritt des Eulerverfahrens anzunähern,

$$u(t + \frac{h}{2}) = u(t) + \frac{h}{2} f(t, u(t)).$$

Als Resultat erhält man folgendes Verfahren zur Berechung der Näherungen u_{n+1} für $n = 0, \ldots, m-1$:

$$\begin{aligned} k_1 &= f(t_n, u_n), \\ k_2 &= f(t_n + \frac{h_n}{2}, u_n + \frac{h_n}{2} k_1), \\ u_{n+1} &= u_n + h_n k_2, \end{aligned}$$

das verbesserte Eulerverfahren.
Durch Taylorentwicklung des lokalen Fehlers lässt sich zeigen, dass das verbesserte Eulerverfahren von der Ordnung 2 ist.

3.2.1 Explizite ODEs

Verwendet man eine s-stufige Quadratur zur Berechnung des Integrals, so folgt für die Näherung:

$$\int_t^{t+h} f(\tau, u(\tau)) d\tau \approx h \sum_{j=1}^{s} b_j \, f(t + c_j h, u(t + c_j h)). \tag{3.10}$$

Man bezeichnet dabei die Zahlen b_j als die Gewichte und die Punkte $t + c_j h$ als die Stützstellen der Quadraturformel. Anstelle der unbekannten Funktionswerte $u(t + c_j h)$ werden Näherungen g_j rekursiv durch Anwendung von $(j-1)$-stufigen Quadraturformeln auf

$$u(t + c_j h) = u(t) + \int_t^{t+c_j h} f(\tau, u(\tau)) d\tau \tag{3.11}$$

berechnet. Damit erhält man:

$$\begin{aligned} g_1 &= u(t), \\ g_2 &= u(t) + h \, a_{21} \, f(t, g_1), \\ g_3 &= u(t) + h \, \left[a_{31} \, f(t, g_1) + a_{32} \, f(t + c_2 h \, , g_2) \right], \\ &\vdots \\ g_s &= u(t) + h \, \left[a_{s1} \, f(t, g_1) + a_{s2} \, f(t + c_2 h \, , g_2) + \cdots + a_{s,s-1} \, f(t + c_{s-1} h \, , g_{s-1}) \right], \end{aligned} \tag{3.12}$$

und somit für die Näherung

$$u_h(t + h) = u(t) + h \sum_{j=1}^{s} b_j \, f(t + c_j h \, , g_j). \tag{3.13}$$

Eine andere Darstellungsmöglichkeit bietet die Setzung

$$k_i = f(t + c_i h \, , g_i). \tag{3.14}$$

Diese führt zu:

$$\begin{aligned}
k_1 &= f(t, u(t)),\\
k_2 &= f(t + c_2 h\,, u(t) + h\,a_{21}\,k_1),\\
k_3 &= f(t + c_3 h\,, u(t) + h\,(a_{31}\,k_1 + a_{32}\,k_2)),\\
&\vdots\\
k_s &= f(t + c_s h\,, u(t) + h\,(a_{s1}\,k_1 + a_{s2}\,k_2 + \ldots + a_{s,s-1}\,k_{s-1}))
\end{aligned} \tag{3.15}$$

und damit auf

$$u_h(t+h) = u(t) + h \sum_{j=1}^{s} b_j\,k_j. \tag{3.16}$$

Die sich aus (3.12) und (3.13) bzw. (3.15) und (3.16) ergebende Methode beschreibt ein explizites s-stufiges Runge-Kutta-Verfahren. Zur Beschreibung der Methode reicht es, das folgende Tableau anzugeben:

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \ldots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \ldots & b_{s-1} & b_s
\end{array}$$

Durch geeignete Wahl der Koeffizienten des Tableaus erreicht man entsprechend hohe Konsistenzordnungen (siehe Kapitel 4).

Üblicherweise gilt die Bedingung:

$$c_i = \sum_j a_{ij}. \tag{3.17}$$

3.2.2 Erweiterung auf semi-explizite DAEs vom Index 1

Semi-explizite DAEs vom Index 1 kann man als folgendes Gleichungssystem betrachten

$$y'(t) = f(t, y(t), z(t)) \tag{3.18a}$$
$$0 = g(t, y(t), z(t)) \tag{3.18b}$$

oder als explizite ODEs

$$y'(t) = f\left(t, y(t), G(t, y(t))\right). \tag{3.19}$$

Es gibt nun zwei Möglichkeiten ein Runge-Kutta-Verfahren auf semi-explizite DAEs vom Index 1 zu erweitern: den *direkten Zugang*, auch *ϵ-Einbettungsmethode* und den *indirekten Zugang* oder auch *Zustandsraumform-Methode*.

Direkter Zugang

Beim direkten Zugang wird das Gleichungssystem (3.18) als Grenzfall $\epsilon = 0$ der expliziten singulär gestörten ODEs

$$y'(t) = f(t, y(t), z(t)) \tag{3.20a}$$
$$\epsilon z'(t) = g(t, y(t), z(t)) \tag{3.20b}$$

betrachtet. Wendet man nun eine Runge-Kutta-Methode auf (3.20) an, so erhält man

$$y_{n+1} = y_n + h_n \sum_{i=1}^{s} b_i \, Y'_{ni} \qquad z_{n+1} = z_n + h_n \sum_{i=1}^{s} b_i \, Z'_{ni} \tag{3.21}$$

mit

$$Y'_{ni} = f(t_{ni}, Y_{ni}, Z_{ni}) \qquad \epsilon Z'_{ni} = g(t_{ni}, Y_{ni}, Z_{ni}) \tag{3.22}$$

und

$$Y_{ni} = y_n + h_n \sum_{j=1}^{s} a_{ij} \, Y'_{nj} \qquad Z_{ni} = z_n + h_n \sum_{j=1}^{s} a_{ij} \, Z'_{nj}. \tag{3.23}$$

Führt man den Grenzübergang $\epsilon \to 0$ durch, so gilt

$$y_{n+1} = y_n + h_n \sum_{i=1}^{s} b_i \, Y'_{ni} \qquad z_{n+1} = z_n + h_n \sum_{i=1}^{s} b_i \, Z'_{ni} \tag{3.24}$$

mit

$$Y'_{ni} = f(t_{ni}, Y_{ni}, Z_{ni}) \qquad 0 = g(t_{ni}, Y_{ni}, Z_{ni}) \tag{3.25}$$

und

$$Y_{ni} = y_n + h_n \sum_{j=1}^{s} a_{ij} \, Y'_{nj} \qquad Z_{ni} = z_n + h_n \sum_{j=1}^{s} a_{ij} \, Z'_{nj}. \tag{3.26}$$

Bei diesem Zugang ist allerdings zu beachten, dass die Näherungen (y_{n+1}, z_{n+1}) nicht notwendigerweise die algebraische Nebenbedingung

$$0 = g(t_{n+1}, y_{n+1}, z_{n+1}) \tag{3.27}$$

erfüllen.
Für $\epsilon = 0$ lässt sich Z'_{ni} nicht aus Gleichung (3.22) bzw. (3.25) berechnen, sondern nur noch mithilfe der Gleichung (3.26). Eine Bedingung für die Auflösbarkeit von Gleichung (3.26) nach Z'_{ni} ist die Invertierbarkeit der Matrix

(a_{ij}). Bezeichnen wir mit w_{ij} die Element der Inversen der Matrix (a_{ij}) so gilt für Z'_{ni} :

$$h_n Z'_{ni} = \sum_{j=1}^{s} w_{ij}(Z_{nj} - z_n). \tag{3.28}$$

Für die Näherungslösung der algebraischen Größe ergibt sich somit durch Einsezten von (3.28) in (3.24):

$$\begin{aligned} z_{n+1} &= \left(1 - \sum_{i,j=1}^{s} b_i w_{ij}\right) z_n + \sum_{i,j=1}^{s} b_i w_{ij}\, Z_{nj}, \\ 0 &= g(t_{ni}, Y_{ni}, Z_{ni}). \end{aligned} \tag{3.29}$$

Eine Voraussetzung für diesen Zugang ist also die Invertierbarkeit von (a_{ij}) die von explizite Runge-Kutta-Verfahren nicht erfüllt wird, da für sie $a_{ij} = 0$ für $i \leq j$ gilt. Eine Möglichkeit, dennoch explizite Runge-Kutta-Verfahren auf semi-explizite DAEs vom Index 1 anzuwenden, bietet der indirekte Zugang.

Indirekter Zugang

Wendet man auf (3.19) eine explizite Runge-Kutta-Methode an, so ergibt sich folgende Vorschrift:

$$y_{n+1} = y_n + h_n \sum_{i=1}^{s} b_i\, Y'_{ni} \tag{3.30}$$

mit

$$Y'_{ni} = f(t_{ni}, Y_{ni}, G(t_{ni}, Y_{ni})) \tag{3.31}$$

$$Y_{ni} = y_n + h_n \sum_{j=1}^{i-1} a_{ij}\, Y'_{nj} \tag{3.32}$$

und $t_{ni} = t_n + c_i h_n$. Für z_{n+1} erhält man

$$z_{n+1} = G(t_{n+1}, y_{n+1}). \tag{3.33}$$

Durch Einführung der Größe

$$Z_{ni} = G(t_{ni}, Y_{ni}) \tag{3.34}$$

kann man die Methode auch folgendermaßen schreiben:

$$y_{n+1} = y_n + h_n \sum_{i=1}^{s} b_i\, Y'_{ni} \qquad z_{n+1} = G(t_{n+1}, y_{n+1}) \tag{3.35}$$

mit

$$Y'_{ni} = f(t_{ni}, Y_{ni}, Z_{ni}) \tag{3.36}$$

und

$$Y_{ni} = y_n + h_n \sum_{j=1}^{i-1} a_{ij}\, Y'_{nj}, \qquad Z_{ni} = G(t_{ni}, Y_{ni}). \tag{3.37}$$

Der indirekte Zugang bietet einerseits eine Möglichkeit auch explizite Verfahren auf das Problem anzuwenden und andererseits benötigt man aufgrund der Reduktion der DAEs auf ein Anfangswertproblem keine neue Theorie bei der Analyse der Methoden. Im Weiteren beschränken wir uns daher auf diesen Zugang.

3.3 Lineare Mehrschrittverfahren (Adams-Verfahren)

Eine andere Verbesserung des Eulerschen Polygonzugsverfahren wurde durch lineare Mehrschrittverfahren erreicht, welche Adams in Zuge eines Problems von Bashforth entwarf. Im Unterschied zu den Einschrittverfahren, bei deren Berechnung die Differentialgleichung und ein Startwert genügen, bestehen Mehrschrittverfahren, genauer gesagt, k-Schritt-Verfahren, aus zwei Teilen:

1. Einer *Startprozedur* zur Berechnung der Näherungen $u_1, \ldots, u_{k-1}$ der exakten Lösung an den Punkten $t_1, \ldots, t_{k-1}$ und

2. einer *Mehrschritt-Formel* zur Berechnung der Näherung der exakten Lösung $u(t_k)$. Diese Formel wird dann rekursiv angewandt zur Berechnung der nächsten Näherung mithilfe der k vorangegangenen Näherungen.

Für die Startprozedur gibt es verschiedene Möglichkeiten. So kann beispielsweise ein beliebiges Einschrittverfahren, z.B. ein Runge-Kutta-Verfahren, verwendet werden, um die ersten k Näherungen zu berechnen. Eine andere Möglichkeit ist die Verwendung eines Einschrittverfahrens zur Berechnung der ersten Näherung, eines Zweischritt- zur Berechnung der zweiten Näherung usw. bis die gewünschten k Näherungen zur Verfügung stehen. Bei der Startprozedur sollte allerdings darauf geachtet werden, dass die Startwerte von derselben Genauigkeit sind wie das k-Schrittverfahren. Dies kann entweder durch ein Einschrittverfahren derselben Ordnung erreicht werden, oder durch hinreichend kleine Schrittweiten.

In diesem Abschnitt werden wir uns auf die klassische Beschreibung der Adams-Verfahren mit konstanter Schrittweite h und fixer Ordnung beschränken, auch wenn bekannt ist, dass zur effizienten Durchführung von Verfahren sowohl Schrittweite wie auch Ordnung variabel sein sollten, worauf in Kapitel 5 eingegangen wird. Eine fixe Ordnung bedeutet im Falle der Adams-Verfahren eine fixe Anzahl von Vorgängern zur Berechnung der nächsten Näherung, wie wir in Abschnitt 4.2 feststellen werden.

3.3.1 Explizite ODEs

Ziel ist es eine Näherung für $u(t_{n+1})$ zu erhalten unter der Annahme, dass die k Vorgänger $u_n, \ldots, u_{n-k+1}$ und damit auch ihre Ableitungen $f_i = f(t_i, u_i)$ bekannt sind. Diese müssen im Laufe der Berechnung stets abgespeichert werden.

Allgemein ist ein lineares Mehrschrittverfahren zur numerischen Lösung des Anfangswertproblems (2.10) von der Form

$$\sum_{i=0}^{k} \alpha_{k-i}\, u_{n+1-i} = h \sum_{i=0}^{k} \beta_{k-i}\, f_{n+1-i}, \tag{3.38}$$

wobei α_i und β_i reele Parameter sind, mit $\alpha_k \neq 0$, und h die konstante Schrittweite ist. Für $\beta_k \neq 0$ heißt die Methode (3.38) implizit, für $\beta_k = 0$ explizit.

Ausgangspunkt der Adams-Verfahren ist wieder die Integraldarstellung des Anfangswertproblems in einem Teilintervall :

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(\tau, u(\tau))d\tau. \tag{3.2}$$

Adams-Bashforth-Verfahren

Die Idee der Adams-Bashforth-Verfahren ist die Berechnung des Integrals

$$\int_{t_n}^{t_{n+1}} f(\tau, u(\tau))d\tau$$

mithilfe von Extrapolation (siehe Abbildung 3.2).
Da die Näherungen $f_{n-k+1}, \ldots, f_n$ bekannt sind, ist es möglich $f(t, u(t))$ mithilfe eines Interpolationspolynoms durch die Punkte $\{(t_i, f_i)\,|i = n-k+1, \ldots, n\}$ vom Grad $k-1$ anzunähern. Damit erhält man für die nächste Näherung:

$$u_{n+1} = u_n + \int_{t_n}^{t_{n+1}} p(\tau)d\tau. \tag{3.39}$$

Aus der Interpolationtheorie wissen wir, dass es genau ein solches Polynom gibt, welches in verschiedenen Formen dargestellt werden kann.

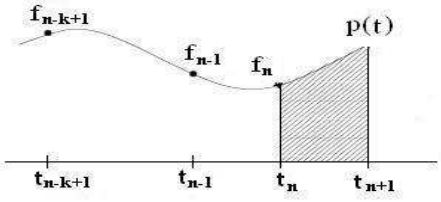


Abbildung 3.2: Idee des Adams-Bashforth-Verfahren

Die Lagrange Form des interpolierenden Polynoms ist:

$$p(t) = \sum_{i=1}^{k} l_i(t) f_{n+1-i} \tag{3.40}$$

mit

$$l_i(t) = \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{t - t_{n+1-j}}{t_{n+1-i} - t_{n+1-j}} \quad i = 1, \ldots, k. \tag{3.41}$$

Eingesetzt in (3.39) erhalten wir:

$$u_{n+1} = u_n + \sum_{i=1}^{k} f_{n+1-i} \int_{t_n}^{t_{n+1}} l_i(\tau) d\tau. \tag{3.42}$$

In der allgemeinen Formulierung (3.38) mit konstanter Schrittweite ergibt sich somit:

$$u_{n+1} = u_n + h \sum_{i=1}^{k} b_{k-i} f_{n+1-i}, \tag{3.43}$$

mit

$$b_{k-i} = \frac{1}{h} \int_{t_n}^{t_{n+1}} l_i(\tau)\, d\tau. \tag{3.44}$$

Ersetzt man in dieser Formulierung die Variable t durch $s = (t - t_n)/h$, so führt dies auf

$$b_{k-i} = \int_0^1 l_i(t_n + sh)\, ds \tag{3.45}$$

$$= \int_0^1 \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{s - 1 + j}{j - i}\, ds. \tag{3.46}$$

Eine andere Möglichkeit bietet sich durch die Darstellung des Polynoms mit dividierenden Differenzen, die sich bei konstanter Schrittweite auf Rückwärtsdifferenzen vereinfachen lassen. Diese sind rekursiv definiert über:

$$\nabla^0 f_n = f_n, \qquad \nabla^{j+1} f_n = \nabla^j f_n - \nabla^j f_{n-1}. \tag{3.47}$$

Mit ihnen kann das Polynom folgendermaßen dargestellt werden (Newton Darstellung):

$$p(t) = p\left(t_n + sh\right) = \sum_{j=0}^{k-1} (-1)^j \binom{-s}{j} \nabla^j f_n. \tag{3.48}$$

Damit ergibt sich zur Berechnung der nächsten Näherung:

$$u_{n+1} = u_n + \int_{t_n}^{t_{n+1}} p\left(\tau\right)\, d\tau \tag{3.49}$$

bzw. wenn man das Polynom durch die Summe ersetzt:

$$u_{n+1} = u_n + h \sum_{j=0}^{k-1} \gamma_j \nabla^j f_n \quad \text{mit} \quad \gamma_j = (-1)^j \int_0^1 \binom{-s}{j} ds. \tag{3.50}$$

Für die Koeffizienten gilt die Formel

$$\sum_{i=0}^{j} \frac{\gamma_{j-i}}{i+1} = 1, \quad j \geq 0, \tag{3.51}$$

aus der diese rekursiv berechnet werden können (siehe Tabelle 3.1).

j	0	1	2	3	4	5	6	7	8
γ_j	1	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{3}{8}$	$\frac{251}{720}$	$\frac{95}{288}$	$\frac{19087}{60480}$	$\frac{5257}{17280}$	$\frac{1070017}{3628800}$

Tabelle 3.1: Koeffizienten der Adams-Bashforth-Methode

Der Vorteil dieser Darstellung ist, dass die Koeffizienten γ_i weder von der Schrittweite noch von der Anzahl k der Vorgänger abhängen. Im Gegensatz zu den Koeffizienten b_{k-i}, die bei einer Änderung von k alle neu berechnet werden müssen. Wie wir später sehen werden, eignet sich die Newton Darstellung für die Implementierung, die Lagrange Darstellung dagegen für die Analyse der Methoden.

Formel (3.50) ergibt sich durch Integration des Interpolationspolynoms von t_n bis t_{n+1}, also außerhalb des Interpolationsintervalls $[t_{n-k+1}, t_n]$. Bekannterweise approximieren Interpolationspolynome nicht sonderlich gut außerhalb ihres Intervalls, weshalb Adams auch Methoden untersuchte, bei denen zusätzlich der Punkt (t_{n+1}, f_{n+1}) interpoliert wird.

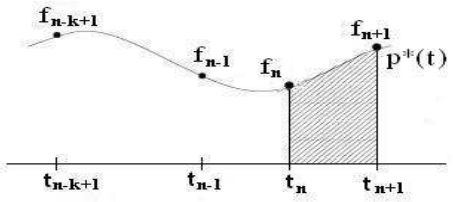


Abbildung 3.3: Idee des Adams-Moulton-Verfahren

Adams-Moulton-Verfahren

Verwendet man bei dem Interpolationspolynom zusätzlich den Punkt (t_{n+1}, f_{n+1}) so ergibt sich für das Polynom in Lagrange Form:

$$p^*(t) = \sum_{i=0}^{k} l_i(t) f_{n+1-i}. \tag{3.52}$$

mit

$$l_i(t) = \prod_{\substack{j=0 \\ j \neq i}}^{k} \frac{t - t_{n+1-j}}{t_{n+1-i} - t_{n+1-j}} \quad i = 0, \ldots, k. \tag{3.53}$$

Somit erhält man für die nächste Näherung in der allgemeinen Formulierung (3.38) mit konstanter Schrittweite:

$$u_{n+1} = u_n + h \sum_{i=0}^{k} b^*_{k-i} f_{n+1-i}, \tag{3.54}$$

mit

$$b^*_{k-i} = \int_0^1 \prod_{\substack{j=0 \\ j \neq i}}^{k} \frac{s - 1 + j}{j - i}\, ds. \tag{3.55}$$

Für die Darstellung des Polynoms mithilfe der Rückwärtsdifferenzen gilt:

$$p^*(t) = p^*\left(t_n + sh\right) = \sum_{j=0}^{k} (-1)^j \binom{-s+1}{j} \nabla^j f_{n+1}. \tag{3.56}$$

Zur Berechnung der nächsten Näherung erhält man damit:

$$u_{n+1} = u_n + h \sum_{j=0}^{k} \gamma^*_j \nabla^j f_{n+1} \quad \text{mit} \quad \gamma^*_j = (-1)^j \int_0^1 \binom{-s+1}{j} ds. \tag{3.57}$$

Die Koeffizienten γ^* (siehe Tabelle 3.2) lassen sich rekursiv mithilfe dieser Formel berechnen:

$$\gamma_0^* = 1$$
$$\sum_{i=0}^{j} \frac{\gamma_{j-i}^*}{i+1} = 0, \quad j \geq 1. \tag{3.58}$$

Die Methode (3.57) nähert die exakte Lösung im Allgemeinen besser an als (3.50). Allerdings ist u_{n+1} nur implizit in der Formel (3.57) gegeben, was normalerweise die Lösung eines nichtlinearen Gleichungssystem in jedem Schritt zur Folge hat.

j	0	1	2	3	4	5	6	7	8
γ_j^*	1	$-\frac{1}{2}$	$\frac{-1}{12}$	$\frac{-1}{24}$	$\frac{-19}{720}$	$\frac{-3}{160}$	$\frac{-863}{60480}$	$\frac{-275}{24192}$	$\frac{-33953}{3628800}$

Tabelle 3.2: Koeffizienten der Adams-Moulton-Methode

Prädiktor-Korrektor-Verfahren

Eine Möglichkeit das nichtlineare Gleichungssystem aus (3.57) zu lösen, ist die Anwendung einer Fixpunktiteration. In der Praxis wird üblicherweise so vorgegangen:

P: (predict) Es wird $\hat{u}_{n+1} = u_n + h \sum_{j=0}^{k-1} \gamma_j \nabla^j f_n$ als Prädiktor mit dem expliziten Adams-Verfahren berechnet.

E: (evaluate) Die Funktion f wird mit dieser Näherung ausgewertet :

$$\hat{f}_{n+1} = f(t_{n+1}, \hat{u}_{n+1}).$$

C: (correct) Die Korrekturformel

$$u_{n+1} = u_n + h(b_k^* \hat{f}_{n+1} + \ldots + b_0^* f_{n-k+1})$$

liefert eine Näherung u_{n+1}.

E: (evaluate) Die Funktion wird nochmals ausgewertet: $f_{n+1} = f(t_{n+1}, u_{n+1})$.

3.3.2 Erweiterung auf semi-explizite DAEs vom Index 1

Wir betrachten wiederum die beiden Darstellungsformen für semi-explizite DAEs vom Index 1:

$$y'(t) = f(t, y(t), z(t)) \tag{3.18a}$$
$$0 = g(t, y(t), z(t)) \tag{3.18b}$$

und

$$y'(t) = f\left(t, y(t), G(t, y(t))\right). \tag{3.19}$$

Wiederum ist ein direkter und indirekter Zugang möglich.

Direkter Zugang

Wendet man auf das Gleichungssystem

$$y'(t) = f(t, y(t), z(t)) \tag{3.20a}$$
$$\epsilon z'(t) = g(t, y(t), z(t)) \tag{3.20b}$$

ein lineares k-Schritt-Verfahren an, so ergibt sich

$$\sum_{i=0}^{k} \alpha_{k-i}\, y_{n+1-i} = h \sum_{i=0}^{k} \beta_{k-i}\, f_{n+1-i} \tag{3.60}$$

$$\epsilon \sum_{i=0}^{k} \alpha_{k-i}\, z_{n+1-i} = h \sum_{i=0}^{k} \beta_{k-i}\, g_{n+1-i} \tag{3.61}$$

mit

$$f_{n+1-i} = f(t_{n+1-i}, y_{n+1-i}, z_{n+1-i}), \quad g_{n+1-i} = g(t_{n+1-i}, y_{n+1-i}, z_{n+1-i}).$$

Betrachtet man das Gleichungssystem (3.18) als Grenzfall $\epsilon = 0$ von diesem System so gilt

$$\sum_{i=0}^{k} \alpha_{k-i}\, y_{n+1-i} = h \sum_{i=0}^{k} \beta_{k-i}\, f_{n+1-i} \tag{3.62}$$

$$0 = \sum_{i=0}^{k} \beta_{k-i}\, g_{n+1-i}. \tag{3.63}$$

Im Falle expliziter Verfahren ist $\beta_k = 0$, wodurch y_{n+1} und z_{n+1} in Gleichung (3.63) nicht mehr vorkommen. Somit ist dieses Gleichungsystem unterbestimmt und nicht mehr eindeutig nach y_{n+1} und z_{n+1} auflösbar. Wie bei den Runge-Kutta-Verfahren ist der direkte Zugang für Mehrschrittverfahren nur auf implizite Verfahren anwendbar.
Eine Möglichkeit, explizite Mehrschrittverfahren auf semi-explizite DAEs vom Index 1 anzuwenden, bietet wiederum der indirekte Zugang.

Indirekter Zugang

Wendet man auf (3.19) eine lineare k-Schritt-Methode an, so erhält man folgende Vorschrift:

$$\sum_{i=0}^{k} \alpha_{k-i}\, y_{n+1-i} = h \sum_{i=0}^{k} \beta_{k-i}\, f_{n+1-i} \tag{3.64}$$

mit

$$f_{n+1-i} = f(t_{n+1-i}, y_{n+1-i}, z_{n+1-i}), \quad z_{n+1-i} = G(t_{n+1-i}, y_{n+1-i}).$$

Beispielsweise ergibt sich bei Anwendung eines Adams PECE-Verfahren auf (3.19) die folgende Berechnungsvorschrift:

P: Berechne

$$\hat{y}_{n+1} = y_n + h \sum_{i=1}^{k} b_{k-i}\, f_{n+1-i},$$
$$\hat{z}_{n+1} = G(t_{n+1}, \hat{y}_{n+1}),$$

als Prädiktoren.

E: Evaluiere $\hat{f}_{n+1} = f(t_{n+1}, \hat{y}_{n+1}, \hat{z}_{n+1})$.

C: Berechne die Näherungen y_{n+1} und z_{n+1} mit der Korrekturformel

$$y_{n+1} = y_n + h \sum_{i=1}^{k} b^*_{k-i}\, f_{n+1-i} + h b^*_k\, \hat{f}_{n+1}$$
$$z_{n+1} = G(t_{n+1}, y_{n+1}).$$

E: Werte die Funktion aus: $f_{n+1} = f(t_{n+1}, y_{n+1}, z_{n+1})$.

Der Vorteil, der sich durch die Reduktion der semi-expliziten DAEs vom Index 1 auf explizite ODEs ergibt, ist die direkte Anwendbarkeit der bereits vorhandenen Theorie wie auch der bekannten numerischen Verfahren. Um dies zu nutzen beschränken wir uns bei den Adams-Verfahren ebenfalls auf den indirekten Zugang.

Kapitel 4

Analyse der numerischen Methoden

Die für die Analyse numerischer Methoden wichtigen Begriffe sind *Konsistenz*, *Stabilität* und *Konvergenz*. In den folgenden Abschnitten werden wir zeigen, dass die Konvergenz der Runge-Kutta-Verfahren und der Adams-Verfahren, aus deren Konsistenz und Stabilität folgt. Hierzu werden der lokale Fehler, die Fortpflanzung von Störungen durch das Verfahren und der globale Fehler untersucht. Da wir uns bei der Erweiterung der Methoden auf eine semi-explizite Algebro-Differentialgleichung vom Index 1 auf den indirekten Zugang beschränkt haben (Abschnitt 3.2.2 und 3.3.2), ist es ausreichend die Methoden anhand des Anfangswertproblems 2.3 zu untersuchen. Die durchgeführte Analyse der Methoden stützt sich dabei auf [3] und [5].

4.1 Explizite Runge-Kutta-Verfahren

Ein Runge-Kutta-Verfahren angewandt auf das Anfangswertproblem 2.3 auf dem Gitter $0 = t_0 < t_1 < \ldots < t_m = T$ mit der Schrittweite $h_n = t_{n+1} - t_n$ kann auch in der folgenden Form geschrieben werden (Einschrittverfahren):

$$\begin{aligned} u_{n+1} &= u_n + h_n \, \phi(t_n, u_n, h_n), \\ u_0 &= u(t_0), \quad n = 0, \ldots, m-1 \end{aligned} \tag{4.1}$$

mit der Verfahrensfunktion $\phi(t_n, u_n, h_n)$, die sich aus den Bedingungen (3.15) und (3.16) ergibt:

$$\phi(t_n, u_n, h_n) = \sum_{j=1}^{s} b_j k_j \tag{4.2}$$

$$k_j = f(t + c_j\, h\,, u(t) + h \sum_{i=1}^{j-1} a_{ji}\, k_i).$$

4.1.1 Konsistenz

Zur Untersuchung der Konsistenz eines Verfahrens wird der lokale Fehler betrachtet. Dieser misst die Differenz der exakten und der Näherungslösung nach einem Schritt des Verfahrens bei selbem Startwert. Für ein Einschrittverfahren gelten folgende Definitionen:

Definition 4.1 (Lokale Fehler und Konsistenz)
Der lokale Fehler des Einschrittverfahrens (4.1) an der Stelle t_n lautet

$$d_n = u(t_n) - (u(t_{n-1}) + h_{n-1}\phi(t_{n-1}, u(t_{n-1}), h_{n-1})) \tag{4.3}$$

für $n = 1, \ldots, m$.
Das Einschrittverfahren heißt konsistent (verträglich), falls (in einem geeigneten Sinn)

$$d_n = o(h_{n-1}), \quad n = 1, \ldots, m. \tag{4.4}$$

Es heißt konsistent mit der Konsistenzordnung p, falls

$$d_n = O(h_{n-1}^{p+1}), \quad n = 1, \ldots, m. \tag{4.5}$$

Durch Taylorentwicklung des lokalen Fehlers ergibt sich für ein Runge-Kutta-Verfahren der Ordnung p, dass die Taylorreihen der exakten und der Näherungslösung bis einschließlich des Terms h^p übereinstimmen.

Betrachten wir beispielsweise die Taylorentwicklung der exakten Lösung $u(t + h)$

und der Näherungslösung $u_h(t+h)$ einer 2-stufige Runge-Kutta-Formel:

$$\begin{aligned}
u(t+h) &= u(t) + hu'(t) + \frac{h^2}{2}u''(t) + \frac{h^3}{6}u'''(t) + \dots \\
&= u + hf + \frac{h^2}{2}\left(f_t + f_u f\right) + \\
&\quad + \frac{h^3}{6}\left(f_{tt} + 2f_{tu}f + f_{uu}f^2 + f_u f_t + f_u^2 f\right) + \dots \\
u_h(t+h) &= u + h\left[b_1 f + b_2 f\left(t + c_2 h, u + h a_{21} f\right)\right] \\
&= u + hb_1 f + hb_2[f + f_t c_2 h + f_u h a_{21} f + \\
&\quad + \frac{1}{2}\left(f_{tt}(c_2 h)^2 + 2 + f_{tu}(c_2 h)(h a_{21} f) + f_{uu}(h a_{21} f)^2\right) + \dots] \\
&= u + h(b_1 + b_2) f + h^2 b_2 (c_2 f_t + a_{21} f_u f) + \\
&\quad + h^3 b_2 \left(\frac{c_2^2}{2} f_{tt} + f_{tu} c_2 a_{21} + \frac{a_{21}^2}{2} f_{uu} f^2\right) + \dots
\end{aligned}$$

Damit diese beiden Reihen bis einschließlich des Terms h^2 übereinstimmen, ergeben sich mithilfe des Koeffizientenvergleichs folgende Bedingungen an die Parameter der Runge-Kutta-Formel:

$$\begin{aligned}
b_1 + b_2 &= 1 & & & b_1 &= 1 - b_2 \\
b_2 c_2 &= 1/2 & \implies & & c_2 &= 1/2b_2 \\
b_2 a_{21} &= 1/2 & & & a_{21} &= 1/2b_2
\end{aligned}$$

Mit beliebig gewähltem b_2 gilt somit für den lokalen Fehler:

$$u(t+h) - u_h(t+h) = \frac{h^3}{6}\left[\left(1 - \frac{3}{4b_2}\right)\left(f_{tt} + 2f_{tu}f + f_{uu}f^2\right) + f_u f_t + f_u^2 f\right] + \dots .$$

Hieraus erkennt man, dass die Ordnung 3 für ein 2-stufiges Verfahren im Allgemeinen nicht erreichbar ist.

Welche Bedingung zumindest Konsistenz gewährleistet zeigt der folgende Satz.

Satz 4.2 *Ein explizites Runge-Kutta-Verfahren ist konsistent genau dann, wenn*

$$\sum_{i=1}^{s} b_i = 1. \tag{4.6}$$

Beweis. Mit der Verfahrensfunktion (4.2)

$$\phi(t, u(t), h) = \sum_{j=1}^{s} b_j k_j,$$

wobei k_j die mit der exakten Lösung $u(t)$ ermittelten Steigungswerte darstellen, erhält man für $h = 0$

$$\phi(t, u(t), 0) = \sum_{j=1}^{s} b_j f(t, u(t)) = f(t, u(t))$$

genau dann, wenn die Konsistenzbedingung erfüllt ist. Also

$$\phi(t, u(t), h) \to f(t, u(t)) \text{ für } h \to 0.$$

Damit gilt für den lokalen Fehler

$$\frac{d_{n+1}}{h_n} = \left[\frac{1}{h_n}\left(u(t_{n+1}) - u(t_n)\right) - u'(t_n)\right] + \left[f(t_n, u(t_n)) - \phi(t_n, u(t_n), h_n)\right],$$

und damit

$$\frac{d_{n+1}}{h_n} \to 0 \text{ für } h_n \to 0.$$

□

Die Ordnung eines Runge-Kutta-Verfahrens ergibt sich aus dessen Konstruktion, also aus der geeigneten Wahl der Parameter des Runge-Kutta-Tableaus:

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \dots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \dots & b_{s-1} & b_s \end{array}$$

Bezeichnet man mit $p(s)$ die maximal erreichbare Konsistenzordnung einer s-stufigen Runge-Kutta-Formel, so gilt:

s	1	2	3	4	5	6	7	8	9	$s \geq 10$
$p(s)$	1	2	3	4	4	5	6	6	≤ 7	$\leq s-3$

Man erkennt hieraus, dass die Ordnung der Runge-Kutta-Verfahren nicht linear mit der Anzahl der Stufen steigt. Eine sorgfältige Untersuchung der erreichbaren Konsistenzordnungen von Runge-Kutta-Verfahren mithilfe von Ordnungsbäumen findet man in [3].

4.1.2 Stabilität

Die Stabilität gibt Auskunft darüber, wie sich Änderungen in den Daten auf das Ergebnis auswirken. Bei stabilen Verfahren ändert sich die Lösung der Näherungsgleichung stetig und gleichmäßig in h mit den Daten.

Betrachtet man das gestörte Einschrittverfahren

$$v_0 = u_0 + \delta_0, \tag{4.7a}$$
$$v_{n+1} = v_n + h_n \, \phi(t_n, v_n, h_n) + h_n \delta_{n+1}, \tag{4.7b}$$

so erhält man folgende Definition der Stabilität eines Einschrittverfahren:

Definition 4.3 (Stabilität)
Ein Einschrittverfahren heißt stabil, falls es eine Konstante $C > 0$ gibt, sodass

$$\|v_n - u_n\| \leq C \max_{0 \leq k \leq n} \|\delta_k\| \tag{4.8}$$

für alle hinreichend kleinen Störungen δ_k.

Ein hinreichendes Kriterium für die Stabilität eines Einschrittverfahrens ist die Lipschitzstetigkeit der Verfahrensfunktion im zweiten Argument: Es gibt eine Konstante $L_\phi \geq 0$, sodass folgende Abschätzung gilt

$$\|\phi(t, v, h) - \phi(t, u, h)\| \leq L_\phi \|v - u\| \quad \forall t, v, u, h. \tag{4.9}$$

Satz 4.4 *Die Verfahrensfunktion $\phi(t, u, h)$ erfülle (4.9). Dann gilt die Abschätzung:*

$$\|v_n - u_n\| \leq e^{L_\phi t_n} \|\delta_0\| + \frac{e^{L_\phi t_n} - 1}{L_\phi} \max_{1 \leq k \leq n} \|\delta_k\|, \tag{4.10}$$

und das Verfahren ist somit stabil.

Beweis. Es gilt

$$\begin{aligned}
\|v_{n+1} - u_{n+1}\| &= \|v_n - u_n + h_n(\phi(t_n, v_n, h_n) - \phi(t_n, u_n, h_n)) + h_n \delta_{n+1}\| \\
&\leq \|(1 + L_\phi h_n)(v_n - u_n) + h_n \delta_{n+1}\| \\
&\leq (1 + L_\phi h_n) \|v_n - u_n\| + h_n \|\delta_{n+1}\|.
\end{aligned}$$

Mit der bekannten Abschätzung $(1 + L_\phi h_n) \leq e^{L_\phi h_n}$ folgt

$$\begin{aligned}
\|v_{n+1} - u_{n+1}\| &\leq e^{L_\phi h_n} \|v_n - u_n\| + h_n \|\delta_{n+1}\| \\
&\leq e^{L_\phi h_n} (e^{L_\phi h_{n-1}} \|v_{n-1} - u_{n-1}\| + h_{n-1} \|\delta_n\|) + h_n \|\delta_{n+1}\|.
\end{aligned}$$

Setzt man dies fort, so erhält man zusammen mit $h_n+h_{n-1}+\ldots+h_j = t_{n+1}-t_j$ und $t_0 = 0$

$$\begin{aligned}\|v_{n+1} - u_{n+1}\| &\leq e^{L_\phi t_{n+1}} \|v_0 - u_0\| + \sum_{j=1}^{n+1} e^{L_\phi(t_{n+1}-t_j)} h_{j-1} \|\delta_j\| \\ &\leq e^{L_\phi t_{n+1}} \|\delta_0\| + \sum_{j=1}^{n+1} e^{L_\phi(t_{n+1}-t_j)} h_{j-1} \max_{1\leq k\leq n+1} \|\delta_k\| .\end{aligned}$$

Der Ausdruck $e^{L_\phi(t_{n+1}-t_j)}h_{j-1}$ lässt sich durch ein Integral abschätzen:

$$e^{L_\phi(t_{n+1}-t_j)}h_{j-1} \leq \int_{t_{j-1}}^{t_j} e^{L_\phi(t_{n+1}-t)}dt.$$

Damit ergibt sich für die Summe folgende Abschätzung

$$\begin{aligned}\sum_{j=1}^{n+1} e^{L_\phi(t_{n+1}-t_j)}h_{j-1} &\leq \sum_{j=1}^{n+1} \int_{t_{j-1}}^{t_j} e^{L_\phi(t_{n+1}-t)}dt \\ &= \int_{t_0}^{t_{n+1}} e^{L_\phi(t_{n+1}-t)}dt \\ &= \frac{e^{L_\phi t_{n+1}} - 1}{L_\phi},\end{aligned}$$

und somit gilt insgesamt die Behauptung. □

Die Verfahrensfunktion eines Runge-Kutta-Verfahrens besteht aus einer Summe über f:

$$\phi(t, u, h) = \sum_{i=1}^{s} b_i f(t + c_i h, g_i).$$

Daher lässt sich zeigen, dass für stetige Funktionen $f(t, u)$, die die Lipschitzbedingung (2.11) erfüllen, die Verfahrensfunktion lispschitzstetig ist und somit die Runge-Kutta-Formel stabil.

Satz 4.5 *Sei L die Lipschitzkonstante von f und $h = \max_i h_i$. Dann erfüllt die Verfahrensfunktion der Runge-Kutta-Methode (4.2) die Lipschitzbedingung (4.9) mit*

$$L_\phi = L\left(\sum_i |b_i| + hL\sum_{i,j} |b_i a_{ij}| + h^2L^2\sum_{i,j,k} |b_i a_{ij} a_{jk}| + \ldots\right).$$

Beweis. Für die Verfahrensfunktion (4.2) gilt folgende Beziehung

$$\begin{aligned}\|\phi(t,v,h)-\phi(t,u,h)\| &\leq \sum_{i=1}^{s} |b_i| \, \|f(t+c_ih, g_i(t,v,h)) - f(t+c_ih, g_i(t,u,h))\| \\ &\leq L \sum_{i=1}^{s} |b_i| \, \|g_i(t,v,h) - g_i(t,u,h)\|\end{aligned}$$

mit

$$g_i(t,v,h) = v + h\sum_{j=1}^{i} a_{ij} f(t+c_jh, g_j(t,v,h)).$$

Aufgrund der Lipschitzstetigkeit von f gilt folgende Abschätzung:

$$\|g_i(t,v,h) - g_i(t,u,h)\| \leq \|v-u\| + hL\sum_{j=1}^{i} a_{ij} \, \|g_j(t,v,h) - g_j(t,u,h)\| \, .$$

Setzt man diese Beziehung oben ein, so erhält man die Behauptung.

□

4.1.3 Konvergenz

Ziel eines jeden Verfahrens ist es bei entsprechend kleiner Schrittweite eine kleine Abweichung von der exakten Lösung zu erhalten. Diese Abweichung entspricht dem globalen Fehler des Verfahrens, der folgendermaßen definiert ist:

Definition 4.6 (Globale Fehler und Konvergenz)
Der globale Fehler an der Stelle t_n ist gegeben durch

$$e_n = u(t_n) - \underbrace{u_h(t_n)}_{=u_n}. \tag{4.11}$$

Ein Einschrittverfahren heißt konvergent, falls

$$\max_n \|e_n(t)\| \to 0 \quad \text{für} \quad h \to 0 \text{ für alle } t \in [0,T] \tag{4.12}$$

mit

$$h = \max_i h_i \quad i = 0, \ldots, n-1.$$

Es ist von der Konvergenzordnung p falls

$$\max_n \|e_n\| = O(h^p). \tag{4.13}$$

Für Einschrittverfahren folgt die Konvergenz des Verfahrens offensichtlich aus dessen Konsistenz und Stabilität. Ebenso gilt, falls die Runge-Kutta-Formel die Konsistenzordnung p hat und f entsprechend oft stetig differenzierbar ist, dann ist auch die Konvergenzordnung p. Der folgende Satz zeigt diesen Zusammenhang.

Satz 4.7 *Die Verfahrensfunktion erfülle die Lipschitzbedingung (4.9) und das Verfahren sei von der Konsistenzordnung p, d.h.*

$$\|d_n\| \leq Ch^{p+1}. \tag{4.14}$$

dann lässt sich der globale Fehler durch

$$\|e_n\| \leq \frac{C}{L_\phi} \left(e^{L_\phi t_n} - 1\right) h^p \tag{4.15}$$

abschätzen.

Beweis. Für den globalen Fehler gilt

$$e_n = d_n + u(t_{n-1}) - u_{n-1} + h_{n-1}(\phi(t_{n-1}, u(t_{n-1}), h_{n-1}) - \phi(t_{n-1}, u_{n-1}, h_{n-1})$$

und somit für seine Norm

$$\|e_n\| \leq Ch^{p+1} + (1 + L_\phi h_{n-1}) \|e_{n-1}\| .$$

Führt man dies wie in dem Beweis von Satz 4.4 fort so erhält man:

$$\|e_n\| \leq Ch^p \sum_{j=1}^{n} e^{L_\phi (t_n - t_j)} h_{j-1} + e^{L_\phi t_n} \underbrace{\|e_0\|}_{=0}$$

und damit die Behauptung. □

4.2 Lineare Mehrschrittverfahren (Adams-Verfahren)

Die Analyse der linearen Mehrschrittverfahren wird anhand der Formulierung (3.38) vorgenommen. Die dadurch erhaltenen Resultate lassen sich ohne Probleme auf die Adams-Verfahren übertragen. Die Analyse der Verfahren wird dabei auf äquidistantem Gitter mit der Schrittweite h durchgeführt. Da man Verfahren mit variabler Schrittweite und Ordnung als Folge von Verfahren mit konstanter Schrittweite mit gelegentlichen Wechsel betrachten kann, ist die Theorie auf nicht-äquidistante Gitter erweiterbar [4].

4.2.1 Konsistenz

Analog zu den Einschrittverfahren heißt ein Mehrschrittverfahren konsistent mit der Konsistenzordnung p, falls für den lokalen Fehler gilt:

$$d_n = O(h^{p+1}), \quad n = k, \ldots, m. \tag{4.16}$$

Definition 4.8 (Lokale Fehler)
Der lokale Fehler einer linearen k-Schrittmethode an der Stelle t_{n+1} ist gegeben durch

$$d_{n+1} = u(t_{n+1}) - u_{n+1} \tag{4.17}$$

wobei für u_{n+1} gilt:

$$\sum_{i=0}^{k} \alpha_{k-i}\, u_{n+1-i} = h \sum_{i=0}^{k} \beta_{k-i}\, f(t_{n+1-i}, u_{n+1-i}), \quad n+1 = k, \ldots, m$$

mit $u_{n+1-i} = u(t_{n+1-i})$ für $i = 1, \ldots, k$.

Der lokale Fehler ergibt sich wiederum als Differenz der exakten Lösung $u(t_{n+1})$ des Anfangswertproblems 2.3 und der Näherungslösung u_{n+1}. Allerdings werden zur Berechnung der Näherungslösung die Werte u_{n+1-i} mit $i = 1, \ldots, k$ als exakt angenommen, also $u_{n+1-i} = u(t_{n+1-i})$.
Eine Darstellungsmöglichkeit des lokalen Fehlers bietet der folgende Operator:

$$L(u, t, h) = \sum_{i=0}^{k} \left[\alpha_i u(t + ih) - h\beta_i u'(t + ih)\right]. \tag{4.18}$$

Lemma 4.9 *Sei $u(t)$ die Lösung des Anfangswertproblems 2.3 mit $f(t, u(t))$ stetig differenzierbar. Dann existiert ein $\delta \in (0, 1)$ sodass für den lokalen Fehler gilt:*

$$d_n = \left(\alpha_k I - h\beta_k f_u(t_n, \eta)\right)^{-1} L(u, t_{n-k}, h), \tag{4.19}$$
$$\text{mit } \eta = \eta(\delta) = u_n + \delta\left[u(t_n) - u_n\right].$$

Beweis. Für u_n gilt laut Definition 4.8 die Gleichung

$$\sum_{i=0}^{k-1} \left[\alpha_i u(t_{n-k+i}) - h\beta f(t_{n-k+i}, u(t_{n-k+i}))\right] + \alpha_k u_n - h\beta_k f(t_n, u_n) = 0.$$

Durch Einsetzen des Operators L erhält man

$$L(u, t_{n-k}, h) = \alpha_k\left[u(t_n) - u_n\right] - h\beta_k\left[f(t_n, u(t_n)) - f(t_n, u_n)\right].$$

Zusammen mit dem Mittelwertsatz folgt die Behauptung. □

Für explizite lineare Mehrschrittverfahren mit $\alpha_k = 1$ stimmen der lokale Fehler und $L(u,t,h)$ überein.

Bei der Bestimmung der Konsistenzordnung linearer Mehrschrittverfahren spielen folgende Polynome eine wichtige Rolle:

$$\rho(\zeta) = \alpha_k \zeta^k + \alpha_{k-1}\zeta^{k-1} + \ldots + \alpha_0, \tag{4.20}$$

$$\sigma(\zeta) = \beta_k \zeta^k + \beta_{k-1}\zeta^{k-1} + \ldots + \beta_0. \tag{4.21}$$

Sie werden als die *erzeugenden Polynome* des Mehrschrittverfahrens bezeichnet. Der folgende Satz gibt Auskunft über den Zusammenhang zwischen den erzeugenden Polynomen und der Ordnung eines Mehrschrittverfahrens.

Satz 4.10 *Die Mehrschrittmethode (3.38) ist von der Ordnung p genau dann, wenn*

$$\sum_{i=0}^{k} \alpha_i = 0 \quad \textit{und} \quad \sum_{i=0}^{k} \alpha_i i^q = q \sum_{i=0}^{k} \beta_i i^{q-1} \quad \textit{für} \quad q = 1, \ldots, p.$$

Beweis. Wir zeigen, dass $L(u,t,h) = O(h^{p+1})$, woraus die Behauptung folgt. Durch Einsetzen der Taylorentwicklung von $u(t+ih)$ und $u'(t+ih)$

$$u(t+ih) = \sum_{j=0}^{p} \frac{u^{(j)}(t)}{j!} i^j h^j + O(h^{p+1}),$$

$$u'(t+ih) = \sum_{j=0}^{p-1} \frac{u^{(j+1)}(t)}{j!} i^j h^j + O(h^p)$$

in (4.18) erhält man:

$$\begin{aligned} L(u,t,h) &= u(t) \sum_{i=0}^{k} \alpha_i + \sum_{j=1}^{p} \frac{u^{(j)}(t)}{j!} h^j \sum_{i=0}^{k} \alpha_i i^j - \sum_{j=1}^{p} \frac{u^{(j)}(t)}{(j-1)!} h^j \sum_{i=0}^{k} \beta_i i^{j-1} + O(h^{p+1}) \\ &= u(t) \underbrace{\sum_{i=0}^{k} \alpha_i}_{=0} + \sum_{j=1}^{p} \frac{u^{(j)}(t)}{j!} h^j \underbrace{\left[\sum_{i=0}^{k} \alpha_i i^j - j \sum_{i=0}^{k} \beta_i i^{j-1} \right]}_{=0 \text{ für } j=1,2,\ldots,p} + O(h^{p+1}) \\ &= O(h^{p+1}). \end{aligned}$$

□

Für den Fall $p = 1$ ergeben sich aus obigem Satz folgende Bedingungen an ein

Mehrschrittverfahren:

$$\rho(1) = 0 \qquad \left[\sum_{i=0}^{k} \alpha_i = 0\right]$$

$$\rho'(1) = \sigma(1) \qquad \left[\sum_{i=0}^{k} i\alpha_i = \sum_{i=0}^{k} \beta_i\right].$$

Mit Satz 4.10 können wir zeigen, dass die Adams-Bashforth-Methode die Ordnung k hat und die Adams-Moulton-Methode die Ordnung $k+1$:

Die verwendetet interpolatorische Quadraturformel der Adams-Bashforth-Methode ist vom Exaktheitsgrad $k-1$, d.h. sie ist für Polynome vom Grad $q \leq k-1$ exakt. Für die Differentialgleichung

$$u'(t) = qt^{q-1} \quad \text{für} \quad q = 1, \ldots, k$$

gilt $u(t) = t^q + c$. Daher ist der lokale Fehler, der bei Anwendung der Adams-Bashforth- Methode entsteht, identisch Null. Somit gilt zusammen mit Lemma 4.9:

$$\begin{aligned}
0 &= [\alpha_k I - 0]\, d_n \\
&= L(t^q + c, t_{n-k}, h) \\
&= \sum_{i=0}^{k} \left[\alpha_i \left[(t_{n-k} + ih)^q + c\right] - h\beta_i q (t_{n-k} + ih)^{q-1}\right] \\
&= c\sum_{i=0}^{k} \alpha_i + \sum_{i=0}^{k} \left[\alpha_i \left[(t_{n-k} + ih)^q\right] - h\beta_i q (t_{n-k} + ih)^{q-1}\right].
\end{aligned}$$

Dabei ist die Gleichung

$$0 = \sum_{i=0}^{k} \alpha_i$$

für Adams-Verfahren stets erfüllt. Für den zweiten Term folgt mit $t_{n-k} = 0$:

$$0 = h^q \sum_{i=0}^{k} \left[\alpha_i i^q - q\beta_i i^{q-1}\right].$$

Somit sind die Bedingungen aus Satz 4.10 erfüllt und das Verfahren ist von der Ordnung k.

Auf diesselbe Art lässt sich mit einem Polynom vom Grad $\leq k$ zeigen, dass die Adams-Moulton-Methode die Ordnung $k+1$ hat.

4.2.2 Stabilität

Zur Untersuchung der Stabilität von Mehrschrittverfahren betrachten wir ihre Anwendung auf die Differentialgleichung mit trivialer rechter Seite:

$$u'(t) = 0. \tag{4.22}$$

Die Gleichung zur Berechnung der Näherungslösung lautet für dieses Differentialgleichungssystem

$$\alpha_k u_{n+1} + \alpha_{k-1} u_n + \ldots + \alpha_0 u_{n-k} = 0. \tag{4.23}$$

Die allgemeine Lösung dieser Gleichung liefert das nächste Lemma:

Lemma 4.11 *Seien $\zeta_1, \zeta_2, \ldots, \zeta_l$ die Wurzeln von $\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j$ mit den Vielfachheiten $m_1, m_2, \ldots, m_l$.*
Dann ist die allgemeine Lösung von (4.23) durch

$$u_n = p_1(n)\zeta_1^n + \ldots + p_l(n)\zeta_l^n$$

gegeben, wobei $p_j(n)$ beliebige Polynome vom Grad $m_j - 1$ sind.

□

Die Forderung nach der Beschränktheit von u_n für $n \to \infty$ motiviert folgenden Stabilitätsbegriff.

Definition 4.12 (0-Stabilität) *Das Mehrschrittverfahren (3.38) heißt 0-stabil, falls die Nullstellen des erzeugende Polynoms $\rho(\zeta)$ folgende Bedingungen erfüllen:*

1. *$|\zeta| \leq 1$, falls ζ eine einfache Nullstelle ist.*
2. *$|\zeta| < 1$, falls ζ eine mehrfache Nullstelle ist.*

Für die Adams-Bashforth- und Adams-Moulton-Methode ist das erzeugende Polynom $\rho(\zeta) = \zeta^k - \zeta^{k-1}$. Es hat eine einfache Nullstelle bei 1 und eine $(k-1)$-fache bei 0, damit sind die Bedingungen der 0-Stabilität erfüllt.

4.2.3 Konvergenz

Durch Formulierung des k-Schrittverfahrens als Einschrittverfahren in einem $(k \cdot N)$-dimensionalen Raum (N ist die Dimension des Differentialgleichungssystems) lässt sich zeigen, dass aus Konsistenz und Stabilität des Verfahrens dessen Konvergenz folgt.

Sei $\varphi = \varphi(t_n, u_n, u_{n+1}, \ldots . u_{n+k-1}, h)$ implizit definiert durch:

$$\varphi = \sum_{j=1}^{k-1} \beta'_j f_{n+j} + h\beta'_k f(t_n + kh, h\varphi - \sum_{j=0}^{k-1} \alpha'_j u_{n+j}), \tag{4.24}$$

mit $\alpha'_j = \alpha_j/\alpha_k$ und $\beta'_j = \beta_j/\alpha_k$. Damit erhält das Mehrschrittverfahren (3.38) die Form:

$$u_{n+k} = -\sum_{j=0}^{k-1} \alpha'_j u_{n+j} + h\varphi. \tag{4.25}$$

Durch Einführung von

$$U_n = \begin{bmatrix} u_{n+k-1} \\ u_{n+k-2} \\ \vdots \\ u_n \end{bmatrix}, \quad A = \begin{bmatrix} -\alpha'_{k-1}I & -\alpha'_{k-2}I & \cdots & -\alpha'_0 I \\ I & O & \cdots & O \\ & \ddots & \ddots & \vdots \\ & & I & O \end{bmatrix}, \quad e_1 = \begin{bmatrix} I \\ O \\ \vdots \\ O \end{bmatrix}$$

kann man das Mehrschrittverfahren (4.25) in kompakter Form im Produktraum $(\mathbb{R}^N)^k = \mathbb{R}^N \times \ldots \times \mathbb{R}^N$ schreiben:

$$U_{n+1} = AU_n + h\Phi(t_n, U_n, h), \quad n = 0, 1, \ldots, m-k \tag{4.26}$$

mit

$$\Phi(t_n, U_n, h) = \varphi(t_n, U_n, h)e_1.$$

Das folgende Lemma sagt aus, dass aus der Konsistenz(-ordnung) des Mehrschrittverfahrens jene des Einschrittverfahrens (4.26) folgt und umgekehrt.

Lemma 4.13 *Sei $u(t)$ die exakte Lösung von (2.10). Der Vektor U_{n+1} sei die numerische Lösung nach einem Schritt des Verfahrens (4.26)*

$$U_{n+1} = AU(t_n) + h\Phi(t_n, U(t_n), h)$$

mit exakten Startwerten

$$U(t_n) = (u(t_{n+k-1}), u(t_{n+k-2}), \ldots, u(t_n))^T.$$

Aus der Konsistenz(-ordnung) des Mehrschrittverfahren folgt die Konsistenz(-ordnung) des zum Mehrschrittverfahren äquivalenten Einschrittverfahren und umgekehrt.

Beweis. Für den lokalen Fehler des Einschrittverfahren (4.26) gilt:

$$U(t_{n+1}) - U_{n+1} = \begin{bmatrix} u(t_{n+k}) \\ u(t_{n+k-1}) \\ \vdots \\ u(t_{n+1}) \end{bmatrix} - \begin{bmatrix} u_{n+1} \\ u(t_{n+k-1}) \\ \vdots \\ u(t_{n++1}) \end{bmatrix}$$

$$= \begin{bmatrix} u(t_{n+k}) - u_{n+1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Daraus folgt:

$$\|U(t_{n+1}) - U_{n+1}\| = \|u(t_{n+1}) - u_{n+1}\| \tag{4.27}$$

und somit die Behauptung. □

Die Stabilität des Einschrittverfahrens (4.26) erhält man aus der Stabilität des Mehrschrittverfahrens und der Lipschitzstetigkeit von f.
Zuvor benötigen wir noch folgendes Lemma:

Lemma 4.14 *Sei das Mehrschrittverfahren 0-stabil. Dann existiert eine Vektornorm in* $(\mathbb{R}^N)^k$*, sodass für die zugeordnete Matrixnorm gilt:*

$$\|A\| \leq 1. \tag{4.28}$$

Beweis: siehe [3] S. 394 f.

Mithilfe dieser Norm lässt sich die Stabilität des Einschrittverfahrens aus der Stabilität des Mehrschrittverfahrens und der Lipschitzstetigkeit von f folgern. Die Vorgehensweise ist dabei analog zu Lemma 4.4. Aus Stabilität und Konsistenz des Einschrittverfahrens (4.26) erhält man offensichtlich dessen Konvergenz. Aufgrund der Äquivalenz von (4.26) und (4.25) folgt somit die Konvergenz des Mehrschrittverfahrens.

Kapitel 5

Praktische Durchführung der numerischen Methoden

Aufgrund des konkreten Problems waren zwei Aspekte bei der praktischen Durchführung der Methoden von besonderem Interesse, die automatische Schrittweitensteuerung und, wie sich im Laufe der Diplomarbeit herausstellte, die kontinuierliche Datenausgabe (engl. dense output), d.h. die Ausgabe der Daten auch zwischen den Gitterpunkten. Für die Realisierung dieser beiden Aspekte sind mehrere Vorgehensweisen möglich, in diesem Kapitel werden jene vorgestellt, die von den in Kapitel 6 getesteten Verfahren verwendet werden.

Für die Runge-Kutta-Verfahren dient DOPRI5 aus [3] als Grundlage zur Beschreibung der für eine effiziente Implementierung wichtigen Aspekte. Eine mögliche Implementierung der Adams-Verfahren wird anhand der Beschreibung aus [3, 4] aufgezeigt, einem PECE-Verfahren mit variabler Ordnung und Schrittweite.

5.1 Explizite Runge-Kutta-Verfahren

Zur effizienten Durchführung von Einschrittverfahren, zu denen die Runge-Kutta-Verfahren zählen, ist die Wahl der richtigen Schrittweite von entscheidender Bedeutung. Die Aufgabe einer Schrittweitensteuerung ist es, den lokalen Fehler in einer gewünschten Größenordnung zu halten. Durch Schrittweitensteuerung werden allerdings die berechneten Gitterpunkte vom Verfahren gewählt und somit beliebig. Wie man dennoch Näherungen an vorgegebenen aber nicht berechneten Punkten erhält, wird in Abschnitt 5.1.3 behandelt.

5.1.1 Eingebettete Runge-Kutta-Verfahren

Eine effiziente Möglichkeit den lokalen Fehler zu schätzen, und somit Schrittweitensteuerung zu ermöglichen, bieten die eingebetteten Runge-Kutta-Verfahren. Die Grundidee der eingebetteten Runge-Kutta-Verfahren besteht darin, neben der zur Berechnung verwendeten Runge-Kutta-Formel der Ordnung p eine weitere Runge-Kutta-Formel der Ordnung $p+1$ zu konstruieren. Die Differenz der beiden Näherungslösungen $u_h(t+h)$ und $\hat{u}_h(t+h)$ liefert dann eine Schätzung für den lokalen Fehler. Um den Aufwand zur Berechnung von $\hat{u}_h(t+h)$ möglichst gering zu halten, benutzt man die selben Werte für die Koeffizienten c_i und a_{ij} aus den beiden Runge-Kutta-Formeln. Diese werden dann in einem gemeinsamem Tableau zusammgefasst:

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \dots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \dots & b_{s-1} & b_s \\
\hline
 & \hat{b}_1 & \hat{b}_2 & \dots & \hat{b}_{s-1} & \hat{b}_s
\end{array}$$

Besonders effizient sind eingebettete Runge-Kutta-Formeln mit der Eigenschaft (Fehlberg-Trick):

$$\begin{aligned} a_{si} &= b_i, \quad i = 1, 2, \dots, s-1 \\ 0 &= b_s. \end{aligned}$$

Für sie gilt:

$$\begin{aligned} k_1(t+h) =& f(t+h, u_h(t+h)) \\ =& f(t+c_s\,h, u_h(t) + h\sum_{i=1}^{s} b_i(t)k_i(t)) \\ =& f(t+c_s\,h, u_h(t) + h\sum_{i=1}^{s-1} a_{si}(t)k_i(t)) \\ =& k_s(t), \end{aligned}$$

da aus der Konsistenzbedingung und Bedingung (3.17) folgt:

$$c_s = \sum_{i=1}^{s-1} a_{si} = \sum_{i=1}^{s} b_i = 1.$$

Man erspart sich also mit solch einer Methode in jedem Schritt eine Auswertung von f. Eine bekannte Runge-Kutta-Formel, die diese Eigenschaft nutzt,

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{503}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Tabelle 5.1: DOPRI5(4)

ist jene von Dormand und Prince, deren Werte-Paare in Tabelle 5.1 dargestellt sind. Das damit konstruierte Verfahren aus [3] nennt sich Dormand-Prince 5(4) (DOPRI5). In ihm wird der Fehlerterm des genaueren Verfahrens 5.Ordnung minimiert und die Näherunge des Verfahrens mit der niedrigeren Ordnung dient der Fehlerschätzung.

In diesem Fall ergibt sich als Schätzung für den Fehler: $u_h(t+h) - \hat{u}_h(t+h)$. Dieser Fehler soll komponentenweise folgende Bedingungen erfüllen:

$$\begin{aligned} |u_{h,i}(t+h) - \hat{u}_{h,i}(t+h)| &\leq tol_i, \\ tol_i = Atol_i + \max(|u_{h,i}(t)|, |u_{h,i}(t+h)|) Rtol_i, & \end{aligned} \tag{5.1}$$

wobei $Atol_i$ und $Rtol_i$ die durch den Benutzer vorgegebenen Fehlertoleranzen sind (relative Fehler ergeben sich mit $Atol_i = 0$, absolute mit $Rtol_i = 0$). Insgesamt erhält man folgendes Fehlermaß

$$err = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\frac{u_{h,i}(t+h) - \hat{u}_{h,i}(t+h)}{tol_i} \right)}. \tag{5.2}$$

Oft werden auch andere Normen, wie beispielsweise die Maximumsnorm, für die Schätzung des Fehlers verwendet. Ausgehend von einer vorhandenen Schätzung des lokalen Fehlers ist es nun möglich die Schrittweite dem Lösungsverlauf anzupassen.

5.1.2 Schrittweitensteuerung

Für die von der gewählten Schrittweite h abhängige Schätzung err des lokalen Fehlers eines Verfahrens der Ordnung p gilt:

$$err \approx Ch^{p+1}. \tag{5.3}$$

Das Ziel ist normalerweise den lokalen Fehler unter einer vorgegebenen Schranke *tol* zu halten. Bei dem hier verwendeten Fehlermaß (5.2) sind die gewünschten Toleranzen ($Atol_i$,$Rtol_i$) schon berücksichtigt, er wird daher mit 1 verglichen.

Die „optimale“ neue Schrittweite h_{neu} wäre somit durch die Bedingung

$$1 \approx C h_{neu}^{p+1} \tag{5.4}$$

definiert. Aus (5.3) und (5.4) ergibt sich die „optimale“ neue Schrittweite durch

$$h_{neu} = h \cdot \sqrt[p+1]{\frac{1}{err}}. \tag{5.5}$$

Daraus lässt sich folgendes Vorgehen zur automatischen Steuerung der Schrittweite ableiten:

> Man führt einen Schritt mit der vorgegebenen Schrittweite h aus und schätzt den entstandenen lokalen Fehler err. Ist dieser unter der vorgegebenen Schranke $err \leq 1$, so wird dieser Schritt akzeptiert und mit der neuen Schrittweite h_{neu} das Verfahren fortgesetzt. Andernfalls wird der Schritt verworfen und noch einmal mit der neuen Schrittweite h_{neu} durchgeführt.

Um sicher zu gehen, dass die neue Schrittweite h_{neu} den lokalen Fehler unter der vorgegebenen Schranke hält, wird in der Praxis die optimale Schrittweite mit einem Sicherheitsfaktor fac (z.B. $fac = 0.8, 0.9, (0.25)^{1/(p+1)}$) multipliziert. Desweiteren sollte sich die Schrittweite weder zu stark vergrößern noch zu stark verkleinern, weshalb zwei weitere Faktoren $facmax$ und $facmin$ zweckmäßig sind. Damit erhält man zur Berechnung der neuen Schrittweite folgende Vorschrift:

$$h_{neu} = h \cdot \min\left(facmax, \max\left(facmin, fac \cdot \frac{1}{err}^{(1/(p+1))}\right)\right) \tag{5.6}$$

Ebenfalls ist es ratsam, nach einem verworfenen Schritt zwei erfolgreiche Schritte abzuwarten, bevor wieder eine Änderung der Schrittweite zugelassen wird.

5.1.3 Dense Output

Im Zuge der Schrittweitensteuerung erhält man die Lösung an Gitterpunkten, die aufgrund der Anforderungen an den lokalen Fehler gewählt wurden. Für die weitere Nutzung der Ergebnisse (z.B. für die graphische Ausgabe) ist die Näherung der Lösung aber oft an vorgegebenen Punkten erwünscht. Würde man diese in der Schrittweitensteuerung berücksichtigen (z.B. als Stützstellen vorgeben), so büßt das Verfahren unter Umständen an Effizienz ein. Eine

Möglichkeit die Vorzüge der Schrittweitensteuerung zu nutzen und dennoch die Lösung an den vorgegebenen Punkten zu erhalten bieten sogenannte *kontinuierliche Runge-Kutta-Formeln.*

Mit einem Parameter $\theta \in (0, 1]$ ermöglichen diese Formeln die Berechnung der Näherungen an beliebigen Zwischenpunkten $t+\theta h$ mit keinen oder nur wenigen zusätzlichen Funktionsauswertungen.

Ausgehend von einer s-stufigen Runge-Kutta-Formel mit den Koeffizienten c_i, a_{ij} und b_j betrachten wir folgende Formeln:

$$v(\theta) = u_h(t) + h \sum_{i=1}^{s^*} b_i(\theta) k_i, \tag{5.7}$$

mit

$$k_i = f(t + c_i\, h\,, u_h(t) + h \sum_{j=1}^{i-1} a_{ij}\, k_j), \quad i = 1, \ldots, s^* \tag{5.8}$$

und den noch zu bestimmenden Polynomen $b_i(\theta)$, die folgende Bedingung erfüllen:

$$v(\theta) - u(t + \theta\, h) = O(h^{p^*+1}). \tag{5.9}$$

Üblicherweise gilt $s^* \geq s + 1$, da zumindest $k_{s+1} = hf(t + h, u_h(t + h))$, also die erste Funktionsauswertung des nachfolgenden Schrittes, mit $a_{s+1,j} = b_j$ für alle j in die Formel miteinbezogen wird.

Ein Runge-Kutta-Verfahren, ausgestattet mit der Formel (5.7), nennt man ein *kontinuierliches Runge-Kutta-Verfahren.*

Die Polynome $b_i(\theta)$ lassen sich mithilfe der Ordnungsbedingungen der Runge-Kutta-Formeln berechnen, hier sei auf [3] verwiesen.

Für die Methode von Dormand und Prince (Tabelle 5.1) wurde eine kontinuierliche Formel der Ordnung 4 konstruiert, die ohne zusätzliche Funktionsauswertungen auskommt $s^* = s$. Aufgrund der Ordnungsbedingungen erhält man für $b_2(\theta) = 0$, $b_7(\theta)$ kann beliebig gewählt werden wodurch die Koeffizienten $b_1(\theta), b_3(\theta), \ldots, b_6(\theta)$ dann eindeutig bestimmt sind. Die von Dormand und Prince gewählte Formel für $b_7(\theta)$ lautet:

$$b_7(\theta) = \theta^2(\theta - 1) + \theta^2(\theta - 1)^2 10 \cdot (7414447 - 829305\,\theta)/29380423.$$

Für die anderen Koeffizienten ergibt sich dadurch:

$$
\begin{aligned}
b_1(\theta) &= \theta^2(3-2\theta)b_1 + \theta(\theta-1)^2 \\
&\quad - \theta^2(\theta-1)^2 5 \cdot (2558722523 - 31403016\,\theta)/11282082432 \\
b_3(\theta) &= \theta^2(3-2\theta)b_3 + \theta^2(\theta-1)^2 100 \cdot (882725551 - 15701508\,\theta)/32700410799 \\
b_4(\theta) &= \theta^2(3-2\theta)b_4 + \theta^2(\theta-1)^2 25 \cdot (443332067 - 31403016\,\theta)/1880347072 \\
b_5(\theta) &= \theta^2(3-2\theta)b_5 + \theta^2(\theta-1)^2 32805 \cdot (23143187 - 3489224\,\theta)/19931678932 \\
b_6(\theta) &= \theta^2(3-2\theta)b_6 + \theta^2(\theta-1)^2 55 \cdot (29972135 - 7076736\,\theta)/822651844
\end{aligned}
$$

In DOPRI5 wird das durch diese Koeffizienten definierte Polynom $v(\theta)$ nur im Mittelpunkt ausgewertet, anstelle es für die kontinuierliche Datenausgabe zu verwenden. Den dadurch erhaltenen Wert $u_{n/2} = v(1/2)$ nutzt man dann mit den Werten u_n, f_n, u_{n+1} und f_{n+1} zur Interpolation durch ein Polynom vom Grad 4. Dies ist einfacher zu implementieren und der Unterschied zur obigen Formel ist nicht signifikant [3].

5.2 Lineare Mehrschrittverfahren (Adams-Verfahren)

Die für uns relevanten Aspekte der praktischen Durchführung von Adams-Verfahren werden basierend auf ihrer Beschreibung in [4] gegeben, welche auch `ode113` zugrundeliegt. Es handelt sich dabei um eine PECE-Adams-Verfahren mit variabler Schrittweite und Ordnung, d.h. sowohl die Schrittweite als auch die Ordnung werden dem Problem angepasst. Wie diese Anpassung erfolgt wird in den nächsten Abschnitten erläutert. Ein weiteres Auswahlkriterium bei der Wahl der Methoden zur Lösung unseres konkreten Problems war die kontinuierliche Datenausgabe, auf sie wird in Abschnitt 5.2.3 eingegangen.

Die Analyse und Beschreibung der Adams-Verfahren in Kapitel 3 und 4 wurde auf einem äquidistanten Gitter durchgeführt, trotz des Hintergedankens, dass für eine effiziente Durchführung der Verfahren eine variable Schrittweite von großer Bedeutung ist. Die Motivation hierfür war, dass PECE-Verfahren mit variabler Schrittweite und Ordnung als Folge von Verfahren mit konstanter Schrittweite mit gelegentlichen Wechsel betrachtet werden können [4]. Für eine genaue Analyse von Adams-Verfahren mit variabler Schrittweite und Ordnung sei auf [3], [4] und [6] verwiesen.

5.2.1 Variable Schrittweite, Variable Ordnung

Indem man die Interpolationspolynome der Adams-Verfahren auf nicht-äquidistante Gitter formuliert, ist es möglich Adams-Verfahren mit variabler Schritt-

weite zu beschreiben.

Betrachten wir eine Folge von beliebigen Gitterpunkten

$$0 = t_0 < t_1 < \ldots < t_m = T$$

mit der Schrittweite

$$h_n = t_{n+1} - t_n.$$

Wir setzen wiederum voraus, dass die Werte $u_j \approx u(t_j)$ für $j = n-k+1, \ldots, n$ bekannt sind und setzen $f_j = f(t_j, u_j)$. Zur Formulierung der Adams-Bashforth-Methode benutzen wir Newtons Interpolationsformel, um das Interpolationspolynom $p(t)$ durch die Werte (t_j, f_j) für $j = n-k+1, \ldots, n$ darzustellen:

$$p(t) = \sum_{j=0}^{k-1} \prod_{i=0}^{j-1} (t - t_{n-i}) \delta^j f[t_n, t_{n-1}, \ldots, t_{n-j}] \tag{5.10}$$

mit den dividierenden Differenzen $\delta^j f[t_n, \ldots, t_{n-j}]$, die rekursiv definiert sind über

$$\begin{aligned} \delta^0 f[t_n] &= f_n \\ \delta^j f[t_n, \ldots, t_{n-j}] &= \frac{\delta^{j-1} f[t_n, \ldots, t_{n-j+1}] - \delta^{j-1} f[t_{n-1}, \ldots, t_{n-j}]}{t_n - t_{n-j}}. \end{aligned} \tag{5.11}$$

Für konstante Schrittweiten vereinfachen sich die dividierenden Differenzen zu Rückwärtsdifferenzen (3.47).

Eine für die Berechnung der Koeffizienten sinnvolle Formulierung des Polynoms ist:

$$p(t) = \sum_{j=0}^{k-1} \prod_{i=0}^{j-1} \frac{t - t_{n-i}}{t_{n+1} - t_{n-i}} \cdot \Phi_j^*(n) \tag{5.12}$$

mit

$$\Phi_j^*(n) = \prod_{i=0}^{j-1} (t_{n+1} - t_{n-i}) \cdot \delta^j f[t_n, \ldots, t_{n-j}]. \tag{5.13}$$

Die Näherungslösung für $u(t_{n+1})$ ist dann definiert als

$$u_{n+1} = u_n + \int_{t_n}^{t_{n+1}} p(\tau)\, d\tau. \tag{5.14}$$

Setzt man die Darstellung (5.12) des Polynoms in (5.14) ein, so erhält man

$$u_{n+1} = u_n + h_n \sum_{j=0}^{k-1} g_j(n) \Phi_j^*(n) \tag{5.15}$$

mit

$$g_j(n) = \frac{1}{h_n} \int_{t_n}^{t_{n+1}} \prod_{i=0}^{j-1} \frac{\tau - t_{n-i}}{t_{n+1} - t_{n-i}} \, d\tau. \tag{5.16}$$

Die Adams-Moulton-Methode mit variabler Schrittweite lässt sich auf ähnliche Weise herleiten. Wie in Abschnitt 3.3.1 sei $p^*(t)$ das Polynom vom Grad k, das die Werte (t_j, f_j) für $j = n - k + 1 \ldots, n + 1$ interpoliert. Mit der Newton-Darstellung ergibt sich

$$p^*(t) = p(t) + \prod_{i=0}^{j-1} (t - t_{n-i}) \delta^k f[t_{n+1}, t_n, \ldots, t_{n-k+1}]. \tag{5.17}$$

Die Näherungslösung

$$u_{n+1} = u_n + \int_{t_n}^{t_{n+1}} p^*(\tau) \, d\tau \tag{5.18}$$

lässt sich damit folgendermaßen schreiben

$$u_{n+1} = p_{n+1} + h_n g_k(n) \Phi_k(n+1), \tag{5.19}$$

wobei p_{n+1} die numerische Lösung der Adams-Bashforth-Verfahren (5.15) ist und

$$\Phi_k(n+1) = \prod_{i=0}^{k-1} (t_{n+1} - t_{n-i}) \delta^k f[t_{n+1}, t_n, \ldots, t_{n-k+1}]. \tag{5.20}$$

Bei Anwendung eines PECE-Verfahren ist $\delta^0 f[t_{n+1}] = f(t_{n+1}, p_{n+1})$ in Formel (5.20).

Der größte Aufwand bei der Anwendung dieser Formeln ist die Berechnung der Intergrations-Koeffizienten $g_j(n)$, $\Phi_j(n)$ und $\Phi_j^*(n)$. Damit der Vorteil der Adams-Verfahren, wenige Funktionsauswertungen für ein genaues Ergebnis zu benötigen, erhalten bleibt, ist die effiziente Berechnung dieser Koeffizienten wichtig. Wie diese Berechnung im Detail aussieht, können interessierte Leser in [3] und [4] nachschlagen.

Die Ordnung der Adams-Verfahren hängt von der Anzahl k der Vorgänger ab. Um die Ordnung zu variieren muss man also lediglich Interpolationspunkte hinzunehmen oder weglassen. Da die Koeffizienten $g_j(n)$, $\Phi_j(n)$ und $\Phi_j^*(n)$ unabhängig von k sind, ist somit eine Änderung der Ordnung ohne großen Aufwand möglich.

Mit obigen Formeln steht uns eine Erweiterung der Adams-Verfahren auf nicht-äquidistante Gitter zur Verfügung. Um nun automatische Schrittweitensteuerung zu bewerkstelligen, benötigen wir eine Schätzung des lokalen Fehlers.

5.2.2 Schrittweitensteuerung und Auswahl der Ordnung

Wir nehmen wiederum an, dass die ersten Näherungen erfolgreich bis t_n berechnet wurden und ein weiterer Schritt mit der Schrittweite h_n und der Ordnung $k+1$ durchgeführt wird. Somit erhalten wir die Näherung $u_{n+1} \approx u(t_{n+1})$. Um zu entscheiden, ob die Näherung u_{n+1} akzeptiert wird, benötigen wir eine Schätzung für den lokalen Fehler. Eine mögliche Schätzung des lokalen Fehlers ist durch

$$le_{k+1}(n+1) = \tilde{u}_{n+1} - u_{n+1} \tag{5.21}$$

gegeben, wobei $\tilde{u}_{n+1}$ die Näherungslösung eines Adams-Moulton-Verfahren der Ordnung $k+2$ ist, also:

$$\tilde{u}_{n+1} = \tilde{p}_{n+1} + h_n g_{k+1}(n)\Phi_{k+1}(n+1) \tag{5.22}$$

$$\tilde{p}_{n+1} = u_n + h_n \sum_{j=0}^{k} g_j(n)\Phi_j^*(n). \tag{5.23}$$

Für u_{n+1} gilt folgende Gleichung:

$$u_{n+1} = p_{n+1} + h_n g_k(n)\Phi_k(n+1) \tag{5.24}$$

$$p_{n+1} = u_n + h_n \sum_{j=0}^{k-1} g_j(n)\Phi_j^*(n). \tag{5.25}$$

Die Differenz der beiden Näherungslösungen ergibt somit:

$$\tilde{u}_{n+1} - u_{n+1} = h_n \left(g_k(n)\Phi_k^*(n) + g_{k+1}(n)\Phi_{k+1}(n+1) - g_k(n)\Phi_k(n+1)\right). \tag{5.26}$$

Aufgrund der Beziehung

$$\Phi_{j+1}(n) = \Phi_j(n) - \Phi_j^*(n+1)$$

folgt für die Schätzung des lokalen Fehlers

$$le_{k+1}(n+1) = h_n(g_{k+1}(n) - g_k(n))\Phi_{k+1}(n+1). \tag{5.27}$$

Ersetzt man in $\Phi_{k+1}(n+1)$ den Ausdruck $f_{n+1} = f(t_{n+1}, u_{n+1})$ durch $f_{n+1} = f(t_{n+1}, p_{n+1})$, was auf $\Phi_{k+1}^p(n+1)$ führt, so erhält man die Schätzung für den lokalen Fehler des Prädiktor-Korrektor Verfahrens:

$$LE_{k+1}(n+1) = h_n(g_{k+1}(n) - g_k(n))\Phi_{k+1}^p(n+1). \tag{5.28}$$

Der einzige Aufwand bei der Schätzung des Fehlers ist die Berechnung von $g_{k+1}(n)$, da sowohl $g_k(n)$ wie auch $\Phi_{k+1}^p(n+1)$ schon zur Berechnung der Näherung benötigt wurden.

Ausgehend von dieser Schätzung des lokalen Fehlers wird der Schritt akzeptiert, falls

$$\|LE_{k+1}(n+1)\| \leq 1. \tag{5.29}$$

Die dabei verwendete Norm entspricht jener, die für den lokalen Fehler der Runge-Kutta-Verfahren (5.2) verwendet wurde .

Unter der Annahme, dass u_{n+1} akzeptiert wurde, müssen nun eine neue Schrittweite und Ordnung gewählt werden. Die Idee zur Auswahl der Schrittweite liegt darin, die größte Schrittweite h_{n+1} zu finden, sodass der geschätzte lokale Fehler die Ungleichung (5.28) erfüllt, also:

$$h_{n+1} \left|g_{k+1}(n+1) - g_k(n+1)\right| \left\|\Phi^p_{k+1}(n+2)\right\| \leq 1. \tag{5.30}$$

Diese Prozedur ist allerdings in der Praxis unbrauchbar, da die Werte $g_j(n+1)$ und $\Phi^p_{k+1}(n+2)$ auf komplizierte Art von der neuen unbekannten Schrittweite h_{n+1} abhängen. Um diese Schwierigkeit zu umgehen, nimmt man das Gitter als äquidistant an. Dadurch lässt sich, wie bei den Runge-Kutta-Verfahren, der Fehler im nächsten Schritt durch den vorangegangen abschätzen und es ergibt sich für die optimale Schrittweite

$$h^{(k+1)}_{opt} = h_n \left(\frac{1}{\|LE_{k+1}(n+1)\|}\right)^{1/(k+1)}. \tag{5.31}$$

Dabei ist der lokale Fehler entweder durch (5.28) gegeben, oder unter Annahme eines äquidistanten Gitters durch

$$LE_{k+1}(n+1) = h_n \gamma^*_{k+1} \Phi^p_{k+1}(n+1). \tag{5.32}$$

Für die Wahl der nächsten Ordnung gibt es zwei Strategien, entweder man wählt jene, für die der geschätzte Fehler minimal oder jene, für die die neue optimale Schrittweite maximal wird. Im Falle des hier zugrundeliegendem Adams-PECE-Verfahren erfolgt die Auswahl der Ordnung auf folgende Weise:

Nach der Ausführung eines Schrittes der Ordnung $k+1$, berechnet man $LE_{k-1}(n+1)$, $LE_k(n+1)$ und $LE_{k+1}(n+1)$ wie in (5.32). Die Ordnung wird nun um eins reduziert, falls

$$\max\left(\|LE_{k-1}(n+1)\|, \|LE_k(n+1)\|\right) \leq \|LE_{k+1}(n+1)\|. \tag{5.33}$$

Die Erhöhung der Ordnung wird nur in Betracht gezogen, falls der Schritt akzeptiert wurde, Ungleichung (5.33) nicht erfüllt ist und eine konstante Schrittweite benutzt wird. In dem Fall wird

$$LE_{k+2}(n+1) = h_n \gamma^*_{k+2} \Phi^p_{k+2}(n+1) \tag{5.34}$$

unter Verwendung von $f_{n+1} = f(t_{n+1}, u_{n+1})$ berechnet. Die Ordnung wird um eins erhöht, falls

$$\|LE_{k+2}(n+1)\| < \|LE_{k+1}(n+1)\|. \tag{5.35}$$

5.2.3 Dense Output

Aufgrund der automatischen Schrittweitensteuerung versucht das Verfahren möglichst große Schritte zu wählen und dabei den lokalen Fehler in einer gewissen Toleranz zu halten, dementsprechend wenig Punkte werden somit tatsächlich ausgewertet. Wünscht man die Ergebnisse an vorgegebenen Punkten, so müssen diese interpoliert werden. Da bei der numerischen Lösung mit Adams-Verfahren ohnehin bereits ein Interpolationspolynom benutzt wird, lässt sich zeigen, dass die Berechnung von $u_h(t_{out})$ an Zwischenpunkten $t_n < t_{out} \leq t_{n+1}$ so genau ist, wie die Näherungen u_n und u_{n+1} an den Gitterpunkten. Dabei definiert man die Näherung $u_I(t)$ von $u(t)$ auf $(t_n, t_{n+1}]$ als

$$u_I(t) = u_{n+1} + \int_{t_{n+1}}^{t} p(\tau)d\tau, \tag{5.36}$$

wobei $p(t)$ als Polynom durch die Punkte (t_{n+1-i}, f_{n+1-i}) für $i = 0, \ldots, k$ definiert ist. Für variable Schrittweiten ist $p(t)$ durch die folgende Formel gegeben.

$$p(t) = \sum_{j=0}^{k} \prod_{i=0}^{j-1} (t - t_{n-i}) \delta^j f[t_{n+1}, t_n, \ldots, t_{n-j}]. \tag{5.37}$$

Analog zur Herleitung der Adams-Verfahren auf nicht-äquidistantem Gitter erhält man somit für die Nährung $u_I(t_{out})$:

$$u_I(t_{out}) = u_{n+1} + h_I \sum_{i=0}^{k} g_i^I(n)\phi_i(n+1), \tag{5.38}$$

mit $h_I = t_{out} - t_{n+1}$ und

$$g_j^I(n) = \frac{1}{h_I} \int_{t_{n+1}}^{t_{out}} \prod_{i=0}^{j-1} \frac{\tau - t_{n-i}}{t_{n+1} - t_{n-i}} \, d\tau. \tag{5.39}$$

Auch wenn man zur Berechnung der Näherung an den gewünschten Punkten stets die Koeffizienten $g_i^I(n)$ berechnen muss, so benötigt diese kontinuierliche Datenausgabe keine zusätzlichen Funktionsauswertungen und die Schrittweitensteuerung wird nicht beeinflusst.
Für Details siehe [4].

5.3 Bemerkungen zu semi-explizite DAEs vom Index 1

Sowohl die Schrittweitensteuerung wie auch die kontinuierliche Datenausgabe beider Klassen von Verfahren nehmen bei der Anwendung auf semi-explizite

DAEs vom Index 1 keine Rücksicht auf die algebraischen Größen. Zur Schätzung des lokalen Fehlers und der damit verbundenen Wahl der nächsten Schrittweite werden lediglich die differentiellen Größen herangezogen.

Da jedoch die Funktion G laut Voraussetzung die Lipschitzbedingung erfüllt

$$\|G(t,v) - G(t,w)\| \leq L_G \|v - w\|,$$

lässt sich der Fehler in den algebraischen Größen z durch die differentiellen Größen y abschätzen.

Eine tatsächliche Einschränkung tritt allerdings bei der kontinuierlichen Datenausgabe auf. Die dabei verwendeten Formeln beschränken sich auf die Auswertung von $y_{out} = y(t_{out})$ an den gewünschten Zwischenpunkten t_{out}. Für unser Problem würde die Auswertung von $z_{out} = G(t_{out}, y_{out})$ an jeden dieser Zwischenpunkte wenig Sinn machen, da diese Auswertung eine zeitintensive Berechnung darstellt und somit die Beschleunigung der transiente Rechnung aufgrund der Veringerung der Anzahl der Funktionauswertungen wieder verloren wäre. Um dennoch Näherungen $z_{out} = z(t_{out})$ zu erhalten, muss also auf geeignete Weise interpoliert werden.

Kapitel 6

Numerische Experimente

Anhand eines einfachen, aber doch aussagekräftigen Modells eines Kühlsystems in KULI wurde die ursprüngliche Berechnungsvorschrift (siehe Abschnitt 6.2) mit der neuen (siehe Abschnitt 6.3) verglichen, wobei zur Ausführung der neuen Berechnungsvorschrift verschiedene Verfahren verwendet wurden. Da die Geschwindigkeit der Verfahren hauptsächlich von der Anzahl der stationären Abgleiche (Auswertung von $z(t) = G(t, y(t))$) abhängt, wurde diese, neben der grob gemessenen Zeit, als Vergleichsgröße herangezogen. Dabei wurde versucht, die absoluten Fehler der verschiedenen Verfahren im selben Bereich zu halten. Um die Verfahren vergleichen zu können, wurde das Problem einmal mit KULI 7.0 und einer extrem kleinen Schrittweite gerechnet und die dadurch erhaltene Lösung als exakte Lösung verwendet. Der absolute Fehler ergibt sich somit als Differenz der Temperaturen der Punktmassen, die man als Näherungslösung der getesteten Verfahren erhält, zu den Temperaturen der Punktmassen, die sich bei der Lösung des Systems mit KULI 7.0 und einer sehr kleinen Schrittweite ergeben.
Die getesteten Verfahren waren `imsl_d_ode_runge_kutta`, die MATLAB- Funktionen `ode45` und `ode113` und eine in C implementierte Version von `DOPRI5`.
Bei der ursprünglichen Berechnungsvorschrift wurde zur Lösung der Differentialgleichung bereits `imsl_d_ode_runge_kutta`, ein 8-stufiges Runge-Kutta-Verner Verfahren von der Ordnung 5(6) aus der imsl-Bibliothek, verwendet, weshalb es nahe stand, diese Routine auch bei der neuen Berechnungsvorschrift zu verwenden. Für den Vergleich von Runge-Kutta- und Mehrschrittverfahren dienten die Matlab-Funktionen `ode45` und `ode113`. Ersteres beruht auf `DOPRI5` (siehe [3]), und `ode113` ist ein Adams-Bashforth-Moulton PECE-Verfahren mit variabler Ordnung und Schrittweite. Aufgrund der guten Ergebnisse von `ode45` und der Tatsache, dass `imsl_d_ode_runge_kutta` nicht über die kontinuierliche Datenausgabe verfügt, wurde zusätzlich eine in C implementierte Version von `DOPRI5` getestet.

6.1 Beschreibung des Modells

Abbildung 6.1 zeigt ein Beispiel mit Luftseite, Wasser- und Ölkreislauf. Die Luftseite simuliert dabei die Durchströmung des Fahrzeugs. In Abbildung 6.1(a) strömt die Luft links durch verschiedene Widerstände ein, verzweigt sich dann, strömt durch einen Wasserkühler, zwei Lüfter und weitere Widerstände bis sie wieder an die Umgebung abgegeben wird. Im Wasserkühler findet dabei ein Wärmetransport statt, die Kühlflüssigkeit im Wasserkreislauf (Abbildung 6.1(b) 1.WK [I]) gibt Wärme an die kältere Luft ab. Wasser- und Ölkreislauf beeinflussen sich wiederum durch einen Parallelstromwärmetauscher (1.PWT [A] in 6.1(b) und 1.PWT [I] in 6.1(c)). In dem KF- und dem Oel-Symbol kann man Massenstrom, Druck und zugeführte Wärme des jeweiligen Kreislaufs vorgeben. Die Komponenten *Wasser* und *Öl* sind sogenannte Punktmassen (siehe Abschnitt 2.1).

Punktmassen	Wasser	Öl
Masse [kg]	5	3
Wärmekap. [J/kg/K]	4190	1000
max. Wärmeübergangsfläche [m^2]	1	1
Wärmeübergangskoeffizient [W/m^2K]	1e+9	1e+6
Starttemperatur [°C]	20	20

Tabelle 6.1: Parameter der Punktmassen

In Tabelle 6.1 stehen die Eingabeparameter der beiden Punktmassen der Fluidkreisläufe. Beide haben einen hohen Wärmeübergangskoeffizienten, was auf einen schnellen Wärmeübergang des Mediums auf die Punktmasse schließen lässt. Die Simulationsparameter sind in Tabelle 6.2 angegeben.

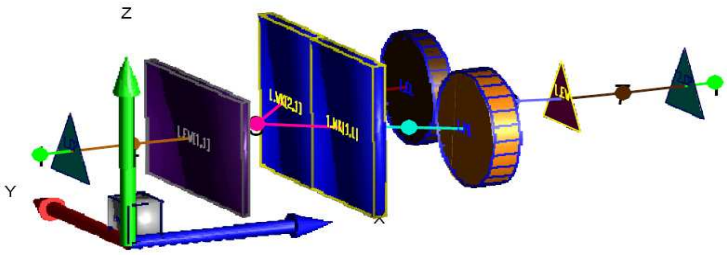
6.2 Berechnung mit Kuli 7.0

Die ursprüngliche Idee zur Lösung des Systems, das auf Problem 2.1 führt, also eine semi-explizite DAE vom Index 1, war die Vorgabe der Schrittweite h mit $h = t_{i+1} - t_i$ und abwechselndes Lösen der Gleichungen (2.7a) und (2.7b),

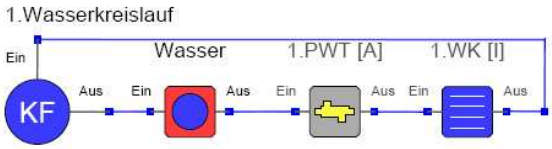
$$y'(t) = f(t, y(t), z(t)), \tag{2.7a}$$
$$0 = g(t, y(t), z(t)). \tag{2.7b}$$

Im ersten Schritt wird daher

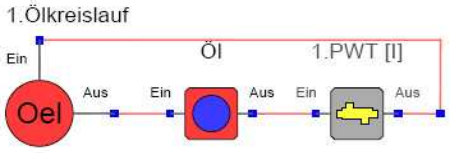
$$0 = g(t_0, y_0, z_0)$$



(a) Luftseite



(b) Innerer Wasserkreislauf



(c) Innerer Ölkreislauf

Abbildung 6.1: Modell eines Kühlsystems

Zeit [s]	Motordrehzahl [U/min]	pme [bar]	Fahrgeschw. [km/h]	E-Lüfter Stufe
0	2650	5.9	40	3
10	2480	8.3	105	2
25	2480	8.3	105	2
30	3970	9.1	200	1
290	3970	9.1	200	1
300	1500	2	10	3
400	1500	2	10	3

Tabelle 6.2: Simulationsparameter: In 30 Sekunden wird stark beschleunigt, was eine hohe Wärmezufuhr im Wasserkreislauf bedeutet. Nach 290 Sekunden wird stark abgebremst, was einerseits weniger Wärmezufur bedeutet aber andererseits auch einen geringeren Wärmeabtransport durch die Luftseite.

nach $z_0 \approx z(t_0)$ aufgelöst. Also ein stationärer Abgleich mit fixer Punktmassentemperatur gerechnet. Mit den berechneten Zuständen z_0 wird dann aus Gleichung (2.7a) $y_1 \approx y(t_1)$, durch Lösen der Anfangswertaufgabe

$$\begin{aligned} y'(t) &= f(t, y(t), z_0) \\ y(t_0) &= y_0 \end{aligned}$$

im Intervall $[\,t_0,\, t_1\,]$ mit einem Runge-Kutta-Verfahren berechnet, in diesem Fall `imsl_d_ode_runge_kutta`.

In den Abbildungen 6.2 und 6.3 sieht man die über den Zeitintervall $[\,0,\, 400\,]$ in 10, 5, 1 und 0.1 Sekunden Schritten berechneten Temperaturverläufe der Punktmassen *Wasser* und *Öl*. Man erkennt deutlich die zu Beginn starke Erwärmung, verursacht durch die Beschleunigung des Fahrzeugs, und den Einfluss des plötzlichen Abbremsens nach 290 Sekunden. Die großen Wärmeübergangskoeffizienten der Punktmassen (vgl. Tabelle 6.1) bewirken, dass die Punktmassen relativ schnell die Temperaturen der umströmenden Fluide annehmen und sich damit ab einem bestimmten Zeitpunkt nicht mehr weiter erwärmen. Dies hat zur Folge, dass bei zu groß gewählten Schrittweiten das Ergebnis nicht zufriedenstellend ist. Erst ab einer Schrittweite von 0,1 Sekunden ist das Ergebnis bei diesem Beispiel hinreichend genau. Abbildung 6.4 zeigt den Fehler der Temperaturen der beiden Punktmasse *Wasser* und *Öl* bei einer Schrittweite von $0,1s$, der Fehler ist dabei kleiner als 0.4. Für ein Ergebnis dieser Genauigkeit sind 4000 stationäre Abgleiche bzw ca. 1590 CPU-Sekunden nötig.

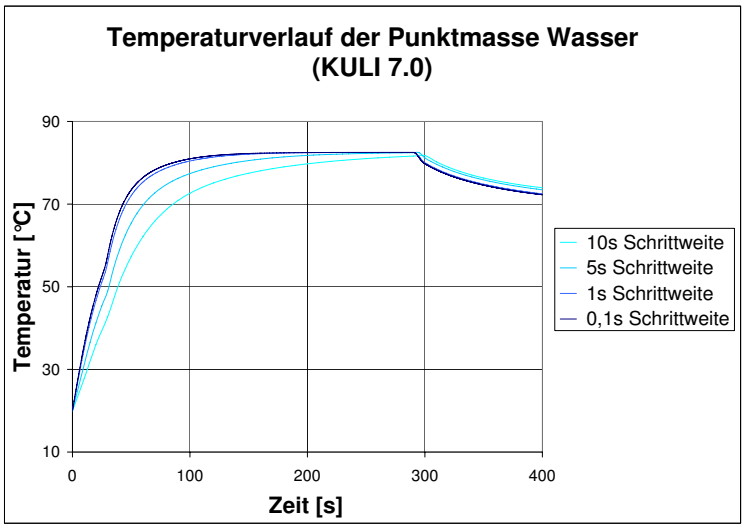


Abbildung 6.2: Temperaturverlauf der Punktmasse *Wasser*

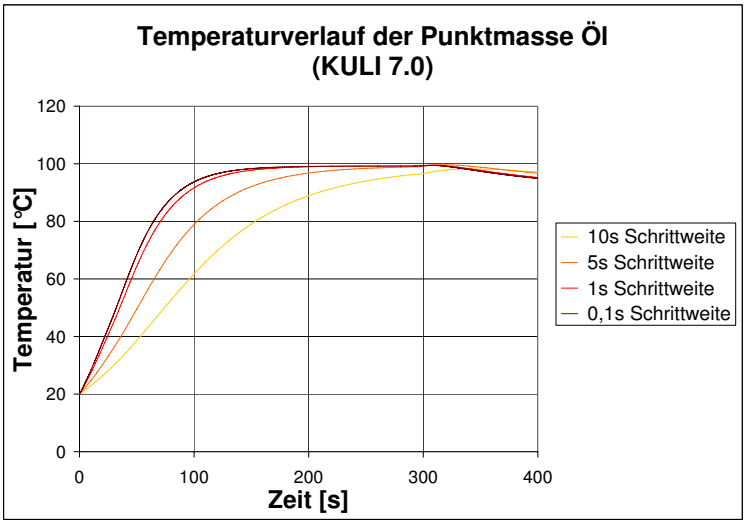


Abbildung 6.3: Temperaturverlauf der Punktmasse *Öl*

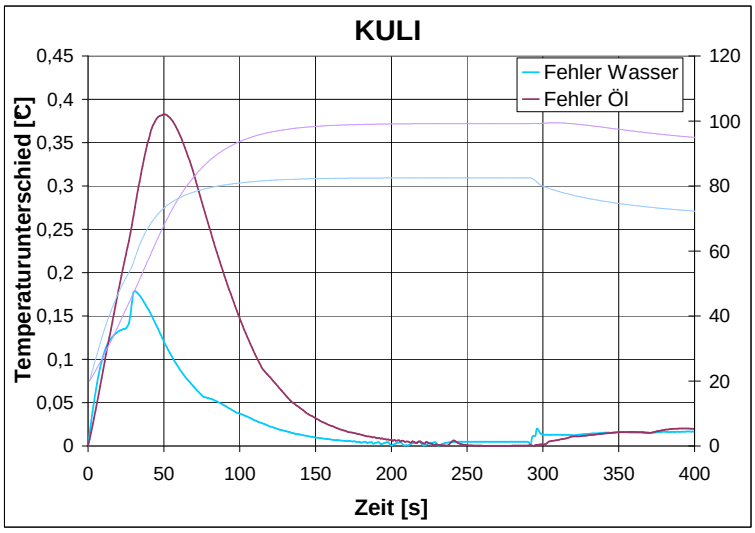


Abbildung 6.4: Fehler (fett) und Temperaturverläufe der beiden Punktmassen mit Kuli 7.0 berechnet.

6.3 Berechnung als semi-explizite DAEs vom Index 1

Die neue Berechnungsvorschrift beruht auf denen in Kapitel 3 vorgestellten Verfahren zur Lösung von semi-expliziten DAEs vom Index 1 mithilfe des indirekten Zugangs.

6.3.1 Explizite Runge-Kutta-Verfahren

imsl_d_ode_runge_kutta: Ein getestetes Verfahren ist ein 8-stufiges explizites Runge-Kutta-Verner Verfahren der Ordnung 5(6) aus der imsl - Bibliothek, namens `imsl_d_ode_runge_kutta`. Es verfügt über automatische Schrittweitensteuerung und ermöglicht dem Benutzer die Einstellung verschiedener Parameter, wie beispielsweise Fehlertoleranz, maximale und minimale Schrittweite, (für eine detaillierte Beschreibung siehe [7]).
Da es 8-stufig ist, braucht es zu Auswertung einer Stützstelle mindesten 8 Funktionsaufrufe, was in unserem Fall 8 stationäre Abgleiche bedeutet. Allerdings lässt die Schrittweitensteuerung aufgrund der hohen Ordnung auch relative große Schrittweiten zu.
Das Problem hierbei ist, dass `imsl_d_ode_runge_kutta` nicht über kontinuierliche Datenausgabe verfügt, die Ergebnisse stehen somit nur an den berech-

neten Gitterpunkten zur Verfügung, die aufgrund der hohen Ordnung, möglicherweise weit auseinanderliegen. Um Zwischenwerte zu erhalten, muss man daher auf herkömmliche Weise interpolieren und verliert damit an Genauigkeit.
Das Modellproblem wurde mit einer Fehlertoleranz von 10^{-5} über den gesamten Zeitintervall gerechnet. Dafür benötigt die Routine 400 stationäre Abgleiche und ca. 290 CPU-Sekunden. Gemessen an der Anzahl der stationären Abgleiche bedeut dies eine Beschleunigung um den Faktor 10 im Vergleich zur ursprünglichen Berechnungsvorschrift. Um die Lösung in Sekundenschritten zu erhalten, wurden die Ergebnisse an den von `imsl_d_ode_runge_kutta` gewählten Gitterpunkten mit MATLAB (spline) interpoliert. In Abbildung 6.5 ist der dabei entstandene Fehler dargestellt.

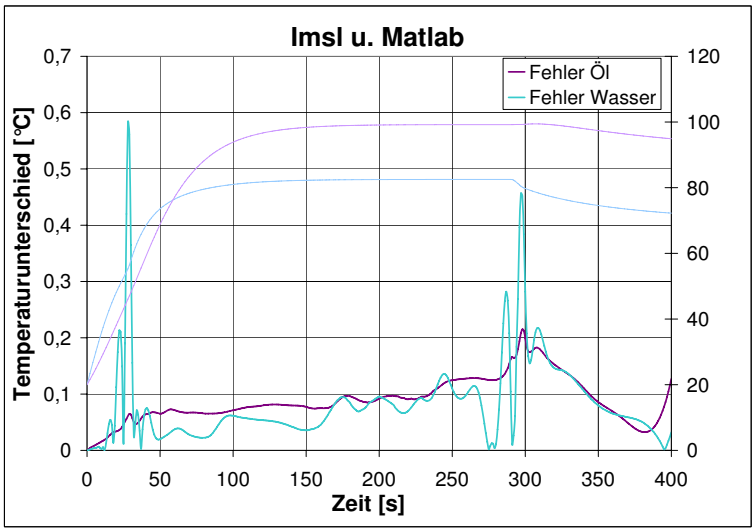


Abbildung 6.5: Fehler (fett) und Temperaturverläufe der beiden Punktmassen mit imsl-Funktion berechnet und MATLAB interpoliert.

ode45: Die MATLAB-Funktion `ode45` beruht auf dem in Kapitel 5 vorgestellten eingebetteten kontinuierlichen Runge-Kutta-Verfahren namens `DOPRI5` aus [3]. Allerdings werden zur kontinuierlichen Datenausgabe andere Parameter verwendet, was die unterschiedlichen Ergebnisse der beiden Verfahren erklärt. In Abbildung 6.6 ist der absolute Fehler der beiden Punktmassentemperaturen dargestellt, den man aus der Berechnung mit `ode45` und der Vorgabe der relativen Fehlertoleranz mit 10^{-5} und der absoluten Fehlertoleranz mit 10^{-3} erhält. Dafür sind 337 Funktionsaufrufe und ca. 170 CPU-Sekunden nötig. Die Berechnung mit diesem Verfahren ist also 11,8 mal schneller als die ursprüngliche Berechnungsvorschrift.

Die Funktion `ode45` war aufgrund der langsamen Schnittstelle zwischen Matlab und Kuli nur für Testzwecke gedacht und geeignet. Da sie jedoch gute Resultate lieferte und über die wünschenswerte Eigenschaft der kontinuierlichen Datenausgabe verfügt, wurde ein weiteres Verfahren implementiert und getestet, eine in C geschriebene Version von `DOPRI5`.

Abbildung 6.6: Fehler (fett) und Temperaturverläufe der beiden Punktmassen mit ode45.

DOPRI5: Der hier verwendete Code von `DOPRI5` ist in C programmiert und beruht zur Gänze auf dem in [3] beschrieben Verfahren, beide Codes sind auf [8] verfügbar. Das beste Ergebnis, also jenes mit den kleinsten Fehlern, erhält man bei einer relativen Fehlertoleranz von 10^{-5} und einer aboluten von 10^{-4}. Die Fehler der Temperaturen der beiden Punktmassen sind in Abbildung 6.7 dargestellt. Die ungefähr benötigt Zeit zur Berechnung liegt bei 155 CPU-Sekunden, bzw. 340 Funktionsauswertungen, was wiederum eine Beschleunigung der Berechnung um den Faktor 11,8 bedeutet.

6.3.2 Lineare Mehrschrittverfahren

ode113: Die Matlab-Funktion `ode113` ist ein Adams-Bashforth-Moulton PECE Verfahren und beruht auf der Beschreibung aus [4]. In Abbildung 6.8 ist der berechnete Fehler der Punktmassentemperaturen dargestellt, wenn man für die relative Fehlertoleranz 10^{-5} und für die absolute 10^{-3} wählt. Die dabei benötigte Anzahl der Funktionsaufrufe, nämlich 290, ist noch einmal um eine

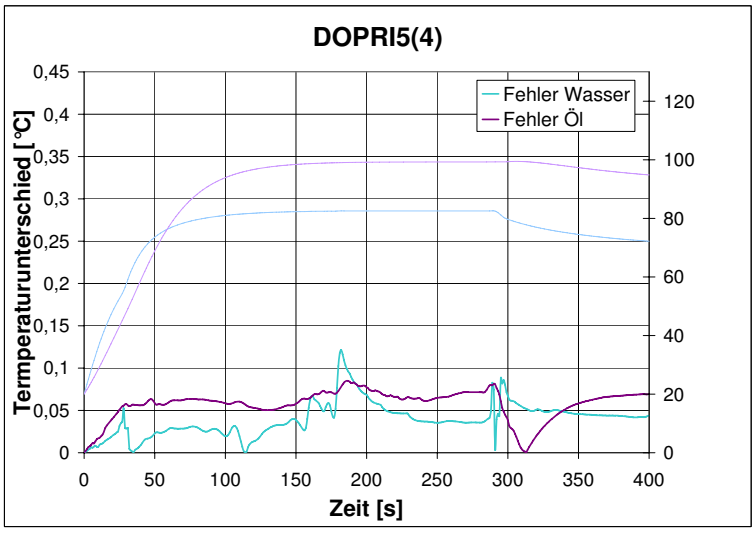


Abbildung 6.7: Fehler (fett) und Temperaturverläufe der beiden Punktmassen mit `DOPRI5`.

Spur geringer als für `ode45` und `DOPRI5`, was mit der Theorie der Mehrschrittverfahren übereinstimmt.

6.4 Schlussfolgerungen

In Abbildung 6.9 ist der Mittelwert der berechneten absoluten Fehler der Verfahren in den kritischen Bereichen, d.h. für $t \in [0, 100]$ und $t \in [250, 350]$, also den Bereichen in denen stark beschleunigt und stark abgebremst wird, dargestellt. Bei ähnlichen Fehlergrößen ergibt sich ein signifikanter Unterschied in der Anzahl der benötigten stationären Abgleiche zwischen der ursprünglichen Berechnungsvorschrift und jener als semi-explizite DAEs vom Index 1. So waren bei der ursprünglichen Berechnungsvorschrift 4000 stationäre Abgleiche nötig, um den Fehler in dem gewünschten Bereich zu halten. Durch die Lösung des Problems als semi-explizite DAEs vom Index 1 benötigte `imsl_d_ode_runge_kutta` 400 stationäre Abgleiche, `ode45` 337, `DOPRI5` 338 und `ode113` 290. Das bedeutet eine Beschleunigung der Berechnung um mindestens den Faktor 10.
Die unterschiedlichen Genauigkeiten zwischen `ode45` und `DOPRI5` ergeben sich aus den unterschiedlichen Algorithmen für die kontinuierliche Datenausgabe.
Eine weitere Verbesserung der Ergebnisse, das bedeutet sowohl eine genauere Lösung, wie auch eine geringere Anzahl an Funktionsauswertungen erkennt man bei der Verwendung von `ode113`, also eine Mehrschrittverfahren, im Vergleich zu den Runge-Kutta-Verfahren.

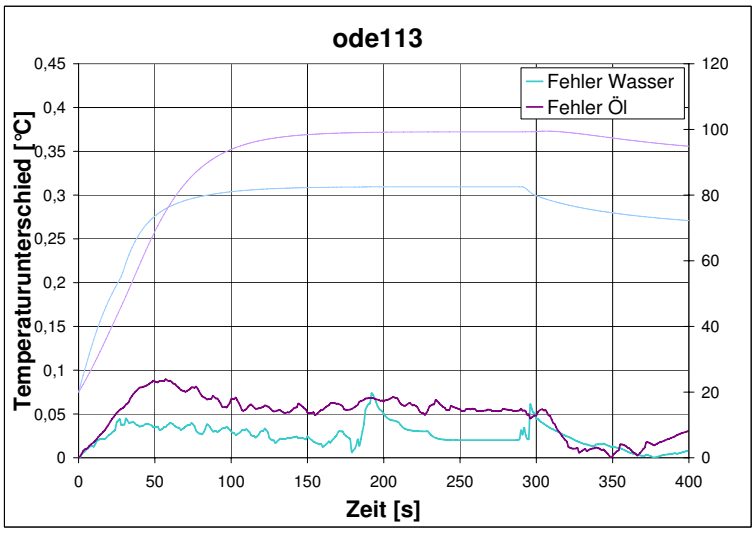


Abbildung 6.8: Fehler (fett) und Temperaturverläufe der beiden Punktmassen mit ode113.

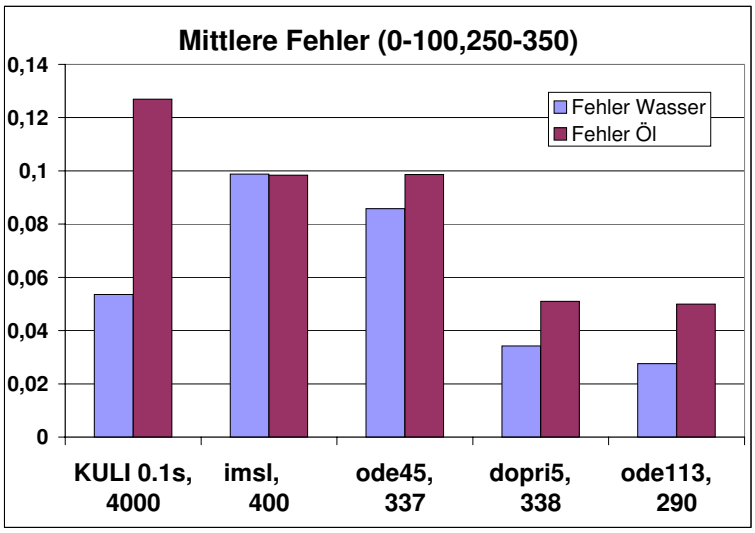


Abbildung 6.9: Mittlere Fehler aller getesteten Verfahren

Literaturverzeichnis

[1] Kathryn Eleda Brenan, Stephen Lavern Campbell, and Linda Ruth Petzold. *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*, volume 14 of *Classics in Applied Mathematics*.

[2] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics. Springer Verlag, second revised edition, December 1996.

[3] Ernst Hairer, Syvert Paul Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Series in Computational Mathematics. Springer Verlag, second revised edition, January 1993.

[4] Lawrence F. Shampine and Marilyn Kay Gordon. *Computer Solution of Ordinary Differential Equations, The Initial Value Problem*. W.H. Freeman and Company, 1975.

[5] Ulrich Langer. *Numerik 3: Numerische Verfahren für Anfangs- und Anfangsrandwertaufgaben*. Insitut für Mathematik, Johannes Kepler Universität Linz, Februar 1997.

[6] Lawrence F. Shampine. Variable order adams codes. In *Computers & Mathematics with Applications*, volume 44, pages 749–761. Elsevier Science, September 2002.

[7] *IMSL C Numerical Library User's Guide*.

[8] http://www.unige.ch/ hairer/software.html.