

Seminararbeit

Integrationsverfahren zur Simulation im Zeitbereich

Bearbeiter: Simon Kerschbaum
21445549

Betreuer: Anatoli Semerow

Abgabedatum: 13.12.2013



Ich versichere, dass ich die vorliegende Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Ort, Datum

Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis	V
Tabellenverzeichnis	VI
1 Einleitung	1
2 Numerische Integrationsverfahren für Anfangswertprobleme	2
2.1 Einschrittverfahren	3
2.1.1 Explizites Eulerverfahren	3
2.1.2 Implizites Eulerverfahren	4
2.1.3 Trapezregel/Heun-Verfahren	4
2.1.4 Runge-Kutta-Verfahren	5
2.2 Schrittweitensteuerung (Adaption)	7
2.3 Eingebettete Runge-Kutta-Verfahren	9
2.4 Mehrschrittverfahren	10
2.4.1 Adams-Verfahren/Prädiktor-Korrektor-Verfahren	11
2.4.2 BDF-Verfahren	12
3 Stabilitätsanalyse der wichtigsten Methoden	13
3.1 Steife Differentialgleichungen	13
3.2 Einschrittverfahren	14
3.3 Mehrschrittverfahren	18
4 Anwendung der Integrationsmethoden in Simulationssoftware	20
5 Zusammenfassung und Ausblick	22
A Überblick einiger Verfahren	23
A.1 Einschrittverfahren	24
A.2 Eingebettete Runge-Kutta-Verfahren (Schrittweitensteuerung)	25
A.3 Mehrschrittverfahren	27
Literaturverzeichnis	28

Abbildungsverzeichnis

3.1	Simulation des Beispielsystems mit explizitem Euler-Verfahren	15
3.2	Stabilitätsgebiete für einige Einschrittverfahren	18

Tabellenverzeichnis

2.1	Allgemeines Butcher-Schema expliziter Runge-Kutta-Verfahren	6
2.2	Butcher-Schemata für (von links nach rechts): Euler-Verfahren, Heun-Verfahren, <i>klassisches</i> Runge-Kutta-Verfahren	7
2.3	Butcher-Schemata für implizite Verfahren (von links nach rechts): Euler, Mittelpunktsregel, Trapezregel	7
2.4	allgemeines Butcher-Schema für eingebettete Runge-Kutta-Verfahren . .	9
2.5	Butcher-Schema zur Bestimmung der unbekannten Koeffizienten	10

1 Einleitung

Die Analyse von komplexen Energieversorgungsnetzen ist ohne das Hilfsmittel Computer kaum denkbar. Die Größe der Netze, sowie die Anzahl deren Komponenten und Speicherelemente macht eine rein analytische Berechnung sehr schwierig. Der Einsatz des Computers hat es dem Ingenieur nicht nur in der Energietechnik ermöglicht, verschiedenste dynamische Systeme zu simulieren, um dadurch Systemeigenschaften zu beobachten oder mit unerprobten Konzepten zu experimentieren. Dynamische Systeme sind dabei im allgemeinen dadurch gekennzeichnet, dass sie Energiespeicher enthalten, die zu zeitveränderlichen Vorgängen führen. Aus physikalischer Sicht sind diese Vorgänge durch Differentialgleichungen charakterisiert, deren Lösungen sich nicht immer in analytischer Form, oder zumindest einfach angeben lassen. Der Computer liefert die Möglichkeit, mit numerischen Methoden Näherungen dieser Lösungen zu berechnen. Hierfür gibt es eine Vielzahl verschiedener Methoden, welche unter gewissen Voraussetzungen zu unterschiedlichen Ergebnissen oder zu unterschiedlicher Genauigkeit führen können. Um Simulationssoftware anwenden zu können ist es deshalb notwendig, dass der Benutzer die Grundlagen der hinter der Software steckenden Numerik versteht.

Aus diesem Grund werden in dieser Arbeit die wichtigsten Verfahren zu numerischen Simulation von dynamischen Systemen vorgestellt. Dabei wird auf Einschrittverfahren, sowie auf Mehrschrittverfahren eingegangen. Darüber hinaus wird die Vorgehensweise der Schrittweitensteuerung erläutert. Im 3. Kapitel werden die verschiedenen Verfahren auf Stabilität untersucht, welche eine essenzielle Eigenschaft der numerischen Verfahren darstellt. Um die Bedeutung der Verfahren zu verdeutlichen, werden in Kapitel 4 einige Softwarepakete der zur Netzsimulation auf ihr verwendete Verfahren untersucht.

2 Numerische Integrationsverfahren für Anfangswertprobleme

Die Modellbildung dynamischer Systeme führt in vielen Fällen naturgemäß zu gewöhnlichen Differentialgleichungen (beziehungsweise Differentialgleichungssystemen) in Abhängigkeit der Zeit t , so genannten *Anfangswertproblemen*:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0. \quad (2.1)$$

Dabei ist die zeitliche Ableitung der Zustände $\dot{\mathbf{x}}(t) = \frac{d}{dt}\mathbf{x}(t)$ zu einem bestimmten Zeitpunkt in Form des Funktionenvektors \mathbf{f} berechenbar. Dabei mag \mathbf{f} auch nicht in analytische Form dargestellt werden können, sondern lediglich der Funktionswert berechenbar sein. Darüber hinaus ist der Anfangswert der Lösung $\mathbf{x}(0) = \mathbf{x}_0$ bekannt.

Um derartige Systeme simulieren zu können, muss eine Lösung $\mathbf{x}(t)$ des Anfangswertproblems gefunden werden. Die exakte Lösung dieses Problems ergibt sich durch Integration der zeitlichen Ableitung:

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \dot{\mathbf{x}}(\mathbf{x}, t) dt. \quad (2.2)$$

Die Herausforderung der Simulation besteht also darin, eine numerische Approximation dieser Integration zu ermitteln, was das Problem auf die numerische Integration von Anfangswertproblemen reduziert. Diese Approximation soll zu bestimmten, diskreten Zeitpunkten $t = t_i = i \cdot T, i = 0, \dots, n$ Näherungslösungen \mathbf{x}_i für den tatsächlichen Funktionswert $\mathbf{x}(t_i)$ liefern. Die Schrittweite T legt dabei den Abstand von zwei benachbarten Abtastzeitpunkten t_i fest.

Es soll also allgemein eine Berechnungsvorschrift

$$\mathbf{x}_{i+1} = \Phi(\mathbf{x}, t) \quad (2.3)$$

mit der Verfahrensfunktion $\Phi(\mathbf{x}, t)$ bestimmt werden, welche zum einen eine möglichst genaue Approximation der tatsächlichen Lösung liefert. Zum anderen soll das Verfahren so gewählt werden, dass auch möglichst komplexe Systeme approximiert werden können, das heißt, es ist erstrebenswert, den dabei anfallenden Rechenaufwand gering zu halten.

Im Folgenden werden verschiedene Verfahren zur Integration vorgestellt. Diese werden in Einschrittverfahren und Mehrschrittverfahren unterteilt. Soweit nicht anders angegeben wird sich dabei auf [1] bezogen.

2.1 Einschrittverfahren

Um die einfachsten Verfahren herzuleiten, sollen die Betrachtungen auf Systeme mit nur einer Zustandsgröße beschränkt werden, sodass sich die Vektordarstellung in Gleichung (2.1) zu einer skalaren Darstellung reduziert:

$$\dot{x}(t) = f(x(t), t), \quad x(0) = x_0. \quad (2.4)$$

2.1.1 Explizites Eulerverfahren

Der einfachste Ansatz, (2.2) zu approximieren, ist, die Steigung der Funktion $x(t)$ im Zeitintervall zwischen t_i und t_{i+1} als konstant anzunehmen. Dies entspricht dem Taylor-Satz erster Ordnung und führt auf die *explizite* Berechnungsvorschrift

$$x_{i+1} = x_i + T f(x_i, t_i). \quad (2.5)$$

Dieses Verfahren wird als *explizites Eulerverfahren* bezeichnet.

2.1.2 Implizites Eulerverfahren

Beim expliziten Eulerverfahren wird angenommen, dass die im Zeitpunkt t_i vorliegende Steigung im gesamten Intervall $t_i \dots t_{i+1}$ gültig ist. Genauso gut könnte angenommen werden, dass im betrachteten Intervall die Steigung des Zeitpunktes t_{i+1} vorliegt:

$$x_{i+1} = x_i + f(x_{i+1}, t_{i+1}) \quad (2.6)$$

In diesem Fall ergibt sich eine *implizite Gleichung* für den approximierten Wert x_{i+1} , welche nur unter Kenntnis von f aufgelöst werden kann. Deshalb gehört das Verfahren zur Klasse der *impliziten Verfahren* und wird als *implizites Eulerverfahren* bezeichnet.

2.1.3 Trapezregel/Heun-Verfahren

Einen Ansatz, die Verfahren zu verbessern, liefert die so genannte *Trapezregel*. Bei dieser werden die beiden Euler-Verfahren gewissermaßen kombiniert. Dabei wird für die Steigung im betrachteten Intervall der Mittelwert der Steigungen im i -ten und $i + 1$ -ten Schritt angenommen:

$$x_{i+1} = x_i + T \cdot \frac{1}{2} \cdot \left(f(x_i, t_i) + f(x_{i+1}, t_{i+1}) \right). \quad (2.7)$$

Dadurch ergibt sich allgemein ebenfalls ein implizites Verfahren. Um das Auflösen der impliziten Gleichung zu umgehen, kann der unbekannte Wert $f(x_{i+1}, t_{i+1})$ durch das explizite Euler-Verfahren abgeschätzt werden. Dies führt auf das *Heun-Verfahren*:

$$x_{i+1} = x_i + T \cdot \frac{1}{2} \cdot \left(f(x_i, t_i) + f\left(x_i + T \cdot f(x_i, t_i), t_{i+1}\right) \right).$$

2.1.4 Runge-Kutta-Verfahren

Um die bisher betrachteten Methoden zu verbessern, gibt es mehrere Ansätze. Ein naheliegender Gedanke ist, statt nur der rechten Seite der Differentialgleichung (2.4), Zeitableitungen höherer Ordnung in die Abschätzung einzubauen. Dies führt zu den Taylor-Methoden, welche auf dem Taylor-Satz basieren. Da die Differentiation der Rechten Seite jedoch mit verhältnismäßig großem rechnerischen Aufwand verbunden ist, sind diese Verfahren in der Praxis eher ungebräuchlich und sollen im Rahmen dieser Arbeit nicht näher betrachtet werden.

Um einen anderen Ansatz zu zeigen, wählt man den *globalen Fehler* $e(t_i) := \|x_i - x(t_i)\|$, den ein Verfahren im Vergleich zur exakten Lösung macht, als ein Maß, mit dem die Verfahren verglichen werden können. Es kann gezeigt werden, dass sich dieser für die betrachteten Integrationsverfahren mit der Landau-Notation durch

$$e(t) = \mathcal{O}(T^p) \quad (2.8)$$

c_1					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
\vdots	\vdots	\vdots	\ddots		
c_s	a_{s1}	a_{s2}	\cdots	a_{ss-1}	
	b_1	b_2	\cdots	b_{s-1}	b_s

Tabelle 2.1: Allgemeines Butcher-Schema expliziter Runge-Kutta-Verfahren

abschätzen lässt. (Getroffene Annahmen über die rechte Seite der Differentialgleichung (2.4) sollen im Rahmen dieser Arbeit außer Acht gelassen werden und können in [1] nachgelesen werden.) Dabei legt die Konsistenzordnung p fest, wie schnell der globale Fehler gegen Null geht, wenn die Schrittweite verkleinert wird und stellt damit ein Maß für die „Güte“ des Verfahrens dar. Es kann gezeigt werden, dass die Konsistenzordnung der Euler-Verfahren $p = 1$ beträgt. Das Heun-Verfahren ist mit $p = 2$ bereits deutlich genauer.

Deshalb ist der erste Ansatz in der Praxis, die Idee der Trapezregel beziehungsweise des Heun-Verfahrens weiter zu führen und die Konvergenzordnung der Verfahren durch weitere Stützstellen zu erhöhen. Dies führt zum allgemeinen Ansatz der *s-stufigen expliziten Runge-Kutta-Verfahren*. Aus Gründen der Übersichtlichkeit wurde der Index i zum Kennzeichnen des aktuell betrachteten Abtastpunktes im Folgenden weggelassen.

$$\Phi(x, t, T) = x + T \sum_{i=1}^s b_i k_i, \quad \text{mit} \quad (2.9)$$

$$k_i = f \left(x + T \sum_{j=1}^{i-1} a_{ij} k_j, t + c_i T \right) \quad \text{für } i = 1, \dots, s. \quad (2.10)$$

Dabei wird die Rechte Seite der DGL an den Stützstellen $t + c_i T$ berechnet, wobei die vorhergehenden Stützstellen gewichtet mit den Koeffizienten a_{ij} zur Berechnung des zur Stützstelle gehörenden Funktionswertes herangezogen werden. Die Gewichtung der berechneten Steigungen bei der Ermittlung der resultierenden Verfahrensfunktion erfolgt durch die Koeffizienten b_i . Mit dieser Darstellung ergeben die Runge-Kutta-Verfahren eine unendliche Menge verschiedener Verfahren zur Lösung des Problems.

Um die Runge-Kutta-Verfahren zu klassifizieren, werden die Koeffizienten a_{ij} , b_i , c_i häufig in Form des Butcher-Schemas, nach Tabelle 2.1 dargestellt. Sowohl das explizite Euler-Verfahren ($s=p=1$), als auch das Heun-Verfahren ($s=p=2$) lassen sich auf diese Weise beschreiben. Darüber hinaus ist das *klassische* Runge-Kutta-Verfahren ($s=p=4$) von großer Bedeutung. Für diese Verfahren ergeben sich die in Tabelle 2.2 dargestellten Butcher-

			0	
			$\frac{1}{2}$	$\frac{1}{2}$
	0		$\frac{1}{2}$	0
0	1	1	1	0
		$\frac{1}{2}$	$\frac{1}{2}$	0
				1
			$\frac{1}{6}$	$\frac{2}{6}$
				$\frac{2}{6}$
				$\frac{1}{6}$

Tabelle 2.2: Butcher-Schemata für (von links nach rechts): Euler-Verfahren, Heun-Verfahren, *klassisches* Runge-Kutta-Verfahren

		0	0	0
		1	$\frac{1}{2}$	$\frac{1}{2}$
1	1		$\frac{1}{2}$	$\frac{1}{2}$
	1		$\frac{1}{2}$	$\frac{1}{2}$

Tabelle 2.3: Butcher-Schemata für implizite Verfahren (von links nach rechts): Euler, Mittelpunktsregel, Trapezregel

Schemata.

Die selbe Vorgehensweise lässt sich für implizite Verfahren anwenden, sodass sich als Berechnungsvorschrift für *s-stufige implizite Runge-Kutta-Verfahren* ergibt:

$$\Phi(x, t, T) = x + T \sum_{i=1}^s b_i k_i \quad \text{mit} \quad (2.11)$$

$$k_i = f \left(x + T \sum_{j=1}^s a_{ij} k_j, t + c_i T \right) \quad \text{für } i = 1, \dots, s. \quad (2.12)$$

Beispiele für implizite Runge-Kutta-Verfahren sind das bereits kennengelernte implizite Euler-Verfahren ($s=p=1$), die implizite Mittelpunktsregel ($s=p=2$), sowie die implizite Trapezregel ($s=p=2$), deren Butcher-Schemata in Tabelle 2.3 dargestellt sind. Der Vorteil der impliziten Verfahren gegenüber den expliziten Verfahren wird sich bei Betrachtung des Stabilitätsverhaltens in Kapitel 3 zeigen.

2.2 Schrittweitensteuerung (Adaption)

Um die Verfahren weiter zu verbessern und gleichzeitig Rechenzeit einzusparen, liegt der Gedanke nahe, die bisher als konstant angenommene Schrittweite T als steuerbar anzusehen, sodass T_i an Stellen, bei welchen sich der Funktionswert stärker ändert, kleiner gewählt werden kann und an Stellen langsamer Änderungen hingegen größer. Aus Gründen der numerischen Effizienz ist es üblich, dass im i -ten Zeitschritt eine „gute“

Schrittweite für den Übergang von t_i nach t_{i+1} bestimmt wird, jedoch keine Anpassung von vorhergehenden Schrittweiten mehr erfolgt.

Um die Schrittweite anzupassen wird gefordert, dass der *lokale* Fehler $\varepsilon(t)$ unter einer gewissen Toleranzgrenze tol liegen soll. Der *lokale* Fehler ist dabei der Anteil am *globalen* Fehler, der durch den Zeitschritt von t_i bis t_{i+1} hervorgerufen wird. Da die exakte Lösung des Funktionsverlaufs nicht bekannt ist, wird für das verwendete Verfahren Φ mit Konsistenzordnung p ein anderes Verfahren $\hat{\Phi}$ zur Abschätzung des Fehlers herangezogen. Die Konsistenzordnung des zweiten Verfahrens \hat{p} wird kleiner als die des verwendeten Verfahrens gewählt, sodass $p \geq \hat{p} + 1$ gilt. Der Schritt von t_i nach $t_{i+1} = t_i + T_i$ wird mit beiden Verfahren berechnet und die Norm der Differenz der Ergebnisse als *Fehlerschätzer* $\bar{\varepsilon}$ bezeichnet.

$$\bar{\varepsilon} := \|\hat{\Phi}(t_i, x_i, T_i) - \Phi(t_i, x_i, T_i)\|. \quad (2.13)$$

Mit Hilfe der Konvergenzeigenschaften lässt sich zeigen, dass die gewünschte Fehlertoleranz näherungsweise eingehalten wird, falls die Schrittweite durch

$$T_{\text{neu}} = \sqrt[p+1]{\frac{\text{tol}}{\bar{\varepsilon}}} T_{\text{alt}} \quad (2.14)$$

angepasst wird. Da dies nur approximativ gilt, wird das Argument der Wurzel in der Praxis jedoch häufig durch einen „Sicherheitsfaktor“ $f_{\text{ac}} \in]0, 1[$ (typisch: 0.9) skaliert.

Nach der Berechnung des Fehlerschätzers $\bar{\varepsilon}$ im Zeitschritt i wird dieser mit der zulässigen Toleranz verglichen. Ist der berechnete Fehler größer, als die zulässige Toleranz ($\bar{\varepsilon} > \text{tol}$), wird der selbe Zeitschritt mit der angepassten Schrittweite T_{neu} erneut berechnet.

Gilt $\bar{\varepsilon} \leq \text{tol}$, so ist die gewünschte Genauigkeit erreicht und die der Berechnung zu Grunde liegende Schrittweite wird für diesen Zeitschritt angenommen. Nun wird erneut (2.14) berechnet und T_{neu} als Schrittweite T_{i+1} für den nächsten Zeitschritt angenommen. Dies sorgt dafür, dass die Schrittweite bei hoher Genauigkeit auch vergrößert werden kann.

2.3 Eingebettete Runge-Kutta-Verfahren

Um den Rechenaufwand der Beschriebenen Schrittweitenanpassung zu reduzieren, können die Verfahrensfunktionen Φ und $\hat{\Phi}$ so geschickt gewählt werden, dass bei deren Auswertungen auf gleiche Zwischenergebnisse zurückgegriffen werden kann. Dies führt auf die sehr häufig verwendeten eingebetteten Runge-Kutta verfahren. Die eingebetteten

c_1					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
\vdots	\vdots	\vdots	\ddots		
c_s	a_{s1}	a_{s2}	\cdots	$a_{s\,s-1}$	
	b_1	b_2	\cdots	b_{s-1}	b_s
	\hat{b}_1	\hat{b}_2	\cdots	\hat{b}_{s-1}	\hat{b}_s

Tabelle 2.4: allgemeines Butcher-Schema für eingebettete Runge-Kutta-Verfahren

0					
$\frac{1}{2}$	$\frac{1}{2}$				
$\frac{1}{2}$	0	$\frac{1}{2}$			
1	0	0	1		
c_5	a_{51}	a_{52}	a_{53}	a_{54}	
	$\frac{1}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{1}{6}$	0
	\hat{b}_1	\hat{b}_2	\hat{b}_3	\hat{b}_4	\hat{b}_5

Tabelle 2.5: Butcher-Schema zur Bestimmung der unbekannten Koeffizienten

Verfahren werden mit $\text{RKp}(\hat{p})$ bezeichnet. Der Gedanke dahinter ist, dass sich die berechneten Werte der Stützstellen k_i und \hat{k}_i nicht unterscheiden, sondern lediglich deren Linearkombination. Das heißt die Koeffizienten $a_{ij} = \hat{a}_{ij}$ und $c_i = \hat{c}_i$ stimmen in beiden Verfahren überein. Es gilt lediglich $b_i \neq \hat{b}_j$. Für eingebettete Runge-Kutta-Verfahren lassen sich deshalb auch sehr einfach Butcher-Schemata nach Tabelle 2.4 erstellen. Wie in [1] gezeigt wird, muss in diesem Fall zur Konstruktion eines Verfahrens der Konsistenzordnung $\hat{p} = p - 1$ ein Verfahren mit Stufenzahl $\hat{s} = s + 1$ gewählt werden. Für das klassische Runge-Kutta-Verfahren ($s=p=4$) ergibt sich dadurch das Butcher-Schema nach Tabelle 2.5. Um das zugehörige Verfahren $\hat{\Phi}$ mit $\hat{s} = 5$ und $\hat{p} = 3$ zu bestimmen, müssen also die unbekannten Koeffizienten in 2.5 bestimmt werden. Dies kann in [1] nachgelesen werden und soll an dieser Stelle nicht weiter vertieft werden. Eines der am weitesten verbreiteten Verfahren dieser Klasse ist das Dormand-Prince-RK5(4)-Verfahren, welches in der Numerik-Software MATLAB unter dem Namen ODE45 standardmäßig eingestellt ist. Die zugehörigen Koeffizienten können in Anhang A.2 nachgelesen werden.

2.4 Mehrschrittverfahren

Die bisher betrachteten Verfahren sind dadurch charakterisiert, dass die Verfahrensfunktion $\Phi(x, t)$ lediglich vom aktuellen Funktionswert x_i abhängt. Deshalb werden diese Verfahren als Einschrittverfahren bezeichnet.

Im Gegensatz dazu hängt der Funktionswert x_{i+1} bei Mehrschrittverfahren zusätzlich dazu von einer beliebigen Anzahl an Vorgängerwerten x_{i-k+1}, \dots, x_i ab. Damit wird ein k -stufiges lineares Mehrschrittverfahren folgendermaßen definiert:

$$\begin{aligned} & a_k x_{i+k} + a_{k-1} x_{i+k-1} + \dots + a_0 x_i \\ & = T \left(b_k f(x_{i+k}, t_{i+k}) + b_{k-1} f(x_{i+k-1}, t_{i+k-1}) + \dots + b_0 f(x_i, t_i) \right) \end{aligned} \quad (2.15)$$

mit $a_k \neq 0$. Durch \mathcal{Z} -Transformation (nachzulesen zum Beispiel in [2]) lassen sich Mehrschrittverfahren auch durch die beiden Polynome

$$\underline{P}_a(z) = a_0 + a_1 z + \dots + a_k z^k \quad (2.16)$$

$$\underline{P}_b(z) = b_0 + b_1 z + \dots + b_k z^k \quad (2.17)$$

charakterisieren. Ein Beispiel für ein Mehrschrittverfahren ist das (implizite) Milne-Simpson-Verfahren:

$$x_{i+1} = x_{i-1} + \frac{T}{3} \left(f(x_{i+1}, t_{i+1}) + 4f(x_i, t_i) + f(x_{i-1}, t_{i-1}) \right). \quad (2.18)$$

2.4.1 Adams-Verfahren/Prädiktor-Korrektor-Verfahren

Wie auch bei den Einschrittverfahren können Mehrschrittverfahren in explizite und implizite Verfahren unterteilt werden. In [3] wird gezeigt, dass implizite Verfahren im allgemeinen erheblich bessere Ergebnisse erzielen, als explizite Verfahren, jedoch das Auflösen der impliziten Gleichung nach dem gesuchten Funktionswert nicht realisierbar sein kann oder zumindest mit erheblichem Rechenaufwand verbunden ist.

Dies führt – ähnlich wie bei dem Heun-Verfahren – zu dem Gedanken, Approximationen, welche durch ein explizites Verfahren gewonnen wurden, durch ein implizites Verfahren zu verbessern. Diese Kombination aus explizitem und implizitem Mehrschrittverfahren wird als Prädiktor-Korrektor-Verfahren bezeichnet.

Die hierfür am häufigsten verwendeten Mehrschrittverfahren sind die so genannten Adams-Verfahren. Dabei werden die expliziten Verfahren als *Adams-Bashforth-Verfahren*

und die impliziten Verfahren als *Adams-Moulton-Verfahren* bezeichnet. Die Koeffizienten dieser Verfahren können in Anhang A.3 nachgeschlagen werden.

Nach [3] liegen Rechenaufwand und Genauigkeit bei einem m -stufigen *Adams-Bashforth-Verfahren* und einem $m-1$ -stufigen *Adams-Moulton-Verfahren* im gleichen Größenbereich, wobei der lokale Fehler in beiden Fällen von T^{m+1} abhängt. Wie bereits eingangs erwähnt, liefern die impliziten Verfahren jedoch deutlich genauere Approximationen.

Zur Durchführung des Prädiktor-Korrektor-Verfahrens wird zunächst mit dem expliziten m -stufigen *Adams-Bashforth-Verfahren* ein Approximationswert $x_{i+1}^{(0)}$ berechnet, also die Prädiktion durchgeführt. Dieser wird dann in die rechte Seite der impliziten Gleichung des *Adams-Moulton-Verfahrens* eingesetzt, wodurch eine bessere Schätzung $x_{i+1}^{(1)}$ berechnet werden kann (Korrektur). Diese wird nun als Funktionswert $x(t_{i+1})$ angenommen. In [4] wird beschrieben, dass der Korrektur-Schritt in der Praxis iterativ wiederholt werden kann, um eine noch höhere Genauigkeit zu erzielen.

Da ein k -stufiges Mehrschrittverfahren zur Berechnung des Funktionswertes x_i alle zurückliegenden Funktionswerte bis x_{i-k} benötigt, ist der erste überhaupt berechenbare Funktionswert $x(t = t_k)$. Aus diesem Grund müssen Mehrschrittverfahren durch geeignete Einschrittverfahren „gestartet“ werden. Man bezeichnet Einschrittverfahren im Gegensatz zu Mehrschrittverfahren aus diesem Grund auch als *selbststartend*.

2.4.2 BDF-Verfahren

Die zweite sehr bedeutende Klasse der Mehrschrittverfahren sind die so genannten *Backwards Differentiation Formulas* (BDF). Für diese Verfahren gilt $\underline{P}_b(z) = \underline{z}^k$. Damit lassen sich eine Reihe impliziter Verfahren erzeugen, die besonders gut für steife Differentialgleichungen geeignet sind. Die Bedeutung dessen wird im folgenden Kapitel erläutert.

3 Stabilitätsanalyse der wichtigsten Methoden

Neben der Konvergenz des Fehlers mit der Schrittweite $e(t) = \mathcal{O}(T^p)$ ist für die Anwendbarkeit eines Verfahrens entscheidend, welche Voraussetzungen an die Schrittweite gestellt werden müssen, damit das Verfahren bei der Lösung von Differentialgleichungen stabil ist. Unter der Stabilität soll in diesem Zusammenhang die exponentielle Stabilität verstanden werden, wie sie in [1] beschrieben wird. Zur Einfachheit werden die Betrachtungen an dieser Stelle auf die Klasse *linearer zeitinvarianter Systeme* beschränkt, für die gilt:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}. \quad (3.1)$$

Aus der Systemtheorie ist bekannt, dass ein solches System genau dann exponentiell stabil ist, wenn für alle Eigenwerte der Matrix \mathbf{A} gilt: $\Re \lambda_i < 0$, das heißt, die Eigenwerte liegen in der linken Halbebene der komplexen Zahlenebene \mathbb{C}^- .

3.1 Steife Differentialgleichungen

Zur Einführung soll das Anfangswertproblem

$$\dot{\mathbf{x}} = \lambda \mathbf{x}, \quad \mathbf{x}(0) = \mathbf{x}_0 \quad (3.2)$$

betrachtet werden. Für diese einfache Differentialgleichung lautet die geschlossene Lösung $\mathbf{x}(t) = e^{\lambda t} \mathbf{x}_0$. Das heißt für $\lambda < 0$ konvergiert diese Lösung gegen 0. Eine Approximation des Systems mit dem expliziten Euler-Verfahren liefert

$$\mathbf{x}_{i+1} = \mathbf{x}_i + T\lambda \mathbf{x}_i = (1 + T\lambda)\mathbf{x}_i. \quad (3.3)$$

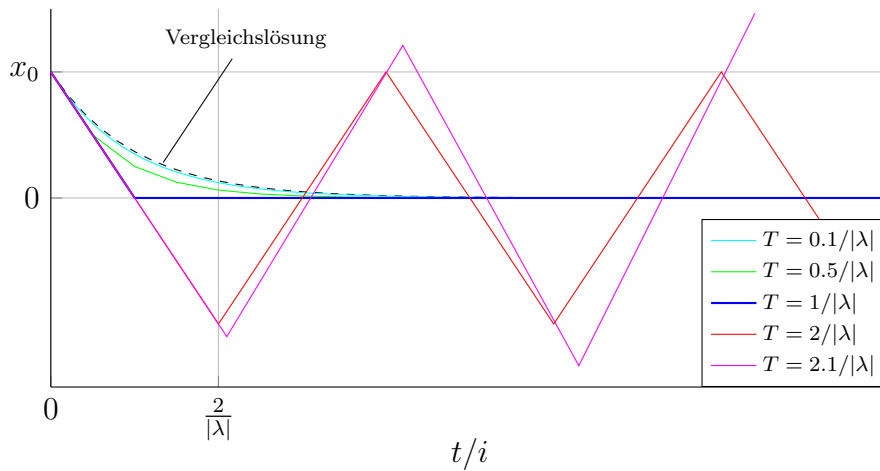


Abbildung 3.1: Simulation des Beispielsystems mit explizitem Euler-Verfahren

Diese Differenzengleichung konvergiert genau dann ebenfalls gegen 0, wenn

$$|1 + T\lambda| < 1 \quad (3.4)$$

gilt. Für die Schrittweite T muss also die Bedingung

$$T < \frac{2}{|\lambda|} \quad (3.5)$$

erfüllt sein, damit die Approximation eine Sinnvolle Näherung darstellt. Es ist augenscheinlich, dass dies bei sehr schnellen Systemen mit $\Re\lambda \ll 0$ zu sehr kleinen notwendigen Schrittweiten führt.

Aus diesem Grund werden Differentialgleichungen mit sehr schnellen stabilen Eigenwerten $\Re\lambda \ll 0$ allgemein als *steif* bezeichnet. Für den Fall, dass das Anfangswertproblem ein Differentialgleichungssystem mit mehreren Eigenwerten darstellt spricht man von einem *steifen* System, falls mindestens ein solcher Eigenwert vorliegt. Die Approximation von steifen Differentialgleichungen führt vor allem bei expliziten Verfahren zu Problemen und stellt die wesentliche Motivation zur Verwendung von impliziten Verfahren dar. In Abbildung 3.1 ist die Approximation von Gleichung (3.2) mit dem expliziten Euler-Verfahren für verschiedene Schrittweiten dargestellt.

3.2 Einschrittverfahren

Das Beispiel der steifen Differentialgleichungen hat gezeigt, dass durch die Wahl der Schrittweite im allgemeinen das Stabilitätsverhalten des Verfahrens beeinflussen lässt. Im

Folgenden sollen Bedingungen aufgezeigt werden, die an die Schrittweite gestellt werden müssen, um die Stabilität eines Verfahrens allgemein zu gewährleisten.

Nach [1] lassen sich alle Runge-Kutta-Verfahren mit konstanter Schrittweite, also alle bisher betrachteten Einschrittverfahren ohne Schrittweitensteuerung durch die Differenzengleichung

$$\mathbf{x}_{i+1} = \tilde{\mathbf{A}} \mathbf{x}_i \quad (3.6)$$

beschreiben. Nach [2] ist eine solche Differenzengleichung genau dann exponentiell stabil, wenn für alle Eigenwerte der Matrix $\tilde{\mathbf{A}}$ stabil sind, das heißt $|\tilde{\lambda}_i| < 1$ gilt. Die Eigenwerte müssen also im Einheitskreis $B_1(0)$ der komplexen Zahlenebene liegen. ($B_u(\underline{y})$ bezeichnet im Folgenden allgemein einen Kreis mit Radius u um den Punkt \underline{y} in der komplexen Zahlenebene.) Zur Untersuchung der Stabilität eines Verfahrens müsste also die Matrix $\tilde{\mathbf{A}}$ aus den Koeffizienten des Verfahrens und der Matrix \mathbf{A} der zu Grunde liegenden Differentialgleichung berechnet werden, also als Funktion der Art

$$\tilde{\mathbf{A}} = \mathbf{R}(\mathbf{T}\mathbf{A}). \quad (3.7)$$

Es kann gezeigt werden, dass diese Funktion für explizite Werte $\underline{z} \in \mathbb{C}$ die Form

$$\underline{\mathbf{R}}(\underline{z}) = 1 + \underline{z} \mathbf{b}^T (\mathbf{I} - \underline{z} \mathbf{A})^{-1} \mathbf{e} \quad (3.8)$$

annimmt, wobei \mathbf{I} für die Einheitsmatrix steht, $\mathbf{b} = (b_1, \dots, b_s)^T$ und $\mathbf{A} = (a_{ij})_{i,j=1,\dots,s}$ die Koeffizienten des Verfahrens beinhalten und $\mathbf{e} = (1, \dots, 1)^T \in \mathbb{R}^s$ gilt. $\underline{\mathbf{R}}(\underline{z})$ wird als die Stabilitätsfunktion des Verfahrens bezeichnet. Die Stabilitätsfunktion an sich ist nun unabhängig von der Schrittweite und der Dynamik der betrachteten Differentialgleichung und kann damit noch keine endgültige Aussage über die Stabilität des Verfahrens treffen. Um diese zu ermöglichen, gilt es herauszufinden, welche Bedingungen \mathbf{A} und \mathbf{T} erfüllen müssen, damit die Eigenwerte der Matrix $\tilde{\mathbf{A}}$ stabil sind. Hierfür wird das Stabilitätsgebiet $S \subset \mathbb{C}$ eines Runge-Kutta-Verfahrens mit Stabilitätsfunktion $\underline{\mathbf{R}}$ definiert als die Menge von Eigenwerten $\underline{\lambda}_i$, die $\mathbf{T}\mathbf{A}$ annehmen darf, damit $\tilde{\mathbf{A}} = \mathbf{R}(\mathbf{T}\mathbf{A})$ exponentiell stabil ist. Dies ist genau dann der Fall, wenn der Betrag der Stabilitätsfunktion kleiner eins ist:

$$S = \{\underline{z} \in \mathbb{C} \mid |\underline{\mathbf{R}}(\underline{z})| < 1\}. \quad (3.9)$$

Für die Stabilität des Verfahrens müssen also alle Eigenwerte $\mathbf{T}\underline{\lambda}_i$ der Matrix $\tilde{\mathbf{A}}$ im Stabilitätsgebiet liegen. Dabei gilt zu beachten, dass für die Menge aller Eigenwerte Σ zweier Matrizen gilt: $\underline{\lambda}_i \in \Sigma(\mathbf{A}) \Rightarrow \mathbf{T}\underline{\lambda}_i \in \Sigma(\mathbf{T}\mathbf{A})$.

Damit lässt sich die Schrittweite T in Abhängigkeit der Eigenwerte von \mathbf{A} so berechnen, dass das verwendete Verfahren stabil ist. Allgemein gilt, je größer das Stabilitätsgebiet S , desto größer kann die Schrittweite T gewählt werden, um Stabilität zu gewährleisten. Ein Verfahren wird als A-stabil bezeichnet, wenn es für eine stabile Dynamik \mathbf{A} unabhängig von der Schrittweite stabil ist. In diesem Fall ist die gesamte linke Halbebene der komplexen Zahlenebene im Stabilitätsgebiet enthalten: $\mathbb{C}^- \subseteq S$.

Zur Vergleichbarkeit sollen nun die betrachteten Verfahren auf das Stabilitätsgebiet untersucht werden. Für das explizite Euler-Verfahren lautet die Stabilitätsfunktion $\underline{R}_{\text{Eu,ex}}(z) = 1 + z$. Damit ergibt sich die Ungleichung

$$|\underline{R}_{\text{Eu,ex}}(z)| < 1 \Leftrightarrow |1 + z| < 1, \quad (3.10)$$

also $S_{\text{Eu,ex}} = \{z \in \mathbb{C} \mid |1 + z| < 1\} = B_1(-1)$. Die Schrittweite muss also so gewählt werden, dass $T\lambda_i \in B_1(-1)$ gilt. Für das betrachtete Beispielsystem entspricht dies genau (3.5).

Für das implizite Euler-Verfahren erhält man $\underline{R}_{\text{Eu,im}}(z) = \frac{1}{1-z}$. Für das Stabilitätsgebiet bedeutet das

$$|\underline{R}_{\text{Eu,im}}(z)| < 1 \Leftrightarrow |1 - z| > 1, \quad (3.11)$$

das heißt $S_{\text{Eu,im}}$ ist \mathbb{C} ohne den Kreis $B_1(1)$. Damit ist das implizite Euler-Verfahren – wie viele implizite Verfahren – A-stabil, was hilfreich bei der Simulation von stabilen Systemen ist. Allerdings enthält das Stabilitätsgebiet auch Bereiche, in denen die Approximation stabil ist, obwohl das eigentliche System instabil ist, wie zum Beispiel $\Re z > 2$. In diesem Fall würde die Instabilität eines Systems in der Simulation möglicherweise unentdeckt bleiben, was eines der Probleme des impliziten Euler-Verfahrens ist und eine Motivation für die Trapezregel liefert.

Für diese ergibt sich als Stabilitätsfunktion $\underline{R}_{\text{Trapez}}(z) = \frac{2+z}{2-z}$. Das Stabilitätsgebiet entspricht damit genau der linken Halbebene $S_{\text{Trapez}} = \mathbb{C}^-$. Damit ist die Trapezregel ebenfalls A-stabil, enthält jedoch keine Bereiche, in denen ein instabiles System stabil approximiert werden würde. Man bezeichnet diese Tatsache mit *Erhaltung der Isometrie*, das heißt, die Approximation ist genau dann stabil, wenn auch das zu Grunde liegende System exponentiell stabil ist. Damit besitzt die Trapezregel quasi ideale Stabilitätseigenschaften.

Aufgrund der schwierigen Realisierbarkeit der impliziten Verfahren wurde das Heun-Verfahren auf Basis der Trapezregel und dem expliziten Euler-Verfahren eingeführt. Hierfür ergibt sich die Stabilitätsfunktion $\underline{R}_{\text{Heun}}(z) = \frac{1}{2}z^2 + z + 1$. Das zugehörige Sta-

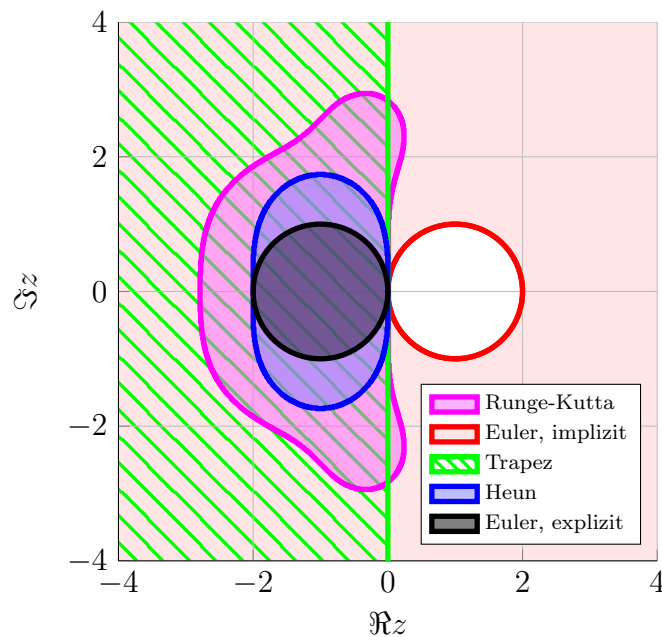


Abbildung 3.2: Stabilitätsgebiete für einige Einschrittverfahren

Stabilitätsgebiet ist in Abbildung 3.2 zu sehen. Vergleicht man die Stabilitätsgebiete von explizitem Euler-Verfahren und Heun-Verfahren, so wird deutlich, dass das Stabilitätsgebiet des Heun-Verfahrens nur wenig größer ist, als das des expliziten Euler-Verfahrens. Zwischen den Stabilitätsgebieten von Heun-Verfahren und Trapezregel besteht hingegen ein deutlicher Größenunterschied.

Ein größeres Stabilitätsgebiet kann zum Beispiel mit dem *klassischen* Runge-Kutta-Verfahren mit der Stabilitätsfunktion $R_{RK}(z) = \frac{1}{24}z^4 + \frac{1}{6}z^3 + \frac{1}{2}z^2 + z + 1$ erreicht werden. Ähnlich, wie bei den impliziten Verfahren treten hier jedoch kleine Bereiche auf, in denen ein instabiles System zu einer stabilen Approximation führen würde. Grundsätzlich gilt für explizite Runge-Kutta-Verfahren: Je Höher die Ordnung des Verfahrens, desto größer das Stabilitätsgebiet. Bei der Betrachtung der Stabilitätsgebiete wird auch der große Vorteil der impliziten Verfahren gegenüber expliziten Verfahren deutlich. Zwar gibt es auch implizite Verfahren, die nur kleine Stabilitätsgebiete besitzen und damit schlecht für die Lösung steifer Differentialgleichungen geeignet sind, jedoch sind die Stabilitätsgebiete von expliziten Verfahren im Allgemeinen kleiner, als die der impliziten Verfahren.

3.3 Mehrschrittverfahren

Die Stabilität von Mehrschrittverfahren ist nicht so einfach allgemein nachzuweisen, wie bei den allgemeinen Runge-Kutta Verfahren. In [1] wird gezeigt, dass die „Stabilität“ des Polynoms $\underline{P}_a(\underline{z})$ eine Voraussetzung für die *Konvergenz* des Fehlers eines Mehrschrittverfahrens darstellt. Dabei gilt das Polynom als stabil, wenn für alle Nullstellen $\underline{\lambda}_i$ gilt:

$$|\underline{\lambda}_i| \leq 1, \quad (3.12)$$

$$|\underline{\lambda}_i| = 1 \Rightarrow \underline{\lambda}_i \text{ ist einfache Nullstelle.} \quad (3.13)$$

Diese „Stabilitätsbedingung“ darf jedoch nicht mit den für Einschrittverfahren gezeigten Stabilitätskriterien verwechselt werden, da sie noch keine Aussage über die tatsächliche Stabilität des Verfahrens trifft, sondern lediglich die Konvergenz des Fehlers ermöglicht, welche bei den Einschrittverfahren ohnehin vorausgesetzt wurde. Da eine allgemeine Aussage über die Stabilität eines Mehrschrittverfahrens schwierig ist, wird in [5] gezeigt, dass die Stabilität eines Verfahrens im konkreten Fall überprüft werden kann, indem es folgendermaßen umgeformt wird:

$$\underline{x}_{i+1} + a_{k-1}\underline{x}_i + a_{k-2}\underline{x}_{i-1} + \dots + a_0\underline{x}_{i+1-k} = 0. \quad (3.14)$$

Das heißt, die Koeffizienten $a_l (l = 0, \dots, s-1)$ beinhalten bereits die Koeffizienten des gewählten Verfahrens, sowie die gewählte Schrittweite. Ähnlich wie bei der Herleitung der Polynome $\underline{P}_a(\underline{z})$ und $\underline{P}_b(\underline{z})$ erhält man damit ein *charakteristisches Polynom*

$$\underline{P}_k(\underline{z}) = \underline{z}^k + a_{k-1}\underline{z}^{k-1} + a_{k-2}\underline{z}^{k-2} + \dots + a_0 = \sum_{l=0}^k a_l \underline{z}^l. \quad (3.15)$$

Das gewählte Verfahren ist mit der verwendeten Schrittweite genau dann stabil, wenn für alle Nullstellen des charakteristischen Polynoms $\underline{z}_{0,l}$, ($l = 1, \dots, k$) gilt:

$$|\underline{z}_{0,l}| < 1. \quad (3.16)$$

4 Anwendung der Integrationsmethoden in Simulationssoftware

Um die Bedeutung der vorgestellten Integrationsmethoden zu verdeutlichen, soll in diesem Kapitel kurz dargelegt werden, welche Verfahren von einigen ausgewählten Programmen, die in der Energietechnik zum Einsatz kommen, verwendet werden.

PSS®Netomac, PSS®E

Die Firma SIEMENS liefert mit den Programmen PSS®NETOMAC und PSS®E gleich zwei Softwarepakete, die in der Energietechnik Anwendung finden. Ersteres verwendet die Trapezregel als Integrationsverfahren. Das heißt, es wird Modellwissen verwendet, um die implizite Gleichung wirklich aufzulösen. Da die Trapezregel Isometrieerhaltend ist, ergeben sich keine Stabilitätsprobleme und die Wahl der Schrittweite beeinflusst lediglich die Genauigkeit.

In letzterem Softwarepaket wird nach [6] das „second-order Euler“-Verfahren, oder „modified Euler“-Verfahren verwendet. Nach der Schreibweise in dieser Arbeit entspricht dies dem Heun-Verfahren (2.8). In diesem Fall muss die Schrittweite klein genug gewählt werden, um Stabilität zu gewährleisten. In [6] wird deshalb empfohlen, die Schrittweite zwischen $\frac{1}{5}$ und $\frac{1}{4}$ der kleinsten im Prozess auftretenden Zeitkonstanten zu wählen.

PowerFactory

Ein weitere relevante Software in der Energietechnik ist POWERFACTORY von DIGSILENT. In [7] wird nicht genau beschrieben, welches Integrationsverfahren verwendet wird. Es wird angedeutet, dass das Verfahren der RMS-Simulation A-stabil ist, was auf ein implizites Verfahren hindeutet. Auch hier sind dynamische Modelle hinterlegt, die das Auswerten der impliziten Gleichung ermöglichen. Darüber hinaus wird

angegeben, dass eine automatische Schrittweitenanpassung erfolgt, beziehungsweise bei EMT-Simulationen mit variabler oder fester Schrittweite gerechnet werden kann.

Eurostag

Die Software EUROSTAG des französischen Energieversorgers RTE greift nach [8] bei der Simulation auf das in [9] beschriebenen „mixed adams - BDF“-Vefahren zurück. Dabei handelt es sich um ein Prädiktor-Korrektor Verfahren, bei dem je nach Problemstellung ein implizites Adams-Verfahren oder ein BDF-Verfahren als Korrektor eingesetzt wird. Darüber hinaus wird die Schrittweite, ähnlich wie bei Einschrittverfahren gezeigt, gesteuert. Wie bei allen Verfahren mit Schrittweitensteuerung kommt es hierbei für die Genauigkeit und für die Stabilität auf die vom Anwender eingestellte Toleranz des lokalen Fehlers an.

5 Zusammenfassung und Ausblick

In dieser Arbeit wurden die wichtigsten numerischen Integrationsverfahren für Anfangswertprobleme kurz vorgestellt. Dabei wurde sowohl auf explizite, als auch implizite Verfahren eingegangen und zwischen Einschritt- und Mehrschrittverfahren unterschieden. Die Vorgehensweise der Schrittweitensteuerung wurde dargelegt und deren Anwendung durch eingebettete Runge-Kutta-Verfahren beschrieben.

Darüber hinaus konnte eine allgemeine Vorgehensweise zur Betrachtung des Stabilitätsverhaltens von Einschrittverfahren aufgezeigt werden und an einigen einfachen Verfahren verdeutlicht werden. Dabei wurden die Vor- und Nachteile der impliziten Verfahren gegenüber den expliziten Verfahren deutlich. Für Mehrschrittverfahren konnte eine Methode zur Überprüfung der Stabilität in konkreten Anwendungsfällen vorgeschlagen werden. Abschließend wurden ausgewählte, in der Energietechnik zum Einsatz kommende Computerprogramme knapp auf deren verwendete Integrationsverfahren untersucht.

Auf die Konstruktion der verschiedenen Verfahren wurde – abgesehen von den einfachsten Verfahren – im Rahmen dieser Arbeit bewusst verzichtet. Diese können in [1] oder entsprechender Literatur nachgelesen werden.

A Überblick einiger Verfahren

A.1 Einschrittverfahren

Name	Konsistenzordnung	Butcher-Schema	Art	Stabilitätsfunktion
Euler	1	$\begin{array}{c c} 0 & \\ \hline & 1 \end{array}$	explizit	$1 + z$
Heun	2	$\begin{array}{c c} 0 & \\ 1 & 1 \\ \hline \frac{1}{2} & \frac{1}{2} \end{array}$	explizit	$\frac{1}{2}z^2 + z + 1$
klassisches Runge-Kutta	4	$\begin{array}{c c} 0 & \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 \quad \frac{1}{2} \\ 1 & 0 \quad 0 \quad 1 \\ \hline \frac{1}{6} & \frac{2}{6} \quad \frac{2}{6} \quad \frac{1}{6} \end{array}$	explizit	$\frac{1}{24}z^4 + \frac{1}{6}z^2 + \frac{1}{2}z^2 + z + 1$
Euler	1	$\begin{array}{c c} 1 & 1 \\ \hline & 1 \end{array}$	implizit	$\frac{1}{1-z}$
Mittelpunktsregel	2	$\begin{array}{c c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$	implizit	$\frac{2+z}{2-z}$
Trapezregel	2	$\begin{array}{c c} 0 & 0 \quad 0 \\ 1 & \frac{1}{2} \quad \frac{1}{2} \\ \hline & \frac{1}{2} \quad \frac{1}{2} \end{array}$	implizit	$\frac{2+z}{2-z}$

A.2 Eingebettete Runge-Kutta-Verfahren (Schrittweitensteuerung)

Name/ Ordnung	Stufen	Butcher-Schema							
Fehlberg- RK4(3)	5	0							
		$\frac{1}{2}$	$\frac{1}{2}$						
		$\frac{1}{2}$	0	$\frac{1}{2}$					
		1	0	0	1				
		1	$\frac{1}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{1}{6}$			
		$\frac{1}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{1}{6}$	0			
		$\frac{1}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	0	$\frac{1}{6}$			
Dormand- Prince- RK5(4)	6	0							
		$\frac{1}{5}$	$\frac{1}{5}$						
		$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
		$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
		$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{5103}{18656}$			
		1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
		1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{92}$	$-\frac{2187}{6487}$	$\frac{11}{84}$	
			$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{92}$	$-\frac{2187}{6487}$	$\frac{11}{84}$	0
			$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

A.3 Mehrschrittverfahren

k	Name	$P_a(z)$	$P_b(z)$	Art	Hinweis
2	Milne-Simpson	$z^2 - 1$	$\frac{1}{3}z^2 + \frac{4}{3}z + \frac{1}{3}$	implizit	
1			1 ($\hat{=}$ Euler, explizit)		
2	Adams-	$z^k - z^{k-1}$	$(3z - 1)/2$	explizit	häufig zusammen als eingebettete Runge-Kutta- Verfahren verwendet
3	Bashforth		$(23z^2 - 16z + 5)/12$		
4			$(55z^3 - 59z^2 + 37z - 9)/24$		
2	Adams-		$(5z^2 + 8z - 1)/12$		
3	Moulton	$z^k - z^{k-1}$	$(9z^3 + 19z^2 - 5z + 1)/24$	implizit	
4			$(251z^4 + 646z^3 + -246z^2 + 106z - 19)$		
1		$z - 1$ ($\hat{=}$ Euler, implizit)			
2		$\frac{3}{2}z^2 - 2z + \frac{1}{2}$			
3	BDF	$\frac{11}{6}z^3 - 3z^2 + \frac{3}{2}z - \frac{1}{3}$	z^k	implizit	unendlich großes Stabi- litätsgebiet \Rightarrow gut für steife DGL
4		$\frac{25}{12}z^4 - 4z^3 + 3z^2 - \frac{4}{3}z + \frac{1}{4}$			

Literaturverzeichnis

- [1] GRÜNE, Lars: Numerische Methoden für gewöhnliche Differentialgleichungen (Numerische Mathematik II) / Mathematisches Institut, Fakultät für Mathematik und Physik, Universität Bayreuth. Sommersemester 2008, 3. Auflage. – Vorlesungsskript
- [2] HARKORT, Christian: Digitale Regelung / Lehrstuhl für Regelungstechnik, Friedrich-Alexander Universität Erlangen-Nürnberg. Sommersemester 2013. – Vorlesungsskript
- [3] FAIRES, J. D. ; BURDEN, Richard L.: *Numerische Methoden*. Spektrum Akademischer Verlag Heidelberg, Berlin, Oxford, 1994
- [4] SIEMENS AG, PTD SE N.: *NETOMAC – Theorie-Buch*
- [5] FRIEDRICH, Hermann ; PIETSCHMANN, Frank: *Numerische Methoden*. De Gruyter, Berlin, New York, 2010
- [6] SIEMENS AG, PTD SE N.: *PSS®E 33.3 – Program Application Guide*. Volume 2
- [7] DIGSILENT: *DIGSILENT PowerFactory*. Rev 2.30/01/2013_D
- [8] EVRARD, C. ; BIHAIN, A.: Powerfull Tools for Various Types of Dynamic Studies of Power systems.
- [9] ASTIC, J.Y. ; BIHAIN, A. ; JEROSOLIMSKI, M.: The mixed Adams - BDF variable step size algortihm to simulate transient and long term phenomena in power systems. In: *IEEE Transactions on Power Systems* Vol. 9, No.2 (May 1994), S. 929–935