Lehrstuhl für Elektrische Energiesysteme
Vorstand: Univ.-Prof. Dr.-Ing. Matthias Luther

Art der Arbeit und Nr

Hier steht das Thema der schriftlichen Ausarbeitung

Bearbeiter: Vorname Nachname
Matrikelnummer

Betreuer: Vorname Nachname

Abgabedatum: TT.MM.JJJJ

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

Aufgabenstellung der Arbeit

Thema: Thema der Arbeit

Hier wird die Aufgabenstellung beschrieben. Die Notwendigkeit dieser hängt vom Betreuer ab.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Dieses Kapitel dient zur Hinführung an das Thema. Hier ist herauszuarbeiten, warum das Thema von Interesse und Bedeutung ist. Des Weiteren ist kurz der Aufbau der Arbeit anzusprechen. Der Umfang des Kapitels sollte eine oder wenige Seiten betragen.

2 Numerische Integrationsverfahren für Anfangswertprobleme

NOCH SCHAUEN, DASS f(x,t) oder f(t,x) konsistent dargestellt ist! EVENTUELL: $f_n = f(x_n, t_n)$ als Schreibweise einfuehren!! Kann einiges erleichtern!

Ausgangslage: Numerische Lösung von Anfangswertproblemen. Systeme beschrieben durch allgemeine Zustandsdarstellung: (Zur Einfachheit Eingrößensysteme.)

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\,\boldsymbol{x}(t) + \boldsymbol{b}\,u(t), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{2.1}$$

$$y(t) = \boldsymbol{c}^T\,\boldsymbol{x}(t) \tag{2.2}$$

Es ergibt sich als Lösung von $\boldsymbol{x}(t)$:

$$x(t) = x_0 + \int_0^t \dot{x}(x,t)\mathrm{d}t \tag{2.3}$$

Dafür numerische Lösung erforderlich. Dabei wird mit der (zunächst festen) Schrittweite T gerechnet. Schreibweise festlegen: x_i bezeichnet numerischen Wert im i-ten Abtastzeitpunkt, t_i, die zugehoerigen Zeiten. Allgemein soll also in jedem Schritt eine Lösung

$$x_{i+1} = \Phi(x,t) \tag{2.4}$$

mit noch unbekannter Verfahrensfunktion $\Phi(x,t)$ bestimmt werden.

2.1 Einschrittverfahren

2.1.1 Explizites Eulerverfahren

Einfachstes Verfahren, ergibt sich aus dem Taylor-Satz erster Ordnung. Steigung wird in jedem Zeitschritt berechnet und als konstant angenommen. (Zur Verdeutlichung im Zuge dieser Arbeit nur eine Zustandsgröße!)

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \lambda \cdot x + b \cdot u \tag{2.5}$$

$$\Rightarrow \frac{\Delta x}{T} = \lambda \cdot x + b \cdot u \tag{2.6}$$

$$\Rightarrow x_{i+1} = x_i + \Delta x_i = x_i + (\lambda \cdot x_i + b \cdot u_i) \cdot T \tag{2.7}$$

$$\Rightarrow \text{Stabilitätsanalyse mit digitaler Regelung Methoden machen!!} \tag{2.8}$$

2.1.2 Impliztes Eulerverfahren

Andere Möglichkeit: Annahme der Steigung des $i+1$-ten Schrittes für Δx.

$$x_{i+1} = x_i + \Delta x_{i+1} \tag{2.9}$$

$$x_{i+1} = x_i + \Delta x_{i+1} \text{ mit} \tag{2.10}$$

$$\Delta x_{i+1} = (\lambda \cdot x_{i+1} + b \cdot u_{i+1}) \cdot T \text{ (vorherige Gl. einsetzen)} \tag{2.11}$$

$$\Rightarrow \Delta x_{i+1} = (1 - \lambda \cdot T)^{-1} \cdot (b \cdot u_{i+1} + \lambda x_i) \cdot T \tag{2.12}$$

Es ergibt sich eine implizite Gleichung für die Steigung, und u_{i+1} erforderlich. Besonders: Führt immer zu Stabilität (unabhängig von der Schrittweite), wenn System stabil! (Eigenwerte anschauen!) ABER: oft auch für instabile Systeme stabil!

2.1.3 Trapezregel/Heun-Verfahren

Referenz auf SIEMENS SKRIPT Kombination der beiden Euler-Verfahren $\Rightarrow$ höhere Genauigkeit!

Mittelwert der Steigungen im i-ten und $i+1$-ten Schritt:

$$x_{i+1} = x_i + \Delta x_i = \tag{2.13}$$
$$= x_i + T \cdot \frac{1}{2} \cdot \Big(f\big(x_i, t\big) + f\big(x_{i+1}, t+T\big) \Big). \tag{2.14}$$

Dadurch ergibt sich allgemein ebenfalls ein implizites Verfahren. Dies kann nun also möglicherweise nichtlineares Gleichungssystem gelöst werden, oder durch Abschätzung des unbekannten Wertes $f\big(x_{i+1}, t+T\big)$ durch das explizite Euler-Verfahren erhält man das Heun-Verfahren nach [1]:

$$x_{i+1} = x_i + \Delta x_i =$$
$$= x_i + T \cdot \frac{1}{2} \cdot \Big(f\big(x_i, t\big) + f\big(x_i + T \cdot f(x_i, t), t+T\big) \Big).$$

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 gew. DGL (REF!), Zeitableitungen höherer Ordnung in die Abschätzung einzubauen. Dies führt zu den Taylor-Methoden, welche auf der Taylor-Satz basieren. Da die Differentation der Rechten Seite jedoch nach [1] 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.

Wie in [1] gezeigt wird, lässt sich der *globale* Fehler $e(t_i) := ||x_i - x(t_i)||$ gängiger Integrationsverfahren durch

$$e(t) = \mathcal{O}(T^p) \tag{2.15}$$

abschätzen. (Getroffene Annahmen über die rechte Seite der DGL (2.1) 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. 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

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

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

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

Runge-Kutta-Verfahren:

$$\Phi(x,t,T) = x + T\sum_{i=1}^{s} b_i\, k_i, \quad \text{mit} \tag{2.16}$$

$$k_i = f\left(t + c_i T, x_k + T\sum_{j=1}^{i-1} a_{ij} k_j\right) \quad \text{für } i = 1,\ldots,s. \tag{2.17}$$

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 des resultierendes Funktionswertes von x_{k+1} 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-Schemata: Die selbe Vorgehensweise lässt sich für implizite Verfahren anwenden, sodass

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \qquad \begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

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

sich als Berechnungsvorschrift für *s-stufige* implizite Runge-Kutta-Verfahren ergibt:

$$\Phi(x,t,T) = x + T\sum_{i=1}^{s} b_i\, k_i \quad \text{mit} \tag{2.18}$$

$$k_i = f\left(t + c_i T, x_k + T\sum_{j=1}^{s} a_{ij} k_j\right) \quad \text{für } i = 1,\ldots,s. \tag{2.19}$$

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 zeigen.

2.2 Schrittweitensteuerung (Adaption)

Um die Verfahren weiter zu verbessern, 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*

$\overline{\varepsilon}$ bezeichnet.

$$\overline{\varepsilon} := ||\hat{\Phi}(t_i, x_i, T_i) - \Phi(t_i, x_i, T_i)||. \tag{2.20}$$

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

$$T_{\text{neu}} = \sqrt[\hat{p}+1]{\frac{tol}{\overline{\varepsilon}}} T_{\text{alt}} \tag{2.21}$$

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

Nach der Berechnung des Fehlerschätzers $\overline{\varepsilon}$ im Zeitschritt i wird dieser mit der zulässigen Toleranz verglichen. Ist der berechnete Fehler größer, als die zulässige Toleranz ($\overline{\varepsilon} > tol$), wird der selbe Zeitschritt mit der angepassten Schrittweite T_{neu} erneut berechnet.
Gilt jedoch $\overline{\varepsilon} \leq tol$, so ist die gewünschte Genauigkeit erreicht und die der Berechnung zu Grunde liegende Schrittweite wird für diesen Zeitschritt angenommen. Nun wird erneut (2.21) 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 Verfahren werden mit RK$p(\hat{p})$ 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 dehalb 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 der Ordnung $\hat{s} = s + 1$ 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

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

und soll an dieser Stelle nicht weiter vertieft werden. Eines der am weitesten verbreiteten Verfahren 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 [1] nachgelesen werden.

2.4 Mehrschrittverfahren

Die bisher betrachteten Verfahren sind dadurch charakterisiert, dass die Verfahrensfunktion $\Phi(x,t)$ lediglich vom aktuellen Funktionswert x_k, sowie bei impliziten Verfahren dem Funktionswert x_{k+1} abhängt. Deshalb werden diese Verfahren als Einschrittverfahren bezeichnet.
Im Gegensatz dazu hängt der Funktionswert x_{k+1} bei Mehrschrittverfahren zusätzlich dazu von einer beliebigen Anzahl an Vorgängerwerten $x_{i-k+1}, \ldots, x_i$ ab. Damit wird ein k-stufiges lineares Mehrschrittverfahren folgendermaßen definiert:

$$\begin{aligned} &a_k x_{i+k} + a_{k-1} x_{i+k-1} + \ldots + a_0 x_i \\ &\quad = T\left(b_k f(x_{i+k}, t_{i+k}) + b_{k-1} f(x_{i+k-1}, t_{i+k-1}) + \ldots + b_0 f(x_i, t_i)\right) \end{aligned} \tag{2.22}$$

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

$$P_a(z) = a_0 + a_1 z + \ldots + a_k z^k \tag{2.23}$$

$$P_b(z) = b_0 + b_1 z + \ldots + b_k z^k \tag{2.24}$$

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

$$x_{i+1} = x_{i-1} + \frac{T}{3}\left(f(x_{i+1}, t_{i+1}) + 4f(x_i, t_i) + f(x_{i-1}, t_{i-1})\right) \tag{2.25}$$

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 mit starkem Rechenaufwand verbunden sein kann.
Dies führt zu dem Gedanken, Approximationen, welche durch ein explizites Verfahren gewonnen wurden, durch ein implizites Verfahren zu verbessern. Diese Kombination aus explizitem und implizitem Verfahren 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. (KOEFFIZIENTEN SIEHE ANHANG) Die Herleitung dieser Verfahren kann in [1] nachvollzogen 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 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-Verfahren* 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 Korrektor-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

(Besonders geeignet für steife DGL aber wo einbringen?) Eine weitere Klasse der Mehrschrittverfahren sind die so genannten *Backwards Differentiation Formulas* (BDF). Für diese Verfahren gilt $P_b(z) = z^k$. Damit lassen sich eine Reihe impliziter Verfahren erzeugen, die besonders gut für steife Differentialgleichungen geeignet sind.

2.5 Stabilitätsanalyse der wichtigsten Methoden

VIELLEICHT EINFACH: METHODEN AUS DIGITALE REGELUNG ANWENDEN, HILFE VON GRUENE UND NETOMAC SKRIPT! 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. Um die Betrachtungen zu vereinfachen, soll sich an dieser Stelle auf die Klasse *linearer zeitinvarianter Systeme* beschränkt werden, für die gilt:

$$\dot{x} = Ax. \tag{2.26}$$

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

2.5.1 Steife Differentialgleichungen

Zur Einführung soll das Anfangswertproblem

$$\dot{x} = \lambda x, \quad x(0) = x_0 \tag{2.27}$$

betrachtet werden. Für diese einfache Differentialgleichung lautet die geschlossene Lösung $x(t) = e^{\lambda t} x_0$. Das heißt für $\lambda < 0$ konvergiert diese Lösung gegen 0. Eine Approximation des Systems mit dem expliziten Euler-Verfahren liefert

$$x_{i+1} = x_i + T\lambda x_i = (1 + T\lambda) x_i. \tag{2.28}$$

Diese Differenzengleichung konvergiert genau dann ebenfalls gegen 0, wenn

$$|1 + T\lambda| < 1 \tag{2.29}$$

gilt. Für die Schrittweite T muss also die Bedingung

$$T < \frac{2}{|\lambda|} \tag{2.30}$$

erfüllt sein, damit die Approximation eine Sinnvolle Näherung darstellt. Es ist augenscheinlich, dass dies bei sehr schnellen Systemen mit $\lambda \ll 0$ zu sehr kleinen notwendigen Schrittweiten führt.
Aus diesem Grund werden solche Differentialgleichungen allgemein als *steif* bezeichnet. 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 2.1 ist die Approximation von Gleichung (2.27) mit dem expliziten Euler-Verfahren für verschiedene Schrittweiten dargestellt.

2.5.2 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 Differen-

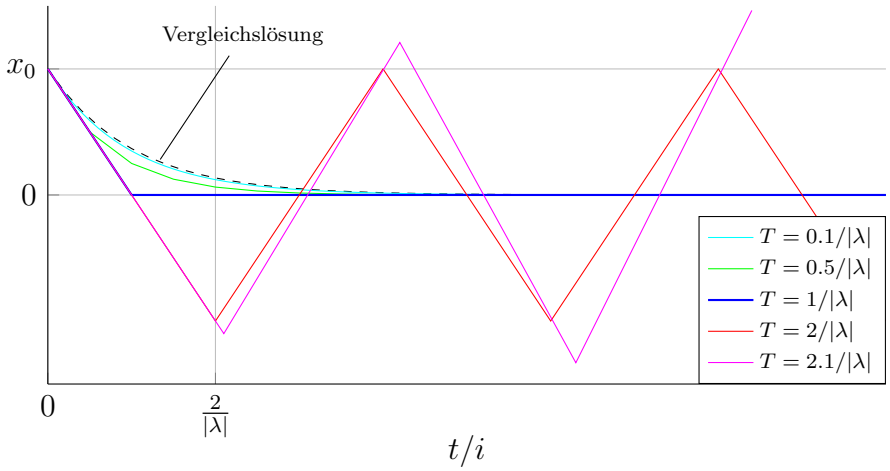


Abbildung 2.1: Simulation des Beispielsystems mit explizitem Euler-Verfahren

zengleichung

$$x_{i+1} = \tilde{A} x_i \tag{2.31}$$

beschreiben. Nach [2] ist eine solche Differenzengleichung genau dann exponentiell stabil, wenn für alle Eigenwerte der Matrix $\tilde{A}$ stabil sind, das heißt $|\tilde{\lambda}_i| < 1$ gilt. Dieser Bereich wird im Folgenden mit $B_1(0)$ bezeichnet. Zur Untersuchung der Stabilität eines Verfahrens müsste also die Matrix $\tilde{A}$ aus den Koeffizienten des Verfahrens und der Matrix A der zu Grunde liegenden Differentialgleichung berechnet werden. Die Matrix der Differenzengleichung $\tilde{A}$ berechnet sich also als Funktion der Art

$$\tilde{A} = R(TA). \tag{2.32}$$

Diese Funktion nimmt für explizite Werte $z \in \mathbb{C}$ die Form

$$R(z) = 1 + zb^T \left(I - z\mathcal{A}\right)^{-1} e \tag{2.33}$$

an, wobei I für die Einheitsmatrix steht, $b = (b_1, \ldots, b_s)^T$ und $\mathcal{A} = (a_{ij})_{i,j=1,\ldots,s}$ die Koeffizienten des Verfahrens beinhalten und $e = (1, \ldots, 1)^T \in \mathbb{R}^s$ gilt. $R(z)$ 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 A und T erfüllen müssen, damit die Eigenwerte der Matrix $\tilde{A}$ stabil sind.

Hierfür wird das Stabilitätsgebiet $S \subset \mathbb{C}$ eines Runge-Kutta-Verfahrens mit Stabilitäts-

funktion R definiert als die Menge von Eigenwerten λ_i, die TA annehmen darf, damit $\tilde{A} = R(TA)$ exponentiell stabil ist. Ein Verfahren wird als A-stabil bezeichnet, wenn es für eine stabile Dynamik A unabhängig von der Schrittweite stabil ist.

Das Stabilitätsgebiet entspricht dem Bereich, indem der Betrag der Stabilitätsfunktion kleiner eins ist:

$$S = \{z \in \mathbb{C} \big| |R(z)| < 1\}. \tag{2.34}$$

Damit lassen sich für alle betrachteten Verfahren Voraussetzungen an die Schrittweite in Abhängigkeit der Eigenwerte von A berechnen. Es gilt zu beachten, dass $\lambda_i \in \Sigma(A) \Rightarrow T\lambda_i \in \Sigma(TA)$. Allgemein gilt somit, je größer das Stabilitätsgebiet S, desto größer kann die Schrittweite T gewählt werden, um Stabilität zu gewährleisten.

Zur Vergleichbarkeit sollen nun die betrachteten Verfahren auf das Stabilitätsgebiet untersucht werden. Für das explizite Euler-Verfahren ergibt sich $R_{\mathrm{Eu,ex}}(z) = 1 + z$. Damit ergibt sich

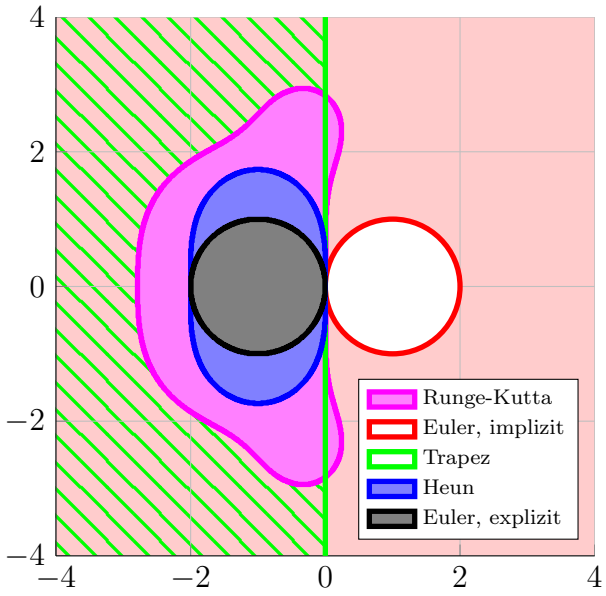
$$|R_{\mathrm{Eu,ex}}(z)| < 1 \;\Leftrightarrow\; |1 + z| < 1, \tag{2.35}$$

also $S = \{z \in \mathbb{C} \big| |1 + z| < 1\} = B_1(-1)$. Die Schrittweite muss also so gewählt werden, dass $T\lambda_i \in B_1(-1)$ gilt. Für das betrachtete Beispielsystem entspricht dies genau (2.30).

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

$$|R_{\mathrm{Eu,im}}(z)| < 1 \;\Leftrightarrow\; |1 - z| > 1, \tag{2.36}$$

das heißt S 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 $\Re z > 1$. 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.

2.6 Anwendung der Integrationsmethoden in Simulationssoftware

2.6.1 PSS Nettomac

2.6.2 PowerFactory

RMS Simulation Algorithms

•Highly accurate, fixed or variable step-size integration technique for solving AC and DC network load flow and dynamic model equations. This is combined with a non-linear electromechanical model representation to enable a high degree of solution accuracy, algorithmic stability and time range validity.

•A-stable simulation algorithm for the efficient handling of stiff systems. This is applicable to all or any individually selected model featuring error-controlled automatic step-size adaptation, ranging from milliseconds up to minutes or even hours, including precise handling of interrupts and discontinuities. EMT Simulation Algorithms

•The calculation of initial conditions is carried out prior to the EMT simulation, and is based on a solved load flow (symmetrical or asymmetrical). Consequently, there is no need for saving steady state conditions being reached after transients are damped out aiming in simulation re-starting under steady state conditions.

•Special numerical integration methods have been implemented in DIgSILENT PowerFactory in order to avoid numerical oscillations caused by switching devices and other non-linear characteristics.
•Highly accurate, fixed or variable step-size integration technique for solving AC and DC network transients and dynamic model equations. This is combined with a non-linear electromechanical model representation to enable a high degree of solution accuracy, algorithmic stability and time range validity.

2.6.3 PSSE

2.6.4 Eurostag

The advanced dynamic functions of EUROSTAG® allow for the full range of transient, mid and long-term stability to be covered thanks to a robust algorithm using an auto-adaptative integration stepsize.

Hier ist das eigentliche Thema zu bearbeiten.

3 Zusammenfassung und Ausblick

In der Zusammenfassung werden die Ergebnisse der Arbeit kurz zusammengefasst. Der Umfang beträgt ca. eine Seite.

A Überblick einiger Verfahren

A.1 Einschrittverfahren

Name	**Ordnung**	**Butcher-Schema**	**Art**	**Stabilität**	Stabilitätsfunktion/I
Euler	1	$\begin{array}{c\|c} 0 & \\ \hline & 1 \end{array}$	explizit	...	
Heun	2	$\begin{array}{c\|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$	explizit	...	
klassisches Runge-Kutta	4	$\begin{array}{c\|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$	explizit	...	
Euler	1	$\begin{array}{c\|c} 0 & \\ \hline & 1 \end{array}$	implizit	...	
Mittelpunktsregel	2	$\begin{array}{c\|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$	implizit	...	
Trapezregel	2	$\begin{array}{c\|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$	implizit	...	

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

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

A.3 Mehrschrittverfahren

k	**Name**	$P_a(z)$	$P_b(z)$	**Art**	**Stabilität**
2	Milne-Simpson	$z^2 - 1$	$\frac{1}{3}z^2 + \frac{4}{3}z + \frac{1}{3}$	implizit	...? (Sinn?)
1	Adams-Bashforth	$z^k - z^{k-1}$	1 ($\hat{=}$ Euler, explizit)	explizit	...
2			$(3z - 1)/2$		
3			$(23z^2 - 16z + 5)/12$		
4			$(55z^3 - 59z^2 + 37z - 9)/24$		
2	Adams-Moulton	$z^k - z^{k-1}$	$(5z^2 + 8z - 1)/12$	implizit	...
3			$(9z^3 + 19z^2 - 5z + 1)/24$		
4			$(251z^4 + 646z^3 + -246z^2 + 106z - 19)$		
1	BDF	$z - 1$ ($\hat{=}$ Euler, implizit)	z^k	implizit	unendlich großes Stabilitätsgebiet $\Rightarrow$ gut für steife DGL
2		$\frac{3}{2}z^2 - 2z + \frac{1}{2}$			
3		$\frac{11}{6}z^3 - 3z^2 + \frac{3}{2}z - \frac{1}{3}$			
4		$\frac{25}{12}z^4 - 4z^3 + 3z^2 - \frac{4}{3}z + \frac{1}{4}$			

B Anhang Teil 2

C Anhang Teil 3

Symbol- und Abkürzungsverzeichnis

Sollten in einer Ausarbeitung viele Abkürzungen und Formelzeichen auftreten, so empfiehlt es sich, diese gesondert in einem Kapitel aufzufuhren. Dieses kann auch nach dem Inhaltsverzeichnis (Abbildungs- und Tabellenverzeichnis) folgen.

A		Abkürzung für ...
$\cos\varphi$		Leistungsfaktor
U_s	V	Betrag der Statorspannung
φ	rad	Winkel zwischen Spannung und Strom

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*