Implizite Peer-Verfahren für große steife Systeme

Dissertation

zur Erlangung des
Doktorgrades der Naturwissenschaften (Dr. rer. nat.)

Der

Naturwissenschaftlichen Fakultät II
Chemie, Physik und Mathematik
der Martin-Luther-Universität
Halle-Wittenberg

vorgelegt

von Herrn Steffen Beck
geboren am 25.09.1982 in Hohenmölsen

Gutachter:

1. Prof. Dr. Rüdiger Weiner (MLU Halle-Wittenberg)
2. Prof. Dr. Jens Lang (TU Darmstadt)

Tag der Verteidigung: 21. November 2014

Danksagung

Mein außerordentlicher Dank gebührt meinem Betreuer Prof. Dr. Rüdiger Weiner, der mir die Möglichkeit zur Promotion gegeben hat und mich bei der Erstellung dieser Arbeit kontinuierlich und sehr intensive unterstützt hat. Seine Ratschläge und die zahlreichen anregenden Diskussionen waren mir eine große Hilfe. Außerdem bin ich für seine mir gegenüber aufgebrachte Geduld dankbar.

Weiterhin möchte ich mich bei Dr. Helmut Podhaisky bedanken, der stets ein offenes Ohr für meine Fragen hatte und mir mit seinen Fachkenntnissen stets zur Seite stand. Ebenfalls danken möchte ich Markus Köbis, der mir viele Hilfestellungen im Umgang mit dem LaTeX-Paket TikZ gegeben hat. Des Weiteren bedanke ich mich bei Dr. Lothar Boltze und Dr. Steffen Weber, die mir bei LaTeX Fragen stets weiterzuhelfen wussten. Allen Mitarbeiterinnen und Mitarbeitern der Arbeitsgruppe Numerik danke ich für das freundliche und kreative Arbeitsklima.

Ebenso bedanke ich mich bei Herrn Prof. Dr. Severiano González-Pinto (Universidad de La Laguna, Spanien), der mir die Forschungsaufenthalte in La Laguna (Teneriffa, Spanien) ermöglichte und mir seine Erfahrungen im Umgang mit der Approximierenden-Matrix-Faktorisierung (AMF) weitervermittelte. Seine wertvollen Hinweise flossen in die vorliegende Arbeit ein.

Außerdem möchte ich mich bei Frau Dr. Soledad Pérez-Rodríguez (Universidad de La Laguna, Spanien) bedanken, die bei der Umsetzung des Programmcodes RadauAMF in Matlab die entscheidende Vorarbeit leistete und mir bei der Implementierung des AMF-Lösers für das Test-Beispiel Radiation-Diffusion half. Dies diente mir als Orientierung für die Programmierung der anderen Test-Probleme mit AMF.

Meinen Eltern bin ich für ihre uneingeschränkte Unterstützung in allen erdenklichen Situationen außerordentlich dankbar. Frau Carolin Andersohn danke ich für ihren immerwährenden Beistand in allen Lebenslagen, sie war mir immer ein großer Rückhalt. Gerade für die Unterstützung in den schwierigen Phasen möchte ich ihr ganz besonders bedanken.

Inhaltsverzeichnis

1 Einleitung

In dieser Arbeit untersuchen wir sequentielle implizite Peer-Verfahren für steife Differentialgleichungssysteme sehr großer Dimension. Die Peer-Methoden sind Zweischritt-Verfahren bei denen die Stufenwerte $Y_{m,i}, i = 1, 2, \ldots, s$ aus zwei aufeinanderfolgenden Zeitschritten verwendet werden. Die Bezeichnung Peer besagt, dass die Stufenwerte alle die gleichen Genauigkeits- und Stabilitätseigenschaften besitzen. Die Stufenordnung ist daher gleich der Konsistenzordnung, wodurch die bei Einschrittmethoden auftretende Ordnungsreduktion vermieden wird.

Die Peer-Methoden wurden 2004 als parallele linear-implizite Verfahren eingeführt [40]. Zuerst wurden einfach implizite (*singly-implicit*) Methoden betrachtet, bei denen in den linearen Gleichungssystemen aller Stufen die gleiche Koeffizientenmatrix auftritt. Später stellte man fest, dass parallele mehrfach implizite (*multi-implicit*) Verfahren bessere Eigenschaften für steife Probleme besitzen, da sie steif genau sind. Diese wurden in [52, 43] ebenfalls als linear-implizite Variante, kombiniert mit dem Krylov-Verfahren FOM, an großen steifen Systemen getestet. In [42] wurde eine implizite Version dieser Methoden mit bis zu 10 Newton-Schritten betrachtet. Sequentielle einfach implizite Peer-Verfahren sind zuerst in [35] untersucht worden, auch diese betrachtete man als linear-implizite Variante. Eine Umsetzung dieser Methoden in Nordsieck-Form und eine Weiterentwicklung bezüglich des Prädiktors findet man in [36]. Später wurden diese Verfahren in die finite-Elemente-Software KARDOS [14] eingebaut [16].

Das Hauptaugenmerk dieser Arbeit liegt auf der Lösung großer steifer Systeme mit Hilfe der impliziten Peer-Verfahren. Die wichtigste Quelle solcher Probleme ist die Semidiskretisierung zeitabhängiger partieller Differentialgleichungen in zwei oder mehr Raumdimensionen mit Hilfe der Linienmethode (MOL). Aufgrund der Steifheit dieser Systeme sind im Allgemeinen implizite Methoden zu bevorzugen. Dadurch müssen jedoch nichtlineare und lineare Gleichungssysteme gelöst werden und infolge der hohen Dimensionen sind dafür spezielle Herangehensweisen erforderlich. Die nichtlinearen Systeme werden wir mit Hilfe des Newton-Verfahrens in eine Folge linearer Systeme überführen.

Zur Lösung der linearen Gleichungssysteme diskutieren wir zwei Ansätze, um den Aufwand erheblich zu reduzieren: Zum einen werden wir das Krylov-Unterraum-Verfahren FOM (*full orthogonalization method*) betrachten, bei dem mit Hilfe einer orthogonalen Projektion die Lösung des eigentlichen Problems hoher Dimension in einem Krylov-Unterraum sehr kleiner Dimension ($\varkappa \leq 20$) approximiert wird. Ein weiterer Vorteil dieser Vorgehensweise ist, dass die Matrix des resultierenden Gleichungssystems obere Hessenbergstruktur besitzt und somit unterhalb der ersten Nebendiagonale nur Nullelemente enthält. Bekannte Verfahren, die Krylov-Techniken verwenden, sind VODPK [10], ROWMAP [51] und EXP4 [22].

Zum anderen werden wir die approximierende Matrix-Faktorisierung (AMF) verwenden, bei der die Jacobi-Matrix in mehrere Teilmatrizen zerlegt wird und eine Reihe von linearen Gleichungssystemen entsteht, die strukturiert und somit sehr viel einfacher zu lösen sind. Da die Teilmatrizen meist Block-Diagonal-Gestalt besitzen, lässt sich die LU-Zerlegung sehr leicht berechnen, was zu erheblichen Zeitersparnissen führt. Man kann verschiedene Verfahren mit dieser Technik zur Lösung der linearen Gleichungssysteme kombinieren, z.B. findet man Rosenbrock-AMF-Methoden in [24] und eine Radau-IIA Methode mit AMF in [19].

Wir werden in dieser Arbeit sequentielle implizite Peer-Verfahren mit diesen beiden Lösungsstrategien für große lineare Gleichungssysteme kombinieren. Im zweiten Kapitel beschäftigen wir uns mit den theoretischen Grundlagen der impliziten Peer-Methoden. Dabei gehen wir zunächst auf das Verfahrensschema ein und wandeln die nichtlinearen impliziten Abhängigkeiten mit Hilfe des Newton-Verfahrens in lineare um. Danach bestimmen wir mittels Taylorentwicklung die Konsistenzbedingungen, welche dann zusammen mit der Nullstabilität die Konvergenz der Methoden garantieren. Um die Nullstabilität gewährleisten zu können, betrachten wir wie in [35, 4] die stärkere Eigenschaft der optimalen Nullstabilität. Dafür leiten wir ein einfaches lineares Gleichungssystem her, mit dem die Koeffizienten des Verfahrens leicht berechnet werden können, welche die optimale Nullstabilität garantieren. Zuletzt erhöhen wir mit Hilfe der Superkonvergenz die Konvergenzordnung für konstante Schrittweiten um eins und gehen auf die linearen Stabilitätseigenschaften der Methoden ein.

Eine Sonderrolle spielen dabei Verfahren, die eine sogenannte FSAL-Bedingung (*first same as last*) erfüllen. Bei diesem Typ von Verfahren übernimmt man den letzten Stufenwert des alten Zeitschritts mit in den aktuellen, d.h., man kopiert ihn sozusagen. Dieses Vorgehen hat zwei Vorteile: Zum einen reduziert sich der Aufwand des Verfahrens, da man die Berechnung der kopierten Stufe einspart und somit entspricht der Aufwand des Verfahrens mit FSAL-Eigenschaft dem eines $s-1$-stufigen Verfahrens ohne FSAL-Eigenschaft. Zum anderen begünstigt es die Bestimmung der Lösung an Zwischenpunkten (*dense output*), da man jetzt die Interpolationspolynome zweier aufeinanderfolgender Integrationsintervalle stetig verknüpfen kann, weil die Polynome an der Schnittstelle den gleichen Wert annehmen. Verwendet man die Hermite-Interpolation, erhält man sogar eine stetig differenzierbare Lösung, da die Ableitungen ebenfalls übereinstimmen. Es gibt allerdings auch zwei Nachteile dieses Vorgehens: Einige Koeffizienten des Verfahrens sind fest vorgegeben, damit besitzt das Verfahren weniger Freiheitsgrade. Weiterhin beeinflusst es die L-Stabilität der Methoden, die Verfahren sind jetzt nicht mehr automatisch L-stabil, sondern sie müssen eine zusätzliche Forderung erfüllen. Außerdem müssen die Superkonvergenz und die optimale Nullstabilität gesondert betrachtet werden. Wir werden Methoden dieser Art für drei und vier Stufen herleiten und Koeffizienten bestimmen.

In Kapitel 3 gehen wir auf die Koppelung der Peer-Verfahren mit FOM ein. Dazu betrachten wir Krylov-Unterräume und zeigen wie man mit Hilfe des Arnoldi-Verfahrens eine orthonormale Basis dieses Raumes berechnet. Rundungsfehler spielen dabei ein große Rolle, sie können dazu führen, dass die Orthogonalität der berechneten Vektoren bereits nach wenigen Iterationsschritten verloren geht. Wir führen an, wann das der Fall ist und wie man das Mittel der Nachorthogonalisierung nutzen kann, um diesem

Problem zu begegnen. Des Weiteren gehen wir darauf ein, wie man mit Hilfe der orthonormalen Basis die im Newton-Verfahren auftretenden linearen Gleichungssysteme lösen kann, indem man fordert, dass das Residuum des linearen Gleichungssystems senkrecht zum Krylov-Unterraum steht. Außerdem geben wir an, wie man das Residuum effizient berechnet, um die Iteration abbrechen zu können, wenn man eine geforderte Genauigkeit erreicht hat. Im zweiten Abschnitt dieses Kapitels beschreiben wir einen Evolutionsalgorithmus, mit dem wir die Koeffizienten, der im zweiten Kapitel beschriebenen einfach impliziten Peer-Verfahren, gesucht haben. Dabei wurden die Eigenschaften möglichst großer α-Winkel der $L(\alpha)$-Stabilität, möglichst kleine Fehlerkonstante und möglichst kleiner Wert γ optimiert. Die gefundenen Verfahren haben wir mit FOM gekoppelt und getestet. Die Methoden, welche sich als gut geeignet erwiesen, haben wir ausgewählt und in einer Tabelle angegeben. Des Weiteren stellen wir die Stabilitätsgebiete der gefundenen Verfahren grafisch dar. Im letzten Abschnitt dieses Kapitels gehen wir auf Fragen der Implementierung ein. Wir beschreiben, wie man die Startwerte berechnet, wie die Schrittweitensteuerung aussieht, wie wir den Prädiktor im Newton-Verfahren gewählt haben, welche Abbruchkritieren verwendet werden und wie wir die Jacobi-Matrix Vektor Produkte approximieren.

Im vierten Kapitel behandeln wir die Kombination der Peer-Methoden mit AMF. Als erstes klären wir, wie man die AMF-Iteration auf die linearen Gleichungssysteme im Newton-Verfahren anwendet. Danach gehen wir auf die Stabilität der Peer-AMF-Methoden anhand einer ganz bestimmten Testgleichung ein und zeigen, wie die Dimension des AMF-Splittings die $A(\alpha)$-Stabilität beeinflusst. Im Weiteren haben wir die in Kapitel 3 gefunden Koeffizienten der Peer-Methoden mit AMF kombiniert und getestet. Die Verfahren, welche bei den Tests am besten abschnitten, geben wir in einer Tabelle an. Im letzten Abschnitt dieses Kapitels gehen wir auf die Fragen der Implementierung dieser Verfahrensklasse ein. Es zeigte sich, dass die Konvergenz des Newton-Verfahrens mit AMF stark von der Wahl der Startwerte abhängt. Insbesondere haben sich Prädiktoren geringer Ordnung als vorteilhaft erwiesen. Wir geben einen Prädiktor an, mit dem wir gute Ergebnisse erzielen konnten. Des Weiteren gehen wir auf die für die Newton-AMF-Iteration wichtigen Abbruchkriterien ein.

In Kapitel 5 werden wir die konstruierten Methoden an steifen Systemen geringer und großer Dimension ausführlich testen. Für kleine Beispiele vergleichen wir mit den Matlab Standardintegratoren Ode15s und Ode23s. Ode15s liegt eine NDF-Methode (*numerical differentiation formula*) mit variabler Ordnung von 1 bis 5 zu Grunde und Ode23s basiert auf einer Rosenbrock-Methode der Ordnung 2. Bei den großen steifen Systemen, bei denen es sich um semidiskretisierte Reaktions-Diffusions-Gleichungen handelt, vergleichen wir mit den Krylov-Integratoren ROWMAP und Exp4. Dabei bildet bei ROWMAP eine Rosenbrock-Methode der Ordnung 4 die Grundlage und Exp4, welches ebenfalls die Ordnung 4 besitzt, gehört zur Klasse der exponentiellen W-Methoden. Außerdem verwenden wir Ode15s und einen RadauAMF-Code als Vergleichsverfahren. RadauAMF basiert dabei auf einer zweistufigen Radau-IIA Methode. Zuletzt werden wir die Peer-Verfahren mit FSAL-Eigenschaft mit den Peer-Verfahren ohne FSAL-Eigenschaft anhand großer steifer Systeme vergleichen.

Abschließend fassen wir die erzielten Ergebnisse zusammen und geben einen Ausblick auf mögliche weitere Untersuchungen.

2 Implizite Peer-Verfahren

2.1 Formulierung der Methoden

Zur numerischen Behandlung von Anfangswertproblemen der Form

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0, \quad t_0 \leq t \leq t_{end}, \quad y_0 \in \mathbb{R}^n, f\colon \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n \tag{2.1}$$

untersuchen wir Peer-Verfahren der Gestalt

$$Y_{m,i} = \sum_{j=1}^{s} b_{ij} Y_{m-1,j} + h_m \sum_{j=1}^{i} g_{ij} F_{m,j}, \quad i = 1, 2, \ldots, s. \tag{2.2}$$

In jedem Zeitschritt von t_m nach $t_{m+1} = t_m + h_m$ werden s Stufenwerte $Y_{m,i} \approx y(t_{m,i})$, $i = 1, 2, \ldots, s$, zu den Zeitpunkten

$$t_{m,i} := t_m + h_m c_i, \quad i = 1, 2, \ldots, s$$

berechnet. Die Ableitungen der Stufenwerte werden mit $F_{m,i} = f(t_{m,i}, Y_{m,i}) \approx y'(t_{m,i})$, $i = 1, 2, \ldots, s$, bezeichnet. Ordnet man die Stufen- und Funktionswerte in Vektoren der Form

$$Y_m = \begin{pmatrix} Y_{m,1} \\ Y_{m,2} \\ \vdots \\ Y_{m,s} \end{pmatrix} \in \mathbb{R}^{sn}, \qquad F(t_m, Y_m) = \begin{pmatrix} f(t_{m,1}, Y_{m,1}) \\ f(t_{m,2}, Y_{m,2}) \\ \vdots \\ f(t_{m,s}, Y_{m,s}) \end{pmatrix} \in \mathbb{R}^{sn}$$

an, so erhält man mit Hilfe des Kronecker-Produkts eine kompakte Darstellung des Verfahrens (2.2)

$$Y_m = (B_m \otimes I_n) Y_{m-1} + h_m (G_m \otimes I_n) F(t_m, Y_m). \tag{2.3}$$

Dabei ist I_n die n-dimensionale Einheitsmatrix. Die Koeffizienten des Verfahrens werden hierbei in den Matrizen $B_m = (b_{ij})_{i,j=1}^{s}$ und $G_m = (g_{ij})_{i,j=1}^{s}$, welche vom Schrittweitenquotienten

$$\sigma_m = h_m / h_{m-1} \tag{2.4}$$

abhängen können, zusammengefasst. B_m kann dabei vollbesetzt sein, G_m besitzt untere Dreiecksgestalt. Die Knoten c_i seien nicht abhängig von σ_m. Wir werden zwei Typen von Verfahren der Form (2.2) untersuchen:

1. Einfach implizite Verfahren:

$$G_m = G_0 + \gamma I_s. \tag{2.5}$$

2. Einfach implizite Verfahren mit FSAL (engl.: *first same as last*) Bedingung, welche wir mit dem Ansatz

$$G_m = G_0 + \gamma(I - \mathbb{1}e_1^T) \tag{2.6}$$

einführen.

Dabei ist G_0 jeweils eine strenge untere Dreiecksmatrix, $e_1 = (1, 0, \ldots, 0)^\top$ der erste Einheitsvektor und $\mathbb{1} = (1, 1, \ldots, 1)^\top$. Außerdem gilt $\gamma > 0$.

Bemerkung 2.1 [Mehrfach implizite Verfahren]
Mehrfach implizite Peer-Verfahren der Form

$$G_m = G_0 + \operatorname{diag}(\gamma_1, \gamma_2, \ldots, \gamma_s), \tag{2.7}$$

mit $\gamma_i > 0$ wurden in [4] untersucht. Sie haben den Vorteil, dass man mehr freie Parameter zur Verfügung hat. Für eine direkte Lösung der linearen Gleichungssysteme in (2.9) *ist dieser Verfahrenstyp eher ungeeignet, da in jeder Stufe eine andere Koeffizientenmatrix auftritt. Bei Krylov-Methoden hingegen ist der Aufwand mit dem bei einfach impliziten Verfahren vergleichbar.*

Um die Stufenwerte $Y_{m,i}$ in (2.2) zu berechnen, muss in jeder Stufe ein nichtlineares Gleichungssystem der Form

$$Y_{m,i} - g_{ii}h_m F_{m,i} = \sum_{j=1}^{s} b_{ij} Y_{m-1,j} + h_m \sum_{j=1}^{i-1} g_{ij} F_{m,j}, \quad i = 1, 2, \ldots, s$$

gelöst werden. Die Terme der rechten Seite sind in jeder Stufe bekannt und werden mit w_i bezeichnet. Das bedeutet, wir müssen die Lösung $Y_{m,i}$ der Gleichung

$$0 = Y_{m,i} - h_m g_{ii} F_{m,i} - w_i =: g(Y_{m,i}) \tag{2.8}$$

bestimmen. Dazu verwenden wir das Newton-Verfahren und erhalten eine Folge von linearen Gleichungssystemen

$$\begin{aligned} (I - \delta_i T_i)\, \Delta Y_{m,i}^k &= w_i - Y_{m,i}^k + \delta_i f\left(t_{m,i}, Y_{m,i}^k\right) \\ Y_{m,i}^{k+1} &= Y_{m,i}^k + \Delta Y_{m,i}^k \end{aligned}, \qquad k = 0, 1, 2, \ldots \tag{2.9}$$

mit $\delta_i = h_m g_{ii}$. Dabei ist T_i eine Approximation an die Jacobi-Matrix $f_y\left(t_{m,i}, Y_{m,i}^k\right)$. Im Normalfall werden diese linearen Gleichungssysteme mit direkten Lösern mit LU-Zerlegung gelöst. In diesem Fall verwendet man im Allgemeinen $T_i = f_y\left(t_{m-1,s}, Y_{m-1,s}\right)$ (vereinfachtes Newton-Verfahren). Da dieses Vorgehen bei Systemen hoher Dimension rechenzeitaufwendig und speicherintensiv ist, werden wir in den Kapiteln 3 und 4 als Alternative Approximationsverfahren zur Lösung linearen Gleichungssysteme in (2.9) untersuchen.

2.2 Ordnungsbedingungen

Um die Ordnungsbedingungen der Peer-Verfahren herzuleiten, betrachten wir das Residuum, das sich ergibt, wenn wir die exakte Lösung des Anfangswertproblems (2.1) in die Verfahrensvorschrift einsetzen. Wir erhalten für das Residuum der i-ten Stufe

$$\Delta_{m,i} = y(t_{m,i}) - \sum_{j=1}^{s} b_{ij} y(t_{m-1,j}) - h_m \sum_{j=1}^{i} g_{ij} y'(t_{m,j}), \qquad i = 1, 2, \ldots, s. \tag{2.10}$$

Definition 2.2 [Konsistenzordnung]
Die implizite Peer-Methode (2.2) *besitzt die Konsistenzordnung* $p \in \mathbb{N}$*, falls für alle hinreichend glatten Anfangswertprobleme der Form* (2.1) *für die Residuen*

$$\Delta_{m,i} = \mathcal{O}(h_m^{p+1}) \quad \textit{für} \quad h_m \to 0$$

für $i = 1, 2, \ldots, s$ *gilt.*

Da dies für alle Stufen gleich ist, gibt es keinen Unterschied zwischen der Ordnung des Verfahrens und der Stufenordnung. Die Verfahren sind deshalb sehr gut für steife Probleme geeignet, da keine Ordnungsreduktion auftritt.

Wenn wir in (2.10) die exakte Lösung durch ihre Taylorentwicklung ersetzen, erhalten wir Ordnungsbedingungen an die Koeffizienten des Verfahrens. Wir benötigen die folgenden Taylor-Entwicklungen, wobei das Schrittweitenverhältnis (2.4) zu beachten ist und wir voraussetzen, dass die Lösung hinreichend glatt ist:

$$\begin{aligned}
y(t_m + c_i h_m) &= \sum_{k=0}^{p} \frac{c_i^k h_m^k}{k!} y^{(k)}(t_m) + \mathcal{O}(h_m^{p+1}), \\
y(t_{m-1} + c_i h_{m-1}) = y(t_m + \frac{c_i - 1}{\sigma_m} h_m) &= \sum_{k=0}^{p} \frac{(c_i - 1)^k h_m^k}{k! \sigma_m^k} y^{(k)}(t_m) + \mathcal{O}(h_m^{p+1}), \\
h_m y'(t_m + c_i h_m) &= \sum_{k=1}^{p} \frac{c_i^{k-1} h_m^k}{(k-1)!} y^{(k)}(t_m) + \mathcal{O}(h_m^{p+1}).
\end{aligned}$$

Einsetzen in (2.10) liefert

$$\begin{aligned}
\Delta_{m,i} = \left(1 - \sum_{j=1}^{s} b_{ij}\right) y(t_m) + \sum_{k=0}^{p} \left(c_i^k - \sum_{j=1}^{s} b_{ij} \frac{(c_j - 1)^k}{\sigma_m^k} - k \sum_{j=1}^{i} g_{ij} c_j^{k-1} \right) \frac{h_m^k}{k!} y^{(k)}(t_m) \\
+ \mathcal{O}(h_m^{p+1}).
\end{aligned} \tag{2.11}$$

Wenn wir die Knoten c_i im Vektor $\boldsymbol{c} = (c_1, c_2, \ldots, c_s)^\top$ zusammenfassen, erhalten wir den folgenden

Satz 2.3 [Konsistenzordnung]
Wenn die Koeffizienten des Verfahrens (2.2) *die Bedingung*

$$\boldsymbol{c}^k = \frac{1}{\sigma_m^k} B_m (\boldsymbol{c} - \mathbb{1})^k + k G_m \boldsymbol{c}^{k-1}, \qquad k = 0, 1, \ldots, p, \tag{2.12}$$

mit $\boldsymbol{c}^k = \left(c_1^k, c_2^k, \ldots, c_s^k\right)^\top$ *für ein* $p \in \mathbb{N}$ *erfüllen, dann besitzt das Verfahren die Konsistenzordnung* p*. Dabei setzen wir hier* 0^0 *gleich 1.*

Die Vektoren $\boldsymbol{c}^k$ und $(\boldsymbol{c}-\mathbb{1})^k$ fassen wir für $k=0,1,\ldots,s-1$ in Matrizen zusammen und erhalten aus (2.12) für $p=s-1$ mit Hilfe der Bezeichnungen

$$S_m = \operatorname{diag}(1,\sigma_m,\sigma_m^2,\ldots,\sigma_m^{s-1}) \quad \text{und} \quad D_0 = \operatorname{diag}(0,1,\ldots,s-1)$$

das folgende Gleichungssystem

$$\begin{aligned}\begin{pmatrix} 1 & c_1 & \cdots & c_1^{s-1} \\ \vdots & & & \vdots \\ 1 & c_s & \cdots & c_s^{s-1} \end{pmatrix} &= B_m \begin{pmatrix} 1 & c_1-1 & \cdots & (c_1-1)^{s-1} \\ \vdots & & & \vdots \\ 1 & c_s-1 & \cdots & (c_s-1)^{s-1} \end{pmatrix} S_m^{-1} \\ &\quad + G_m \begin{pmatrix} 0 & 1 & c_1 & \cdots & c_1^{s-2} \\ \vdots & & & & \vdots \\ 0 & 1 & c_s & \cdots & c_s^{s-2} \end{pmatrix} D_0.\end{aligned} \tag{2.13}$$

Weiterhin bezeichnen wir

$$\begin{aligned}(V_0)_{i,j}^s &= c_i^{j-1} \\ (V_1)_{i,j}^s &= (c_i-1)^{j-1},\end{aligned} \quad D = \operatorname{diag}(1,2,\ldots,s) \quad \text{und} \quad (F_0)_{i,j}^s = \delta_{i,j+1} = \begin{cases} 1 & i=j+1 \\ 0 & \textit{sonst} \end{cases} \tag{2.14}$$

und erhalten damit für (2.13) die Darstellung

$$V_0 = B_m V_1 S_m^{-1} + G_m V_0 D F_0^\top. \tag{2.15}$$

Ist diese Gleichung erfüllt, besitzt die Peer-Methode die Konsistenzordnung $p=s-1$. Sind die Knoten c_i paarweise verschieden und damit die Vandermonde-Matrizen V_0 und V_1 regulär, können wir (2.15) nach

$$B_m = (V_0 - G_m V_0 D F_0^\top) S_m V_1^{-1} \tag{2.16}$$

umstellen. Somit ist die Matrix B_m für jedes G_m und für beliebiges σ_m eindeutig durch (2.16) bestimmt. Das Verfahren hat damit für variable Schrittweiten die Konsistenzordnung $p=s-1$.

Bemerkung 2.4 [General Linear Methods]

- *Die hier vorgestellten Peer-Methoden sind ein Spezialfall allgemeiner linearer Methoden (engl.: General Linear Methods). Einen Überblick über diese Verfahren findet man z.B. in [9, 25].*
- *Für die erste Spalte in* (2.13) *ergibt sich* $B\mathbb{1} = \mathbb{1}$. *Diese Bedingung wird auch als Präkonsistenzbedingung bezeichnet, vgl. [9, 25].*

2.3 Stabilität und Konvergenz

Definition 2.5 [Konvergenzordnung]
Das Peer-Verfahren (2.2) *besitzt die Konvergenzordnung* $p \in \mathbb{N}$ *falls für den globalen*

Fehler

$$\epsilon_m := Y_m - y(t_m + \boldsymbol{c}h_m) \qquad mit \qquad y(t_m + \boldsymbol{c}h_m) = \begin{pmatrix} y(t_m + c_1 h_m) \\ y(t_m + c_2 h_m) \\ \vdots \\ y(t_m + c_s h_m) \end{pmatrix} \tag{2.17}$$

für $h_{max} = \max_m h_m \to 0$

$$\epsilon_m = \mathcal{O}(h_{max}^p) \tag{2.18}$$

gilt.

Für den Nachweis der Konvergenz der Peer-Verfahren ist zusätzlich zur Konsistenz noch die Nullstabilität erforderlich.

Definition 2.6 [Nullstabilität]
Das Peer-Verfahren (2.2) *heißt nullstabil, wenn eine Konstante* $K < \infty$ *existiert, so dass für alle* $m \geq 0$ *und* $l \geq 0$

$$\|B_{m+l} B_{m+l-1} \cdots B_{m+1} B_m\| \leq K$$

gilt.

Dann gilt (vgl. [42])

Satz 2.7 [Konvergenz]
Jedes nullstabile Peer-Verfahren (2.2) *der Konsistenzordnung* $p \in \mathbb{N}$ *besitzt die Konvergenzordnung* p.

Die gleichmäßige Beschränktheit von Produkten der Koeffizienten-Matrizen, welche in Definition 2.6 verlangt wird, lässt sich oft nur schwer direkt nachweisen. Für spezielle Familien von Matrizen kann man sie allerdings leicht überprüfen. Wir konzentrieren uns deshalb wie in [35, 4] auf Methoden, welche optimal nullstabil sind (siehe Definition 2.8), da die Verfahren dann robuster gegenüber beliebigen Schrittweitenwechseln sind und in numerischen Tests gute Ergebnisse lieferten. Um solche Verfahren zu konstruieren, wird die Matrix

$$Q_m = V_1^{-1} B_m V_1 = P(I - V_0^{-1} G_m V_0 D F_0^\top) S_m \tag{2.19}$$

verwendet. Dabei ist P die Pascal-Matrix, welche durch

$$(P)_{ij}^s = \begin{cases} \binom{j-1}{i-1}, & i \leq j \\ 0, & i > j \end{cases}, \qquad P = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & \binom{s}{0} \\ 0 & 1 & 2 & 3 & & \binom{s}{1} \\ 0 & 0 & 1 & 3 & & \\ 0 & 0 & 0 & 1 & & \vdots \\ \vdots & & & \ddots & \ddots & \binom{s}{s-1} \\ 0 & & \cdots & & 0 & \binom{s}{s} \end{pmatrix} \tag{2.20}$$

definiert ist. Es gilt $P = V_1^{-1} V_0$. Im folgenden setzen wir voraus, dass $\|B_m\|$ für $\sigma_m \leq \sigma^* \in \mathbb{R}_+$ gleichmäßig beschränkt ist (Für $p = s - 1$ folgt dies aus (2.16)).

Definition 2.8 [Optimale Nullstabilität]
Das Peer-Verfahren (2.2) *heißt optimal nullstabil, wenn für die Matrix*

$$Q_m = V_1^{-1} B_m V_1 = \begin{pmatrix} 1 & * & \cdots & * \\ 0 & 0 & \ddots & \vdots \\ \vdots & & \ddots & * \\ 0 & 0 & \cdots & 0 \end{pmatrix} \tag{2.21}$$

gilt.

Damit erhalten wir folgenden

Satz 2.9 [Nullstabilität]
Wenn $\sigma_k \leq \sigma^* \in \mathbb{R}_+$ gilt, sind optimal nullstabile Peer-Verfahren nullstabil.

Beweis: Mit (2.21) gilt

$$B_{m+l} B_{m+l-1} \cdots B_m = V_1 Q_{m+l} Q_{m+l-1} \cdots Q_m V_1^{-1}. \tag{2.22}$$

Außerdem lässt sich Q_k durch die in (2.21) geforderte Form wie folgt zerlegen

$$Q_k = e_1 e_1^\top + U_k.$$

Dabei ist U_k eine strenge obere Dreiecksmatrix. Wegen $U_k e_1 = 0$ erhalten wir

$$\begin{aligned} Q_{m+s-1} Q_{m+s-2} \cdots Q_m &= (e_1 e_1^\top + U_{m+s-1}) Q_{m+s-2} \cdots Q_m \\ &= e_1 e_1^\top Q_{m+s-2} \cdots Q_m + U_{m+s-1}(e_1 e_1^\top + U_{m+s-2}) Q_{m+s-3} \cdots Q_m \\ &= e_1 e_1^\top Q_{m+s-2} \cdots Q_m + U_{m+s-1} U_{m+s-2} Q_{m+s-3} \cdots Q_m \\ &= \ldots \\ &= e_1 e_1^\top Q_{m+s-2} \cdots Q_m + U_{m+s-1} U_{m+s-2} \cdots U_m \\ &= e_1 e_1^\top Q_{m+s-2} \cdots Q_m, \end{aligned}$$

da die U_k strenge obere Dreiecksmatrizen sind und somit $U_{m+s-1} U_{m+s-2} \cdots U_m = 0$ gilt. Da bei optimaler Nullstabilität $Q_l e_1 = e_1$ ist, gilt dann

$$Q_{m+l} Q_{m+l-1} \cdots Q_m = e_1 e_1^\top Q_{m+s-2} Q_{m+s-3} \cdots Q_m$$

für alle $l \geq s-1$. Damit erhalten wir für alle $l \geq s-1$

$$\|B_{m+l} B_{m+l-1} \cdots B_{m+1} B_m\| = \|V_1 e_1 e_1^\top Q_{m+s-2} Q_{m+s-3} \cdots Q_m V_1^{-1}\|.$$

Somit lässt sich jedes beliebige Produkt von Matrizen B_k auf ein Produkt von höchstens $(s-1)$ Matrizen Q_k und den Matrizen V_1, V_1^{-1} und $e_1 e_1^\top$ reduzieren. Da für alle Schrittweitenquotienten $\sigma_k \leq \sigma^* \in \mathbb{R}_+$ gilt, ist auch $\|Q_k\| \leq \widetilde{K}$ für alle k. Wir erhalten somit für jede submultiplikative Matrixnorm

$$\begin{aligned} \|B_{m+l} B_{m+l-1} \cdots B_{m+1} B_m\| &\leq \|V_1\| \|e_1 e_1^\top\| \|Q_{m+s-2}\| \|Q_{m+s-3}\| \cdots \|Q_m\| \|V_1^{-1}\| \\ &\leq \|V_1\| \|e_1 e_1^\top\| \widetilde{K}^{s-1} \|V_1^{-1}\| \\ &\leq K \end{aligned}$$

□

Wir betrachten jetzt die Konstruktion von Verfahren, welche die Eigenschaft (2.21) besitzen sollen. Dazu schauen wir uns zunächst einmal die erste Spalte von Q_m an. Die Matrix Q_m ist definiert durch (2.19), es gilt

$$\begin{aligned} Q_m e_1 &= P(I - V_0^{-1} G_m V_0 D F_0^\top) S_m e_1 \\ &= P e_1 - P V_0^{-1} G_m V_0 D F_0^\top e_1 \\ &= P e_1 \qquad\qquad \text{da} \quad F_0^\top e_1 = 0 \\ &= e_1. \end{aligned}$$

Die erste Spalte von Q_m entspricht also für eine beliebige Matrix G_m immer e_1 und hat deshalb schon die in (2.21) gewünschte Form. D.h., wir müssen jetzt noch dafür sorgen, dass die Elemente von Q_m auf und unterhalb der Diagonale ab der zweiten Spalte Null sind. Dazu führen wir die folgenden Operatoren ein.

Definition 2.10 [$\mathbf{tril_2}(\cdot), \mathbf{tril}(\cdot)$]
Sei $A \in \mathbb{R}^{s,s}$, wir definieren den Operator $\mathrm{tril}_2 \colon \mathbb{R}^{s,s} \to \mathbb{R}^{s(s-1)/2}$ *durch*

$$\begin{aligned} \mathrm{tril}_2(A) &:= \sum_{j=2}^{s} \sum_{i=j}^{s} a_{ij} e_{k(i,j)} \\ \textit{mit} \qquad k(i,j) &= (j-2)(s-1) + (i-1) - \frac{(j-2)(j-1)}{2}. \end{aligned} \tag{2.23}$$

Dabei ist e_k ein $s(s-1)/2$-dimensionaler Einheitsvektor (wie man an (2.25) sieht), welcher an der k-ten Stelle eine Eins hat und sonst Nullen.

Wir definieren noch einen zweiten Operator $\mathrm{tril} \colon \mathbb{R}^{s,s} \to \mathbb{R}^{s(s+1)/2}$ *durch*

$$\begin{aligned} \mathrm{tril}(A) &:= \sum_{j=1}^{s} \sum_{i=j}^{s} a_{ij} e_{\ell(i,j)} \\ \textit{mit} \qquad \ell(i,j) &= (j-1)s + i - \frac{(j-1)j}{2}, \end{aligned} \tag{2.24}$$

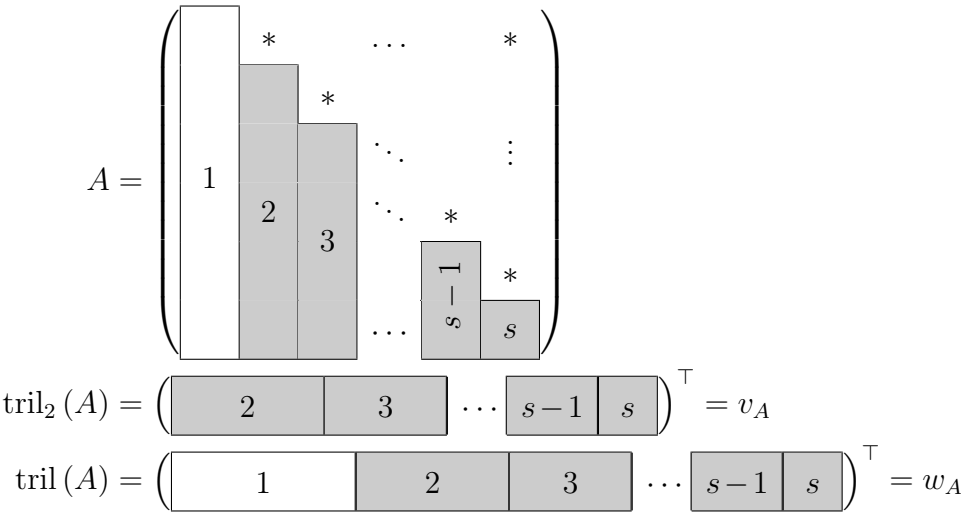
e_ℓ ist in diesem Fall ein $s(s+1)/2$-dimensionaler Einheitsvektor.

Der Operator tril_2 erzeugt einen Vektor v_A, der die Elemente der Matrix A enthält, welche auf und unterhalb der Diagonale von der zweiten bis zur letzten Spalte stehen. Der von tril erzeugte Vektor w_A enthält s zusätzliche Elemente, da er zusätzlich die erste Spalte von A berücksichtigt.

$$\begin{aligned} v_A &= (a_{22}, a_{32}, \ldots, a_{s2}, a_{33}, a_{43}, \ldots, a_{s3}, a_{44}, a_{54}, \ldots, a_{ss})^\top \\ w_A &= (a_{11}, a_{21}, \ldots, a_{s1}, a_{22}, a_{32}, \ldots, a_{s2}, a_{33}, a_{43}, \ldots, a_{ss})^\top \end{aligned}$$

Um besser zu verstehen, welche Elemente einer $(s \times s)$-Matrix durch die Operatoren $\mathrm{tril}_2(\cdot)$ und $\mathrm{tril}(\cdot)$ in einen Vektor geschrieben werden, wollen wir hier eine grafische

Veranschaulichung angeben:



Mit Hilfe dieser Darstellung kann man auch leicht ausrechnen, wieviele Elemente v_A und w_A enthalten, bzw. welche Dimension sie haben

$$\dim v_A = \sum_{i=1}^{s-1} i = \frac{s\,(s-1)}{2}, \qquad \dim w_A = \sum_{i=1}^{s} i = \frac{s\,(s+1)}{2}. \tag{2.25}$$

Lemma 2.11
Sei $A \in \mathbb{R}^{s,s}$, dann gilt

$$\operatorname{tril}(A) = Z^\top \sum_{j=1}^{s} (e_j \otimes A e_j), \tag{2.26}$$

dabei sind die e_j s-dimensionale Einheitsvektoren und $Z \in \mathbb{R}^{(s^2, s(s+1)/2)}$ ist gegeben durch:

$$Z = \begin{pmatrix} Z_1 & & & \\ & Z_2 & & \\ & & \ddots & \\ & & & Z_s \end{pmatrix}, \qquad \begin{aligned} Z_j &= \left(e_j e_{j+1} \dots e_s\right) \\ Z_j &\in \mathbb{R}^{s,s-j+1} \\ j &= 1, 2, \dots, s \end{aligned} \tag{2.27}$$

Die $e_i, i = j, j+1, \dots, s$ sind wieder s-dimensionale Einheitsvektoren.

Beweis: Nach Definition 2.10 ist

$$\operatorname{tril}(A) = \sum_{j=1}^{s} \sum_{i=j}^{s} a_{ij} e_{k(i,j)}, \qquad k(i,j) = (j-1)\,s + i - \frac{(j-1)\,j}{2}. \tag{2.28}$$

Wir betrachten jetzt nur die innere Summe über i und halten j fest, dabei gilt $e_{k(i,j)} \in$

$\mathbb{R}^{s(s+1)/2}$

$$\sum_{i=j}^{s} a_{ij} e_{k(i,j)} = (\underbrace{0,0,\ldots,0}_{n_{1j}\text{Nullen}}, a_{jj}, a_{j+1,j}, \ldots, a_{sj}, \underbrace{0,0,\ldots,0}_{n_{2j}\text{Nullen}})^\top =: v_j \in \mathbb{R}^{s(s+1)/2} \tag{2.29}$$

$$n_{1j} = s + (s-1) + \cdots + (s-j+2) = \sum_{\ell=1}^{j-1} s - \ell + 1$$

$$n_{2j} = 1 + 2 + \cdots + (s-j) = \sum_{\ell=1}^{s-j} \ell.$$

Den Vektor $(a_{jj}, a_{j+1,j}, \ldots, a_{sj})$ erhält man, indem man die transponierte j-te Spalte von A mit der Matrix Z_j multipliziert

$$(a_{jj}, a_{j+1,j}, \ldots, a_{sj}) = (Ae_j)^\top Z_j.$$

Jetzt muss man sich noch überlegen, wie man die n_{1j} Nullen davor und die n_{2j} Nullen danach erzeugt. Dazu nehmen wir den Vektor

$$z^\top = (0, 0, \ldots, 0) \in \mathbb{R}^s$$

her, welcher aus s Nullen besteht. Mit Hilfe der Matrizen $Z_1, Z_2, \ldots, Z_{j-1}$ kann man jetzt die Nullen davor und mit den Matrizen $Z_{j+1}, Z_{j+2}, \ldots, Z_s$ die Nullen danach erzeugen, denn

$$\dim\left(z^\top Z_k\right) = s - k + 1, \qquad k = 1, 2, \ldots, s, \quad k \neq j.$$

Den Vektor v_j aus Gleichung (2.29) kann man jetzt wie folgt schreiben:

$$\begin{aligned} v_j &= \left(z^\top Z_1, z^\top Z_2, \ldots, z^\top Z_{j-1}, (Ae_j)^\top Z_j, z^\top Z_{j+1}, z^\top Z_{j+2}, \ldots, z^\top Z_s\right)^\top \\ &= \left(\left(z^\top, z^\top, \ldots, z^\top, (Ae_j)^\top, z^\top, z^\top, \ldots, z^\top\right) Z\right)^\top \\ &= \left(\left(e_j^\top \otimes (Ae_j)^\top\right) Z\right)^\top \\ &= Z^\top \left(e_j \otimes Ae_j\right). \end{aligned}$$

Jetzt ersetzen wir die innere Summe in (2.28) durch den Vektor v_j und erhalten die Behauptung. □

Lemma 2.12
Die Operatoren $\mathrm{tril}(\cdot)$ *und* $\mathrm{tril}_2(\cdot)$ *sind linear und es gilt der Zusammenhang*

$$\mathrm{tril}_2(A) = \mathrm{tril}(M^\top A M), \tag{2.30}$$

mit $A \in \mathbb{R}^{s,s}$ *und*

$$M = \left(e_2 e_3 \ldots e_s\right) \in \mathbb{R}^{s,s-1} \quad \textit{mit} \quad \begin{array}{l} e_i, i = 2, 3, \ldots, s \\ \textit{s-dimensionaler Einheitsvektor} \end{array} \tag{2.31}$$

Beweis: Man rechnet leicht nach, dass die beiden Operatoren linear sind. Sei M durch (2.31) gegeben und $A \in \mathbb{R}^{s,s}$, dann gilt:

$$\mathrm{tril}_2(A) = \sum_{j=2}^{s}\sum_{i=j}^{s} a_{ij} e_{k(i,j)} = (a_{22}, a_{32}, \ldots, a_{s2}, a_{33}, a_{43}, \ldots, a_{s3}, a_{44}, a_{54}, \ldots, a_{ss})^\top . \tag{2.32}$$

Das sind alles Elemente der Untermatrix von A, welche man erhält, wenn man die erste Spalte und die erste Zeile streicht. Genau das macht die Matrix M

$$M^\top A M = \widehat{A} = \begin{pmatrix} a_{22} & a_{23} & \cdots & a_{2s} \\ a_{32} & a_{33} & \cdots & a_{3s} \\ \vdots & \vdots & & \vdots \\ a_{s2} & a_{s3} & \cdots & a_{ss} \end{pmatrix} \in \mathbb{R}^{s-1,s-1}.$$

Schauen wir uns jetzt an, was wir erhalten, wenn wir $\mathrm{tril}(\cdot)$ auf diese Matrix anwenden. Zu beachten ist dabei, dass die Matrix $\widehat{A} \in \mathbb{R}^{s-1,s-1}$ ist

$$\begin{aligned} \mathrm{tril}(M^\top A M) &= \sum_{j=1}^{s-1}\sum_{i=j}^{s-1} \left(M^\top A M\right)_{ij} e_{k(i,j)} = \sum_{j=1}^{s-1}\sum_{i=j}^{s-1} \widehat{a}_{ij} e_{k(i,j)} \\ &= (a_{22}, a_{32}, \ldots, a_{s2}, a_{33}, a_{43}, \ldots, a_{s3}, a_{44}, a_{54}, \ldots, a_{ss})^\top . \end{aligned} \tag{2.33}$$

Wenn wir jetzt (2.32) und (2.33) vergleichen, sehen wir, dass die beiden Vektoren übereinstimmen und somit die Behauptung folgt. □

Lemma 2.13
Sei $a, b \in \mathbb{R}^s$ und Z wie in Lemma 2.11, dann gilt

$$\mathrm{tril}(ab^\top) = Z^\top (b \otimes a). \tag{2.34}$$

Beweis:

$$\begin{aligned} \mathrm{tril}(ab^\top) &\underset{(2.26)}{=} Z^\top \sum_{j=1}^{s} e_j \otimes (ab^\top e_j) = Z^\top \sum_{j=1}^{s} e_j \otimes a b_j \\ &= Z^\top \sum_{j=1}^{s} b_j e_j \otimes a = Z^\top (b \otimes a) \end{aligned}$$

□

Jetzt können wir die Bedingung (2.21) mit Hilfe des Operators $\mathrm{tril}_2(\cdot)$ schreiben. Die Forderung (2.21) ist erfüllt, wenn gilt:

$$\mathrm{tril}_2(Q_m) = 0. \tag{2.35}$$

Wie man an Gleichung (2.19) sieht, lassen sich die Koeffizienten B_m und somit die Matrix Q_m durch die Wahl der Koeffizienten G_m direkt beeinflussen. Wir leiten jetzt Bedingungen für G_m her, so dass (2.35) erfüllt ist. Mit (2.19) folgt.

$$\mathrm{tril}_2(Q_m) = \mathrm{tril}_2(P(I - V_0^{-1} G_m V_0 D F_0^\top) S_m) = 0 \tag{2.36}$$

Da S_m eine Diagonalmatrix ist und somit in dem Gesamtprodukt die Spalten mit Potenzen von σ_m skaliert sind, ist (2.36) unabhängig von S_m und damit von σ_m. Somit gilt:

$$\begin{aligned} 0 &= \mathrm{tril}_2(Q_m) \\ &= \mathrm{tril}_2(P(I - V_0^{-1} G_m V_0 D F_0^\top)) \\ &= \mathrm{tril}_2(P) - \mathrm{tril}_2(P V_0^{-1} G_m V_0 D F_0^\top) \\ \Rightarrow \qquad \mathrm{tril}_2(I) &= \mathrm{tril}_2(P V_0^{-1} G_m V_0 D F_0^\top), \qquad \text{da} \quad \mathrm{tril}_2(P) = \mathrm{tril}_2(I). \end{aligned} \tag{2.37}$$

Aus der Bedingung (2.37) können die Koeffizienten von G_0 bestimmt werden. Wir betrachten zur Illustration den Fall einfach impliziter Verfahren. G_m hat dabei die Form:

$$G_m = G_0 + \gamma I.$$

Dies setzen wir in (2.37) ein und erhalten:

$$\begin{aligned} \mathrm{tril}_2(I) &= \mathrm{tril}_2(P V_0^{-1} \left(G_0 + \gamma I\right) V_0 D F_0^\top) \\ &= \mathrm{tril}_2(P V_0^{-1} G_0 V_0 D F_0^\top) + \mathrm{tril}_2(\gamma P D F_0^\top). \end{aligned}$$

Da $PDF_0^\top$ eine strenge obere Dreiecksmatrix ist, gilt $\mathrm{tril}_2(\gamma P D F_0^\top) = 0$. Die optimale Nullstabilität ist also in diesem Fall unabhängig von γ und wird nur durch G_0 bestimmt. Die Bedingung (2.35) lautet jetzt

$$\mathrm{tril}_2(I) = \mathrm{tril}_2(P V_0^{-1} G_0 V_0 D F_0^\top). \tag{2.38}$$

Um diese Forderung noch etwas übersichtlicher zu schreiben, ersetzen wir PV_0^{-1} durch V_1^{-1} und führen $\widehat{V_0}$ ein:

$$\widehat{V_0} := V_0 D F_0^\top = \begin{pmatrix} 0 & 1 & 2c_1 & 3c_1^2 & \cdots & (s-1)c_1^{s-2} \\ 0 & 1 & 2c_2 & 3c_2^2 & \cdots & (s-1)c_2^{s-2} \\ \vdots & & & & & \vdots \\ 0 & 1 & 2c_s & 3c_s^2 & \cdots & (s-1)c_s^{s-2} \end{pmatrix}. \tag{2.39}$$

Damit erhalten wir die folgende Bedingung für die optimale Nullstabilität:

$$\mathrm{tril}_2(I) = \mathrm{tril}_2(V_1^{-1} G_0 \widehat{V_0}). \tag{2.40}$$

Da G_0 eine strenge untere Dreiecksmatrix ist, enthält sie $s(s-1)/2$ Unbekannte g_{ij}, genau so viele Bedingungen ergeben sich auch aus Gleichung (2.38), wie man an (2.25)

sieht. Wir wollen jetzt aus (2.38) ein lineares Gleichungssystem für die einzelnen Elemente g_{ij} erzeugen. Dazu zerlegen wir G_0 wie folgt:

$$G_0 = \sum_{j=1}^{s-1} \sum_{i=j+1}^{s} g_{ij} E_{ij} \qquad \text{mit} \qquad E_{ij} = e_i e_j^\top \in \mathbb{R}^{s,s}.$$

E_{ij} ist eine Matrix, bei der in der i-ten Zeile an der j-ten Stelle eine Eins steht und alle anderen Elemente Null sind. Diese Zerlegung von G_0 setzen wir in (2.40) ein und erhalten:

$$\begin{aligned}
\operatorname{tril}_2(I) &= \operatorname{tril}_2(V_1^{-1} G_0 \widehat{V}_0) = \operatorname{tril}_2(V_1^{-1} (\sum_{j=1}^{s-1} \sum_{i=j+1}^{s} g_{ij} E_{ij}) \widehat{V}_0) \\
&= \operatorname{tril}_2(\sum_{j=1}^{s-1} \sum_{i=j+1}^{s} g_{ij} V_1^{-1} E_{ij} \widehat{V}_0) = \sum_{j=1}^{s-1} \sum_{i=j+1}^{s} g_{ij} \operatorname{tril}_2(V_1^{-1} E_{ij} \widehat{V}_0) \\
&= \sum_{j=1}^{s-1} \sum_{i=j+1}^{s} g_{ij} \operatorname{tril}_2(V_1^{-1} e_i e_j^\top \widehat{V}_0).
\end{aligned}$$

In der äußeren Summe läuft j von 1 bis $s-1$, d.h. von $\widehat{V}_0$ werden nur die Zeilen 1 bis $s-1$ verwendet. In der inneren Summe läuft i jeweils von $j+1$ bis s, demzufolge werden nur die Spalten 2 bis s von V_1^{-1} benötigt. Wir betrachten deshalb nur $\overline{M}^\top \widehat{V}_0$ und $V_1^{-1} M$ wobei

$$\overline{M} = \big(e_1 e_2 \dots e_{s-1}\big) \in \mathbb{R}^{s,s-1} \quad \text{mit} \quad \begin{array}{l} e_i, i = 1, 2, \dots, s-1 \\ s\text{-dimensionaler Einheitsvektor} \end{array}$$

ist. Dabei läuft die innere Summe dann von j bis $s-1$ und die Dimension der Einheitsvektoren e_i, e_j verringert sich um eins, d.h. $e_i, e_j \in \mathbb{R}^{s-1}$. Wir erhalten also

$$\begin{aligned}
\sum_{j=1}^{s-1} \sum_{i=j+1}^{s} g_{ij} \operatorname{tril}_2(V_1^{-1} e_i e_j^\top \widehat{V}_0) &= \sum_{j=1}^{s-1} \sum_{i=j}^{s-1} g_{i+1,j} \operatorname{tril}_2(V_1^{-1} M e_i e_j^\top \overline{M}^\top \widehat{V}_0) \\
&\overset{\text{Lemma}}{\underset{2.12}{=}} \sum_{j=1}^{s-1} \sum_{i=j}^{s-1} g_{i+1,j} \operatorname{tril}((M^\top V_1^{-1} M) e_i e_j^\top (\overline{M}^\top \widehat{V}_0 M)).
\end{aligned}$$

Wir führen jetzt neue Bezeichnungen ein, wir betrachten im Folgenden die Matrizen:

$$V_2 := M^\top V_1^{-1} M \qquad V_1^{-1} = \left(\begin{array}{c|ccc} * & * & \cdots & * \\ \hline * & & & \\ \vdots & & V_2 & \\ * & & & \end{array}\right) \qquad D_1 := \operatorname{diag}(1,2,\ldots,s-1) = \begin{pmatrix} 1 & & & \\ & 2 & & \\ & & \ddots & \\ & & & s-1 \end{pmatrix}$$

$$W := (\overline{M}^\top V_0 \overline{M})^\top = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ c_1 & c_2 & \cdots & c_{s-1} \\ \vdots & \vdots & & \vdots \\ c_1^{s-2} & c_2^{s-2} & \cdots & c_{s-1}^{s-2} \end{pmatrix}$$

$$\Rightarrow \widehat{W} := D_1 W = (\overline{M}^\top \widehat{V}_0 M)^\top = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 2c_1 & 2c_2 & \cdots & 2c_{s-1} \\ \vdots & \vdots & & \vdots \\ (s-1)c_1^{s-2} & (s-1)c_2^{s-2} & \cdots & (s-1)c_{s-1}^{s-2} \end{pmatrix}.$$

Damit erhalten wir:

$$\begin{aligned}
&\sum_{j=1}^{s-1}\sum_{i=j}^{s-1} g_{i+1,j}\,\operatorname{tril}((M^\top V_1^{-1} M) e_i e_j^\top (\overline{M}^\top \widehat{V}_0 M)) \\
&= \sum_{j=1}^{s-1}\sum_{i=j}^{s-1} g_{i+1,j}\,\operatorname{tril}(V_2 e_i e_j^\top (\widehat{W})^\top)) \overset{\text{Lemma 2.13}}{=} \sum_{j=1}^{s-1}\sum_{i=j}^{s-1} g_{i+1,j} Z^\top (\widehat{W} e_j \otimes V_2 e_i) \\
&= Z^\top \sum_{j=1}^{s-1} g_{j+1,j}(\widehat{W} e_j \otimes V_2 e_j) + g_{j+2,j}(\widehat{W} e_j \otimes V_2 e_{j+1}) + \cdots + g_{s,j}(\widehat{W} e_j \otimes V_2 e_{s-1}) \\
&= Z^\top \sum_{j=1}^{s-1} (\widehat{W} e_j \otimes I_{s-1}) \left[g_{j+1,j}(1 \otimes V_2 e_j) + g_{j+2,j}(1 \otimes V_2 e_{j+1}) + \cdots + g_{s,j}(1 \otimes V_2 e_{s-1})\right] \\
&= Z^\top \sum_{j=1}^{s-1} (\widehat{W} e_j \otimes I_{s-1}) \left[g_{j+1,j} V_2 e_j + g_{j+2,j} V_2 e_{j+1} + \cdots + g_{s,j} V_2 e_{s-1}\right] \\
&= Z^\top \sum_{j=1}^{s-1} (\widehat{W} e_j \otimes I_{s-1}) \left[V_2 Z_j (g_{j+1,j}, g_{j+2,j}, \ldots, g_{s,j})^\top\right] \\
&= Z^\top \Big[(\widehat{W} e_1 \otimes I_{s-1})(V_2 Z_1 (g_{2,1}, g_{3,1}, \ldots, g_{s,1})^\top) \\
&\qquad + (\widehat{W} e_2 \otimes I_{s-1})(V_2 Z_2 (g_{3,2}, g_{4,2}, \ldots, g_{s,2})^\top) + \cdots + (\widehat{W} e_{s-1} \otimes I_{s-1})(V_2 Z_{s-1} g_{s,s-1})^\top\Big] \\
&= Z^\top (\widehat{W} \otimes I_{s-1})(I_{s-1} \otimes V_2) Z \underbrace{(g_{2,1}, g_{3,1}, \ldots, g_{s,1}, g_{3,2}, g_{4,2}, \ldots, g_{s,2}, g_{4,3}, g_{5,3}, \ldots, g_{s,s-1})^\top}_{=: v_{G_0}} \\
&= Z^\top (\widehat{W} \otimes V_2) Z v_{G_0} = Z^\top (D_1 W \otimes V_2) Z v_{G_0}.
\end{aligned}$$

Das bedeutet, die Gleichung (2.38) ist erfüllt und damit die Forderung (2.21), wenn die Elemente g_{ij} von G_0 der Gleichung

$$\operatorname{tril}_2(I_s) = Z^\top (D_1 W \otimes V_2) Z v_{G_0} \tag{2.41}$$

genügen. Die Gleichung (2.41) stellt ein lineares Gleichungssystem für die Elemente g_{ij} mit $i > j$ der Matrix G_0 dar, welche aus diesem berechnet werden.

Bemerkung 2.14
Die beiden anderen Fälle (mehrfach implizit und FSAL) liefern ähnliche Gleichungssysteme, in die allerdings die Parameter γ bzw. γ_i eingehen und welche somit etwas schwieriger zu behandeln sind.

Lineare Stabilität

Zur Untersuchung der linearen Stabilität betrachten wir die Dahlquist'sche Testgleichung

$$y'(t) = \lambda y(t), \qquad \lambda \in \mathbb{C} \quad \text{mit} \quad \operatorname{Re}\lambda \leq 0.$$

Wendet man darauf das Peer-Verfahren (2.2) mit konstanter Schrittweite h an, so erhält man

$$Y_m = BY_{m-1} + hG\lambda Y_m \qquad \text{und} \qquad Y_m = (I_s - zG)^{-1}BY_{m-1} = M(z)Y_{m-1}$$

mit $z = h\lambda$ und der Stabilitätsmatrix des Peer-Verfahrens

$$M(z) = (I_s - zG)^{-1}B. \tag{2.42}$$

Definition 2.15 [Stabilitätsgebiet, A- und L-Stabilität]
Sei $M(z)$ die Stabilitätsmatrix bei konstanter Schrittweite. Dann heißt die Menge

$$\mathcal{S} = \left\{\overline{z \in \mathbb{C} : \varrho(M(z)) < 1}\right\} \tag{2.43}$$

Stabilitätsgebiet *des Peer-Verfahrens. Die Methode heißt* A-stabil *bzw.* A(α)-stabil *mit $\alpha \in (0, \frac{\pi}{2})$, wenn*

$$\{z \in \mathbb{C} : \operatorname{Re} z \leq 0\} \subset \mathcal{S}, \qquad \textit{bzw.} \qquad \{z \in \mathbb{C} : |\arg(z) - \pi| \leq \alpha\} \subset \mathcal{S}$$

gilt. Dabei gilt $\arg(z) \in [0, 2\pi)$.

Ein A-stabil *bzw.* A(α)-stabil *Verfahren heißt* L-stabil *bzw.* L(α)-stabil, *wenn*

$$\lim_{z\to\infty} \varrho(M(z)) = 0$$

gilt.

Bemerkung 2.16 [$L(\alpha)$-Stabilität]
Für $z \to \infty$ erhält man für $g_{ii} > 0$

$$\lim_{z\to\infty} M(z) = \lim_{z\to\infty} (I_s - zG)^{-1} B = \lim_{z\to\infty} \frac{1}{z}\left(\frac{1}{z}I_s - G\right)^{-1} B = 0.$$

D.h., eine $A(\alpha)$-stabile Peer-Methode mit $g_{ii} > 0, i = 1, 2, \ldots, s$ ist $L(\alpha)$-stabil.

Superkonvergenz

Optimal nullstabile einfach implizite Peer-Verfahren mit konstanter Matrix G besitzen nicht genügend freie Parameter, um die Konsistenzbedingung (2.12) bis zur Ordnung s zu erfüllen, auch nicht für konstante Schrittweiten. Wenn jedoch die Residuen (2.10) eine spezielle Struktur haben, kann man eine Bedingung angeben, sodass die Methode mit der Ordnung $p = s$ für konstante Schrittweiten konvergiert. Dieses Konzept ist gut untersucht und hat verschiedene Namen: Quasi-Konsistenz [28, 45], effektive Ordnung [8] oder Superkonvergenz [53].

Für einfach implizite Peer-Verfahren wurde diese Eigenschaft in [4] untersucht. Man erhält den folgenden

Satz 2.17 [Superkonvergenz]
Es sei ein optimal nullstabiles einfach implizites Peer-Verfahren der Ordnung $p = s-1$ der Form (2.2) gegeben, dessen Matrix G_m die Gestalt (2.5) besitzt. Die Knoten c_i seien paarweise verschieden und die Matrix B_m ergebe sich aus (2.16). Außerdem gelte $c_s = 1$. Dann ist das Verfahren genau dann konvergent von der Ordnung $p = s$ für konstante Schrittweiten, wenn γ eine Nullstelle des Polynoms

$$p(\gamma) = \det(I - B_0 - \gamma B_1 + (r_0 + \gamma r_1)e_s^\top) \tag{2.44}$$

ist. Dabei sind

$$B_0 = (V_0 - G_0 V_0 D F_0^\top)V_1^{-1}, \qquad r_0 = \frac{1}{s!}\boldsymbol{c}^s - \frac{1}{(s-1)!}G_0\boldsymbol{c}^{s-1} - \frac{1}{s!}B_0(\boldsymbol{c} - \mathbb{1}),$$
$$B_1 = -V_0 D F_0^\top V_1^{-1}, \qquad r_1 = -\frac{1}{(s-1)!}\boldsymbol{c}^{s-1} - \frac{1}{s!}B_1(\boldsymbol{c} - \mathbb{1}).$$

2.4 FSAL-Methoden

Die Abkürzung FSAL kommt aus dem Englischen und steht für *first same as last*, was soviel bedeutet wie der Erste entspricht dem Letzten. Im Kontext der numerischen Lösung von Anfangswertproblemen meint man damit, dass der letzte Stufenwert im vorherigen Schritt mit dem ersten im aktuellen Schritt übereinstimmen soll. Dies gilt dann auch automatisch für die Funktionswerte dieser Stufenwerte. Man spart demzufolge in jedem Zeitschritt einen Funktionsaufruf.

Dieses Vorgehen wurde zuerst bei expliziten Runge-Kutta-Verfahren angewendet, wobei dort der Stufenwert der ersten Stufe im nächsten Schritt bereits im aktuellen verwendet wird. Dazu fügt man dem Verfahren eine zusätzliche Stufe hinzu, indem man den Gewichtsvektor b als letzte Zeile an die Verfahrensmatrix A anhängt. Dadurch entspricht der neue letzte Stufenwert der in diesem Zeitschritt berechneten Näherungslösung. Da die erste Stufe im nächsten Schritt auch der aktuellen Näherungslösung entspricht, stimmen diese beiden Stufen und somit auch die Funktionswerte überein und es entsteht kein zusätzlicher Aufwand, außer wenn ein Schritt verworfen wird.

Der Funktionswert dieses zusätzlichen Stufenwertes wird nun dazu verwendet, eine eingebettete Näherungslösung zur Schrittweitensteuerung zu berechnen. Ein sehr bekanntes Verfahren, welches diese Eigenschaft besitzt, ist das 7-stufige Verfahren DOPRI5(4) von Dormand und Prince [12] der Ordnung 5 mit einer eingebetteten Lösung der Ordnung 4.

Anders als bei expliziten Runge-Kutta-Verfahren wollen wir bei den Peer-Verfahren die letzte Stufe des vorherigen Zeitschrittes im aktuellen verwenden, um die Konsistenzordnung des Verfahrens um eins zu erhöhen. Man spart dabei das Lösen einer Stufengleichung ein und der Aufwand des Verfahrens mit s Stufen entspricht dem eines Verfahrens mit $s-1$ Stufen. Dazu fordern wir

$$Y_{m,1} = Y_{m-1,s}. \tag{2.45}$$

Dieses Vorgehen wurde bereits für explizite und einfach implizite Peer-Verfahren in [41] untersucht. Diese Bedingung hat einen weiteren Vorteil: Bei Peer-Verfahren haben alle Stufen die volle Ordnung $p = s-1$, dadurch kann man die Lösung an Zwischenpunkten mit Hilfe eines Interpolationspolynoms der Ordnung $s-1$ leicht berechnen (engl.: *dense output*). Das Problem ist nur, dass die Interpolationspolynome zweier aufeinander folgender Integrationsintervalle im Allgemeinen nicht zusammen passen, also nicht stetig verknüpft werden können. Wenn man jetzt aber (2.45) fordert, haben die beiden Interpolationspolynome an der Stelle t_m den gleichen Wert, man kann sie also stetig verbinden. Da sogar noch die Ableitung an dieser Stelle übereinstimmt, also $F_{m-1,s} = F_{m,1}$ gilt, ist das verknüpfte Interpolationspolynom an dieser Stelle sogar stetig differenzierbar, wenn man Hermit-Interpolation verwendet. Dies ist von Vorteil für viele Anwendungen, in denen eine global stetig oder sogar global stetig differenzierbare Lösung benötigt wird. Zum Beispiel, wenn man eine Kurve der Lösung $y(t)$ gegenüber der Zeit graphisch darstellen möchte oder bei retardierten Differentialgleichungen, bei denen die Lösung an zurückliegenden Zeitpunkten, an denen noch keine Lösung berechnet wurde, benötigt wird.

Ein Nachteil der Forderung (2.45) ist, dass einige Koeffizienten des Verfahrens fest vorgegeben sind, das Verfahren somit weniger Freiheitsgrade hat. Außerdem beeinflusst die Bedingung (2.45) die Stabilität, wie man später an Lemma 2.19 sehen wird.

Schauen wir uns zunächst an, welche Auswirkungen (2.45) auf die Koeffizienten des Verfahrens hat. Dazu betrachten wir die Verfahrensvorschrift (2.2)

$$Y_{m,i} = \sum_{j=1}^{s} b_{ij} Y_{m-1,j} + h_m \sum_{j=1}^{i} g_{ij} F_{m,j}, \quad i = 1, 2, \ldots, s.$$

Man erhält durch die Forderung (2.45)

$$\begin{aligned} & b_{1,s} = 1, \qquad b_{1,j} = 0, \qquad j = 1, 2, \ldots, s-1 \\ & g_{1,1} = 0 \\ \text{bzw.} \qquad & e_1^\top B_m = e_s^\top, \quad e_1^\top G_m = 0^\top. \end{aligned} \tag{2.46}$$

Weiterhin muss der Zeitpunkt der ersten Stufe im neuen Schritt mit dem der letzten Stufe im alten Schritt für beliebige Schrittweitenänderungen übereinstimmen

$$t_{m-1} + c_s h_{m-1} = t_m + c_1 h_m. \tag{2.47}$$

Daraus folgt, dass $c_s = 1$ und $c_1 = 0$ sein muss, da $t_m = t_{m-1} + h_{m-1}$ ist. Die Koeffizienten-Matrizen haben somit die folgende Form

$$B_m = \left(\begin{array}{c|ccc} 0 & 0 & \cdots & 0 \quad 1 \\ \hline b_1^m & & \widehat{B_m} & \end{array}\right) = \begin{pmatrix} 0 & e_{s-1}^\top \\ b_1^m & \widehat{B_m} \end{pmatrix}, \quad G_m = \left(\begin{array}{c|ccc} 0 & 0 & \cdots & 0 \\ \hline g_1^m & & \widehat{G_m} & \end{array}\right) = \begin{pmatrix} 0 & 0^\top \\ g_1^m & \widehat{G_m} \end{pmatrix}$$
$$c = (0, c_2, c_3, \ldots, c_{s-1}, 1). \tag{2.48}$$

Diese Wahl der Koeffizienten hat Auswirkung auf die Eigenschaften des Verfahrens. Schauen wir uns zuerst die Konsistenzbedingungen an. Dazu betrachten wir (2.16)

$$B_m = (V_0 - G_m V_0 D F_0^\top) S_m V_1^{-1}.$$

Wie wir in (2.46) gesehen haben, muss $e_1^\top B = e_s^\top$ sein. Dies prüft man leicht nach, denn

$$\begin{aligned} e_1^\top B_m &= e_1^\top (V_0 - G_m V_0 D F_0^\top) S_m V_1^{-1} = e_1^\top V_0 S_m V_1^{-1} - e_1^\top G_m V_0 D F_0^\top S_m V_1^{-1} \\ &= e_1^\top V_0 S_m V_1^{-1}, && \text{da} \quad e_1^\top G_m = 0^\top \\ &= e_1^\top S_m V_1^{-1} = e_1^\top P V_0^{-1} = \mathbb{1}^\top V_0^{-1} \\ &= e_s^\top, && \text{da} \quad e_s^\top V_0 = \mathbb{1}^\top. \end{aligned}$$

Die restlichen $s-1$ Zeilen von B_m ergeben sich aus der Gleichung (2.16). Damit hat das Verfahren die Konsistenzordnung $p = s - 1$. Als nächstes wollen wir die Koeffizienten der Matrix G_m bestimmen. Wir fordern auch hier, dass das Verfahren optimal nullstabil (siehe Definition 2.8) ist. Es zeigt sich, dass der Ansatz (2.6)

$$G_m = G_0 + \gamma(I - \mathbb{1} e_1^T)$$

dafür sehr gut geeignet ist. Dies setzen wir in Gleichung (2.37) ein und erhalten

$$\begin{aligned} \operatorname{tril}_2(I) &= \operatorname{tril}_2(P V_0^{-1} G_m V_0 D F_0^\top) \\ &= \operatorname{tril}_2(P V_0^{-1} \left(G_0 + \gamma \left(I - \mathbb{1} e_1^T\right)\right) V_0 D F_0^\top) \\ &= \operatorname{tril}_2(P V_0^{-1} G_0 V_0 D F_0^\top) + \gamma \operatorname{tril}_2(P D F_0^\top) - \gamma \operatorname{tril}_2(P V_0^{-1} \mathbb{1} e_1^\top V_0 D F_0^\top). \end{aligned}$$

Es gilt wie vorher $\operatorname{tril}_2(P D F_0^\top) = 0$, da $P D F_0^\top$ eine strenge obere Dreiecksmatrix ist. Für den dritten Term erhalten wir

$$\operatorname{tril}_2(P V_0^{-1} \mathbb{1} e_1^\top V_0 D F_0^\top) = \operatorname{tril}_2(e_1 e_2^\top) = 0,$$

da $P V_0^{-1} \mathbb{1} = e_1$, $e_1^\top V_0 D F_0^\top = e_2^\top$ und $e_1 e_2^\top$ eine strenge obere Dreiecksmatrix ist. Damit sind auch die FSAL-Peer-Methoden genau dann optimal nullstabil, wenn G_0 der Bedingung (2.38)

$$\operatorname{tril}_2(I) = \operatorname{tril}_2(P V_0^{-1} G_0 V_0 D F_0^\top)$$

genügt.

Auch für FSAL-Peer-Methoden kann man mit Hilfe der Superkonvergenz die Konvergenzordnung für konstante Schrittweiten um eins auf $p = s$ erhöhen. Analog zu Satz 2.17 erhalten wir den folgenden

Satz 2.18 [Superkonvergenz, FSAL]
Es sei ein optimal nullstabiles einfach implizites Peer-Verfahren der Ordnung $p = s-1$ der Form (2.2) gegeben, dessen Matrix G_m die Gestalt (2.6) besitzt. Die Knoten c_i seien paarweise verschieden und die Matrix B_m ergebe sich aus (2.16). Außerdem gelte $c_0 = 0$ und $c_s = 1$. Dann ist das Verfahren genau dann konvergent von der Ordnung $p = s$ für konstante Schrittweiten, wenn γ eine Nullstelle des Polynoms

$$p(\gamma) = \det(I - B_0 - \gamma B_1 + (r_0 + \gamma r_1)e_s^\top) \tag{2.49}$$

ist. Dabei sind

$$B_0 = (V_0 - G_0 V_0 D F_0^\top)V_1^{-1}, \qquad r_0 = \frac{1}{s!}\boldsymbol{c}^s - \frac{1}{(s-1)!}G_0\boldsymbol{c}^{s-1} - \frac{1}{s!}B_0(\boldsymbol{c} - \mathbb{1}),$$
$$B_1 = (\mathbb{1}e_2^\top - V_0 D F_0^\top)V_1^{-1}, \qquad r_1 = -\frac{1}{(s-1)!}\boldsymbol{c}^{s-1} - \frac{1}{s!}B_1(\boldsymbol{c} - \mathbb{1}).$$

Da bei FSAL-Methoden $g_{11} = 0$ gilt, folgt aus $A(\alpha)$-Stabilität nicht zwangsläufig $L(\alpha)$-Stabilität. Man erhält stattdessen das folgende

Lemma 2.19 [L-Stabilität]
Besitzen die Koeffizienten-Matrizen des Peer-Verfahrens die Form (2.48) und ist die Matrix $\widehat{G}$ regulär, so ist die Methode L-stabil bzw. $L(\alpha)$-stabil, wenn gilt:

$$e_s^\top \left(e_1 e_1^\top + G_m\right)^{-1} e_1 = 0. \tag{2.50}$$

Beweis: Um zu überprüfen, ob das Verfahren L-stabil ist, müssen wir den Spektralradius von $M(\infty)$ berechnen. Es gilt:

$$M(z) = (I - zG)^{-1}B, \qquad G = \begin{pmatrix} 0 & 0^\top \\ g_1 & \widehat{G} \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & e_{s-1}^\top \\ b_1 & \widehat{B} \end{pmatrix}$$
$$(I - zG)^{-1} = \begin{pmatrix} 1 & 0^\top \\ -zg_1 & I - z\widehat{G} \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0^\top \\ -z(I - z\widehat{G})^{-1}g_1 & (I - z\widehat{G})^{-1} \end{pmatrix}$$
$$\Rightarrow \qquad (I - zG)^{-1} \to \begin{pmatrix} 1 & 0^\top \\ -\widehat{G}^{-1}g_1 & 0 \end{pmatrix}, \quad \text{für} \quad (z \to \infty)$$
$$M(\infty) = \begin{pmatrix} 1 & 0^\top \\ -\widehat{G}^{-1}g_1 & 0 \end{pmatrix}\begin{pmatrix} 0 & e_{s-1}^\top \\ b_1 & \widehat{B} \end{pmatrix} = \begin{pmatrix} 0^\top & 1 \\ 0 & -\widehat{G}^{-1}g_1 \end{pmatrix}$$
$$\Rightarrow \qquad \varrho(M(\infty)) = \left| e_{s-1}^\top \widehat{G}^{-1} g_1 \right|. \tag{2.51}$$

Wenn man jetzt die Inverse

$$\left(e_1 e_1^\top + G_m\right)^{-1} = \begin{pmatrix} 1 & 0^\top \\ g_1 & \widehat{G} \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0^\top \\ -\widehat{G}^{-1}g_1 & \widehat{G}^{-1} \end{pmatrix}$$

bildet, sieht man, dass der letzte Eintrag der ersten Spalte dem Spektralradius (2.51) entspricht

$$\varrho(M(\infty)) = \left| e_{s-1}^\top \widehat{G}^{-1} g_1 \right| = \left| e_s^\top \left(e_1 e_1^\top + G_m \right)^{-1} e_1 \right|. \tag{2.52}$$

Damit ist das Verfahren L-stabil bzw. L(α)-stabil, wenn dieses Element Null ist. □

Konstruktion von FSAL-Methoden für $s = 3$ und 4

Die Konstruktion von superkonvergenten optimal nullstabilen FSAL-Methoden mit $\varrho(M(\infty)) = 0$ geschieht nach folgenden Schema:

1. Bestimme G_0 nach Gleichung (2.38), es bleiben die freien Parameter $c_2, c_3, \ldots, c_{s-1}$ und γ.
2. Berechne B_m für $\sigma = 1$ nach Gleichung (2.16), da die Superkonvergenz nur für konstante Schrittweiten gilt. Das Verfahren besitzt unabhängig davon die Konsistenzordnung $p = s - 1$ für variable Schrittweiten, wenn B_m für beliebiges σ nach (2.16) berechnet wird.
3. Löse (2.49) und (2.50) nach den freien Parametern auf.

Methoden mit $s = 3$

Wenn man G_0 nach Gleichung (2.38) bestimmt und B_m für $\sigma = 1$ nach (2.16) berechnet, bleiben c_2 und γ als freie Parameter. Nun setzen wir G_0 und B_m in (2.49) und (2.50) ein und erhalten die zwei Gleichungen

$$\begin{aligned} c_2 &= -\frac{2}{3} + 6\gamma - 4\gamma^2, \\ c_2 &= 1 - 2\gamma + 2\gamma^2. \end{aligned}$$

Daraus ergeben sich die beiden Lösungen:

$$\begin{aligned} \gamma &= \frac{1}{6}(4 - \sqrt{6}) \approx 0.258, \quad c_2 = \frac{1}{9}(8 - \sqrt{6}) \approx 0.617, \\ \gamma &= \frac{1}{6}(4 + \sqrt{6}) \approx 1.075, \quad c_2 = \frac{1}{9}(8 + \sqrt{6}) \approx 1.161. \end{aligned}$$

In Tabelle 2.1 geben wir die Koeffizienten G und c_i dieser beiden Verfahren sowie die Eigenschaften der Methoden an.

Methoden mit $s = 4$

G_0 wird wieder nach Gleichung (2.38) bestimmt und B_m für $\sigma = 1$ nach (2.16). Die beiden Gleichungen (2.49) und (2.50) sind diesmal um einiges komplizierter, können aber nach c_2 und c_3 in Abhängigkeit von γ aufgelöst werden. Für jedes γ existieren zwei Methoden. Wir haben uns dabei für die Werte entschieden, bei denen c_2 und

	FSAL3a	FSAL3b
c_1	0.0000000000000000	0.0000000000000000
c_2	0.6167233619129802	1.1610544158647975
c_3	1.0000000000000000	1.0000000000000000
g_{21}	0.3583049857101766	0.0861394587342676
g_{31}	0.4308453506502256	−0.0055579943283866
g_{32}	0.3107362731469707	−0.0693569628021431
γ	0.2584183762028037	1.0749149571305296
α	81.7°	90.0°
err	0.0044	0.7767
$\|B\|_\infty$	1.85	15.26

Tabelle 2.1: Koeffizienten und Eigenschaften der beiden dreistufigen superkonvergenten und $L(\alpha)$-stabilen Methoden FSAL3a und FSAL3b. Der Fehler err wird dabei nach Formel (3.16) berechnet. Die Norm $\|B\|_\infty$ haben wir für $\sigma = 1$ bestimmt.

c_3 im Intervall $[-1, 2]$ liegen. Wir wählen den Parameter γ so, dass der Winkel der $L(\alpha)$-Stabilität möglichst groß und der Fehler err (3.16) möglichst klein wird. Wir haben $c_2, c_3, \alpha, \text{err}$ und $\|B\|_\infty$ in Abhängigkeit von γ in Abbildung 2.1 dargestellt. Da der Fehler err in der Nähe von $\gamma = 0.16$ ein markantes Minimum aufweist, haben wir uns dafür entschieden, das γ zu wählen, für das der Fehler minimal wird, da auch die anderen Eigenschaften für dieses γ akzeptable Werte annehmen. Wir haben das entsprechende γ in Abbildung 2.1 mit einer gestrichelten Linie markiert. Die Koeffizienten G und c_i sowie die Merkmale dieser Methode haben wir in Tabelle 2.2 angegeben.

FSAL4			
c_1	0.0000000000000000	c_3	0.7385867307067128
c_2	0.4120290561242265	c_4	1.0000000000000000
g_{21}	0.2503551264476477	g_{32}	0.2926870831948634
g_{31}	0.2842257178352702	g_{42}	0.3327263257572196
g_{41}	0.2931770845512618	g_{43}	0.2124226600149393
γ	0.1616739296765787	α	74.6°
err	0.00017	$\|B\|_\infty$	5.88

Tabelle 2.2: Koeffizienten und Eigenschaften der vierstufigen, superkonvergenten und $L(\alpha)$-stabilen FSAL-Peer-Methode FSAL4. Der Fehler err wird dabei nach Formel (3.16) berechnet. Die Norm $\|B\|_\infty$ haben wir für $\sigma = 1$ bestimmt.

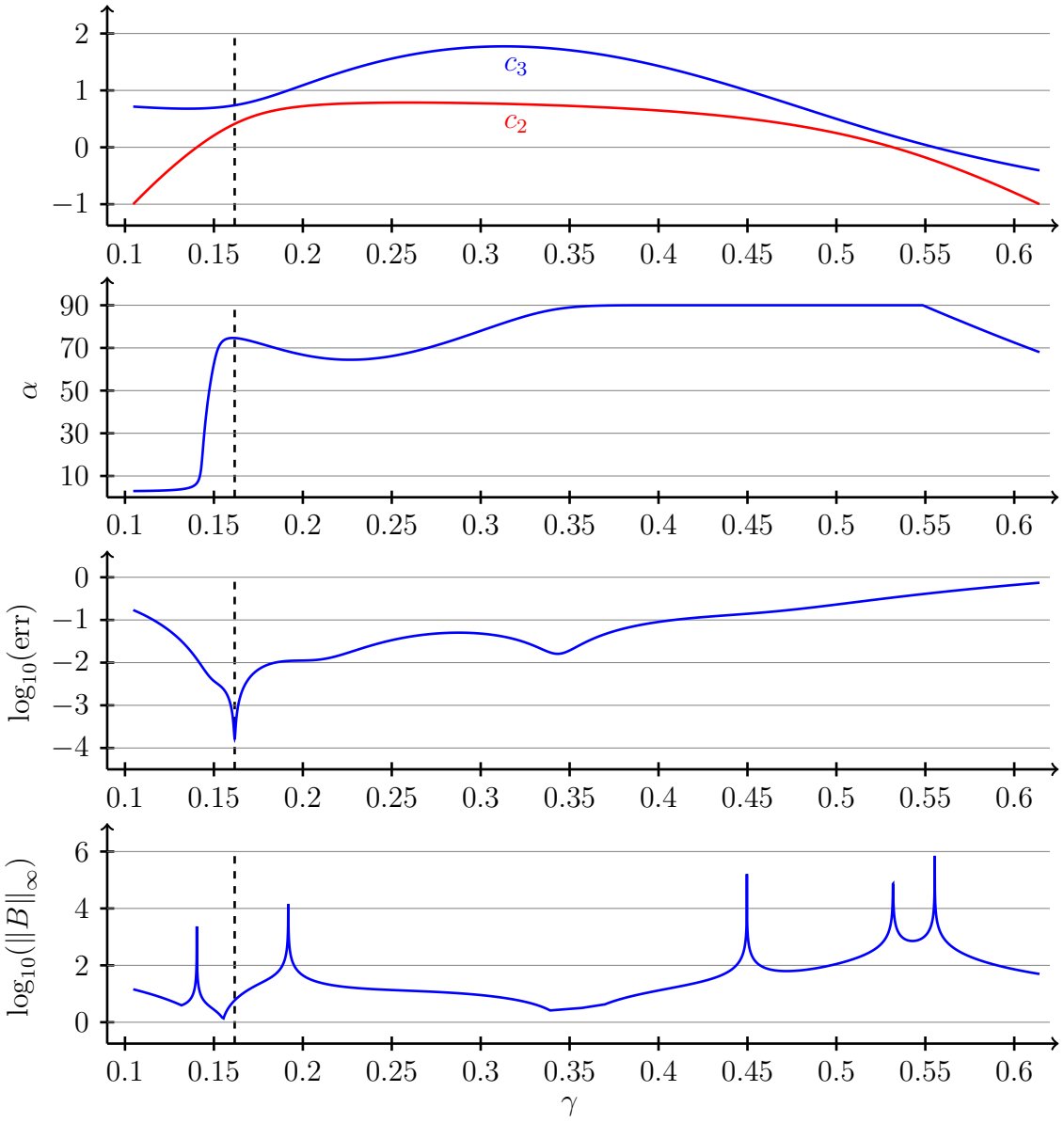


Abbildung 2.1: Eigenschaften vierstufiger, superkonvergenter und L(α)-stabiler FSAL-Peer-Methoden. Der Fehler err wird dabei nach Formel (3.16) berechnet. Die gestrichelte Linie markiert die Methode FSAL4

3 Krylov-Peer-Verfahren

Wir werden uns in diesem Kapitel damit beschäftigen, Peer-Verfahren zu konstruieren, die steife Anfangswertprobleme sehr großer Dimension ($n \gtrsim 10000$) möglichst effizient numerisch lösen. Die Schwierigkeiten, die dabei auftreten, sind die Berechnung der Jacobi-Matrix und das Lösen der linearen Gleichungssysteme (2.9). Eine Möglichkeit, den Aufwand dafür zu reduzieren, ist das Verwenden von Krylov-Unterraum-Verfahren, welche eine Approximation der Lösung des linearen Gleichungssystems iterativ berechnen. Dafür benötigen sie die Jacobi-Matrix $J = f_y(t_m, u_m)$ nicht explizit, sondern nur in Verbindung mit Matrix-Vektor-Produkten Jv. Die exakte Lösung von (2.9) wird mit Hilfe eines Krylov-Unterraumes approximiert, wofür nur Gleichungssysteme geringer Dimension gelöst werden müssen.

Dieses Vorgehen wird bereits bei einigen anderen Verfahren erfolgreich angewendet. Ein sehr bekanntes Beispiel hierfür ist der BDF-Mehrschrittcode VODPK von Brown, Byrne und Hindmarsh [6, 10], welcher zur iterativen Lösung der linearen Gleichungssysteme das GMRES-Verfahren (Generalized Minimal Residual) von Saad und Schultz [38] verwendet. Ein weiteres Beispiel ist der Solver ROWMAP von Weiner, Schmitt und Podhaisky [51], welcher auf Rosenbrock-Methoden basiert und zur Lösung der linearen Gleichungssysteme einen mehrfachen Arnoldi-Prozess (MAP) verwendet. Eine Matlab Version von ROWMAP wurde von Beck [2] implementiert.

Weiterhin gibt es noch die Exponentielle W-Methode EXP4 von Hochbruck, Lubich und Selhofer [22], welche Krylov-Techniken zur Approximation von

$$\varphi_1(h\gamma J)v, \quad \varphi_1(z) = (e^z - 1)/z, \quad J = f_y(t_m, u_m), \tag{3.1}$$

verwendet.

Wir werden in diesem Kapitel zunächst auf die theoretischen Grundlagen der Krylov-Unterraum-Verfahren eingehen und eine Möglichkeit zur Anwendung auf lineare Gleichungssysteme vorstellen. Im zweiten Abschnitt werden wir spezielle Peer-Verfahren konstruieren, welche wir mit den Krylov-Techniken kombinieren. Im letzten Abschnitt gehen wir auf Fragen der Implementierung ein.

3.1 Theoretische Grundlagen

Die im Folgenden aufgeführten Aussagen findet man z.B. in den Büchern von Saad [37] und Meister [30], deshalb werden wir die Beweise hier weglassen. Zunächst einmal führen wir den Krylov-Unterraum ein, mit dessen Hilfe wir später die Nährerungslösung von (2.9) approximieren.

Definition 3.1 [Krylov-Unterraum]
Der Raum

$$\mathcal{K}(A, \omega, \varkappa) = \operatorname{span}\left\{\omega, A\omega, \ldots, A^{\varkappa-1}\omega\right\} \tag{3.2}$$

wird als Krylov-Unterraum zur Matrix $A \in \mathbb{R}^{n,n}$ und zum Startvektor $\omega \in \mathbb{R}^n$ bezeichnet. Es besitzt die Dimension $\varkappa$. □

Da sich die numerische Berechnung der Krylov-Vektoren $A^k\omega$ als numerisch ungünstig erweist, berechnet man mit dem Arnoldi-Verfahren eine orthonormale Basis

$$\{q_1, q_2, \ldots, q_\varkappa\} \tag{3.3}$$

zum Krylov-Unterraum $\mathcal{K}(A, \omega, \varkappa)$. Dies wird durch die Verbindung von Krylov-Schritten (Av) und dem modifizierten Gram-Schmidt-Verfahren zur Orthogonalisierung realisiert. Die erzeugten Basisvektoren aus (3.3) fügen wir zu Matrizen $Q_j = (q_1, q_2, \ldots, q_j) \in \mathbb{R}^{n,j}$ zusammen und erhalten das Arnoldi-Verfahren in folgender Form:

$$q_1 = \frac{\omega}{\|\omega\|_2}\,, \quad q_{j+1} = \frac{v}{\|v\|_2}\,, \quad v = Aq_j - \sum_{i=1}^{j} q_i h_{ij}\,, \quad h_{ij} = q_i^T A q_j \tag{3.4}$$

Die Elemente h_{ij} bilden die Matrix

$$H_j = Q_j^T A Q_j \in \mathbb{R}^{j,j}\,. \tag{3.5}$$

Wir wollen jetzt die für weitere Untersuchungen benötigten Eigenschaften des Arnoldi-Verfahrens angeben (vgl. [49]). Wir setzen dabei vorerst stets voraus, dass die in (3.4) auftretenden Vektoren v nicht der Nullvektor sind (siehe „lucky breakdown“ auf Seite 33).

Lemma 3.2
Die Matrizen Q_j sind orthogonal, d.h., $Q_j^T Q_j = I \in \mathbb{R}^{j,j}$ □

Lemma 3.3
Die durch (3.5) *definierte Matrix*

$$H_\varkappa = Q_\varkappa^T A Q_\varkappa \in \mathbb{R}^{\varkappa,\varkappa}$$

besitzt obere Hessenbergform, d.h. $h_{ij} = 0$ für $i > j+1$. Speziell gilt

$$h_{j+1,j} = \left\|(I - Q_j Q_j^T) A q_j\right\|_2\,. \quad \square$$

Bemerkung 3.4 [Nachorthogonalisierung]
Obwohl das Arnoldi-Verfahren schon modifiziert wurde, um den Rundungsfehlereinfluss zu reduzieren, können dennoch Probleme bei der Berechnung des Vektors q_{j+1} auftreten. Dieser soll der bereits berechneten Basis $q_1, q_2, \ldots, q_j$ hinzugefügt werden, so dass der Vektor Aq_j mit Hilfe der neuen Basis $q_1, q_2, \ldots, q_j, q_{j+1}$ dargestellt werden kann.

Wenn jetzt der Vektor Aq_j schon fast linear abhängig von der Basis $q_1, q_2, \ldots, q_j$ ist, treten immer noch erhebliche Rundungsfehler auf und der neu berechnete Basisvektor q_{j+1} ist ungenau [15]. Dies zeigt sich dahingehend, dass die Skalarprodukte $q_{j+1}^\top q_i$,

für $i = 1, 2, \ldots, j$ mit $i \neq j$ nicht mehr exakt Null sind und somit ein sogenannter Orthogonalitätsverlust in der Matrix Q_{j+1} auftritt.

Diesen Sachverhalt haben wir in [2] untersucht und eine Möglichkeit gefunden, die auftretenden Rundungsfehler zu minimieren. Dazu führt man eine sogenannte Nachorthogonalisierung durch. Darunter versteht man die wiederholte Orthogonalisierung des beim Arnoldi-Prozess berechneten Vektors v, bevor aus diesem der neue Basisvektor $q_{j+1} = v/\|v\|_2$ erzeugt wird.

Hierbei ist sehr interessant, dass bereits eine Nachorthogonalisierung ausreicht, damit die Rundungsfehler im Bereich der Maschinengenauigkeit liegen. Dies wurde zuerst von Parlett und Kahan für zwei Vektoren gezeigt [32] und später für das modifizierte Arnoldi-Verfahren von Giraud, Langou und Rozložník [17] verallgemeinert. Das bedeutet natürlich, dass sich der Aufwand verdoppelt. Deshalb möchte man eine Nachorthogonalisierung nur dann durchführen, wenn sie auch wirklich nötig ist. Dafür wurde in [2] das L-Kriterium, von Giraud, Langou und Rozložník [18], verwendet. Beim Arnoldi-Verfahren (3.4) wird eine Nachorthogonalisierung durchgeführt, wenn

$$\frac{\sum_{i=1}^{j} |h_{ij}|}{h_{j+1,j}} \geq L \tag{3.6}$$

gilt. Dabei ist $L \in \mathbb{R}$ mit $L \geq 0$. Wir erhalten damit insgesamt den Algorithmus 1. Eine

Algorithm 1 mod.-Arnoldi-Verfahren mit L-Kriterium und Nachorthogonalisierung

for $i = 1$ to $\varkappa$ **do**
  $v = Aq_j$
  for $i = 1$ to j **do**
    $h_{ij} = q_i^T v$
    $v = v - h_{ij} q_i$
  end for
  $h_{i+1,j} = \|v\|_2$
  if $h_{i+1,j} = 0$ **then**
    Stop.
  end if
  if $\sum_{i=1}^{j} |h_{ij}| / h_{j+1,j} \geq L$ **then** {L-Kriterium}
    for $i = 1$ to j **do** {Nachorthogonalisierung}
      $v = v - \left(q_i^T v\right) q_i$
    end for
  end if
  $q_{j+1} = v/\|v\|_2$
end for

Nachorthogonalisierung wird also immer nur dann ausgeführt, wenn das L-Kriterium erfüllt ist. Wie in [2] wählen wir auch hier $L = 10000$.

Lemma 3.5
Die Vektoren q_j lassen sich darstellen durch

$$q_j = p_{j-1}(A) q_1 \ ,$$

wobei $p_{j-1}(A)$ ein Polynom vom Grad $j-1$ in A ist. □

Die $\varkappa$ linear unabhängigen Vektoren q_i lassen sich also als Linearkombination der $\varkappa$ Vektoren $q_1, Aq_1, \ldots, A^{\varkappa-1}q_1$ darstellen. Da diese folglich auch linear unabhängig sein müssen erhalten wir:

Folgerung 3.6
Die Vektoren $q_1, \ldots, q_\varkappa$ bilden eine orthonormale Basis für den Krylovraum $\mathcal{K}(A, q_1, \varkappa)$, d.h.

$$\operatorname{span}\left\{q_1, Aq_1, \ldots, A^{\varkappa-1}q_1\right\} = \operatorname{span}\left\{q_1, q_2, \ldots, q_\varkappa\right\} \ . \quad \square$$

Der folgende Satz bildet die Grundlage für die effiziente Berechnung des Residuums bei der Lösung linearer Gleichungssysteme mit dem Arnoldi-Verfahren.

Satz 3.7
Es gilt die Beziehung

$$AQ_\varkappa = Q_\varkappa H_\varkappa + h_{\varkappa+1,\varkappa} q_{\varkappa+1} e_\varkappa^T \ , \tag{3.7}$$

mit $e_\varkappa^T = (0, \ldots, 0, 1) \in \mathbb{R}^\varkappa$. □

Anwendung auf lineare Gleichungssysteme

Die Gleichungssysteme aus (2.9) haben die Form

$$(I - \delta A)\, x = z \qquad I, A \in \mathbb{R}^{n,n}, \quad x, z \in \mathbb{R}^n. \tag{3.8}$$

Dabei ist $\delta = h\gamma$ und A eine Approximation an die Jacobi-Matrix, z.B. $A = f_y\left(t_{m-1,s}, Y_{m-1,s}\right)$. Wir suchen eine Näherungslösung k von (3.8) im Krylov-Unterraum

$$\mathcal{K}_\varkappa = \mathcal{K}(A, q_1, \varkappa) = \operatorname{span}\left\{q_1, Aq_1, \ldots, A^{\varkappa-1}q_1\right\} = \operatorname{span}\left\{q_1, q_2, \ldots, q_\varkappa\right\} \ .$$

Wir betrachten im weiteren die Methode der vollständigen Orthogonalisierung (FOM - *full orthogonalization method*). Hier bestimmt man die Näherungslösung unter der zusätzlichen Bedingung

$$Q_\varkappa^T \left((I - \delta A)\, k - z\right) = 0 \ . \tag{3.9}$$

Diese *Petrov-Galerkin-Bedingung* besagt, dass das Residuum senkrecht zum Krylov-Unterraum sein soll. Weiterhin wird gefordert, dass die Näherungslösung in $\mathcal{K}_\varkappa$ liegt, d.h., sie lässt sich darstellen in der Form

$$k = Q_\varkappa l \ , \qquad l \in \mathbb{R}^\varkappa \ .$$

Einsetzen in (3.9) ergibt mit $Q_\varkappa^T Q_\varkappa = I_\varkappa$ und $H_\varkappa = Q_\varkappa^T A Q_\varkappa$

$$Q_\varkappa^T \left((I - \delta A)\, Q_\varkappa l - z\right) = 0$$

und

$$(I - \delta H_\varkappa)\, l = Q_\varkappa^T z$$

Wir erhalten

$$l = (I - \delta H_\varkappa)^{-1} Q_\varkappa^T z \tag{3.10}$$

und damit

$$k = Q_\varkappa (I - \delta H_\varkappa)^{-1} Q_\varkappa^T z \; . \tag{3.11}$$

Das $(\varkappa, \varkappa)$- lineare Gleichungssystem

$$(I - \delta H_\varkappa)\, l = Q_\varkappa^T z \tag{3.12}$$

wird durch LU- oder QR-Zerlegung gelöst. Für eine QR-Zerlegung wendet man Givens-Drehungen an, da die Matrix H Hessenberg-Struktur hat.

Beachte: $(I - \delta H)$ ist für kleine h immer regulär, für Matrizen A mit logarithmischer Matrixnorm $\mu \leq 0$ ist es für alle h regulär.

Wegen $k = Q_\varkappa l$ liegt die Lösung im Krylov-Unterraum und es gilt

$$Q_\varkappa Q_\varkappa^T k = Q_\varkappa \underbrace{Q_\varkappa^T Q_\varkappa}_{I_\varkappa} l = Q_\varkappa l = k \; .$$

Die Lösung k ist natürlich im Allgemeinen nicht die exakte Lösung von (3.8). Es ist daher wichtig zu wissen, wie genau diese Lösung ist. Die Genauigkeit wird charakterisiert durch das Residuum

$$r = (I - \delta A)\, k - z \; .$$

Mit (3.9) ist dies äquivalent zu

$$\begin{aligned} r &= \left(I - Q_\varkappa Q_\varkappa^T\right) \left[(I - \delta A)\, k - z\right] \\ &= -\delta \left(I - Q_\varkappa Q_\varkappa^T\right) Ak - \left(I - Q_\varkappa Q_\varkappa^T\right) z \\ &= -\delta \left(I - Q_\varkappa Q_\varkappa^T\right) AQ_\varkappa l - \left(I - Q_\varkappa Q_\varkappa^T\right) z \; . \end{aligned}$$

Mit

$$AQ_\varkappa = Q_\varkappa H_\varkappa + h_{\varkappa+1,\varkappa} q_{\varkappa+1} e_\varkappa^T$$

folgt

$$r = -\delta \left(I - Q_\varkappa Q_\varkappa^T\right) \left[Q_\varkappa H_\varkappa + h_{\varkappa+1,\varkappa} q_{\varkappa+1} e_\varkappa^T\right] l - \left(I - Q_\varkappa Q_\varkappa^T\right) z \; .$$

Mit $\left(I - Q_\varkappa Q_\varkappa^T\right) Q_\varkappa = Q_\varkappa - Q_\varkappa = 0$ und $q_{\varkappa+1}^T q_i = 0$ für $i \leq \varkappa$ ergibt sich

$$r = -\delta h_{\varkappa+1,\varkappa} q_{\varkappa+1} e_\varkappa^T l - \left(I - Q_\varkappa Q_\varkappa^T\right) z \; .$$

Dabei ist zu beachten, dass $h_{\varkappa+1,\varkappa}$ und der Vektor $q_{\varkappa+1}$ sowieso für den nächsten Arnoldi-Schritt benötigt werden. Weiterhin ist $e_\varkappa^T l$ die letzte Komponente des Lösungsvektors $l \in \mathbb{R}^\varkappa$, die ohne wesentlichen Aufwand berechnet werden kann, wenn die Faktorisierung von $I - \delta H$ aufdatiert wird. Man muss also nicht die gesamte Lösung des LGS kennen um das Residuum zu berechnen.

Der Term $(I - Q_\varkappa Q_\varkappa^T)z$ in der Darstellung des Residuums ist für theoretische Untersuchungen etwas störend. I. Allg. wird daher der Krylov-Raum mit dem Startwert

$$q_1 = \frac{z}{\|z\|_2} \tag{3.13}$$

konstruiert, da dann offensichtlich

$$(I - Q_\varkappa Q_\varkappa^T)z = 0 \tag{3.14}$$

gilt. Das Residuum vereinfacht sich damit zu

$$r = -\delta h_{\varkappa+1,\varkappa} q_{\varkappa+1} e_\varkappa^T l \; . \tag{3.15}$$

Die Wahl des Startvektors nach (3.13) ist noch aus einem anderen Grund vorteilhaft. An (3.4) kann man erkennen, dass der Arnoldi-Prozess zusammenbricht, wenn in einem Schritt j der Vektor v der Nullvektor ist. Das bedeutet aber

$$h_{j+1,j} = \left\|(I - Q_j Q_j^T)Aq_j\right\|_2 = 0 \; .$$

Das Residuum (3.15) wird dann Null, wir haben die exakte Lösung gefunden. Man spricht daher von einem „lucky breakdown“.

3.2 Konstruktion spezieller Methoden

Wir wollen in diesem Abschnitt die Koeffizienten konkreter Verfahren angeben. Wir beschränken uns dabei auf einfach implizite Methoden, d.h. $g_{ii} = \gamma$, und damit $G = G_0 + \gamma I_s$. Bei direkter Lösung der linearen Gleichungssysteme sind einfach implizite Verfahren vorteilhaft, da sie nur eine LU-Zerlegung benötigen. Für Krylov-Methoden sind mehrfach implizite Verfahren vom Aufwand her vergleichbar. Testrechnungen in [4] zeigen, dass aber auch hier einfach implizite Verfahren genauso gute Ergebnisse liefern. Wir konzentrieren uns deshalb im Weiteren auf diese Verfahrensklasse.

Wir konstruieren jetzt einfach implizite Verfahren, welche die folgenden Eigenschaften besitzen sollen:

- Konvergenzordnung $p = s - 1$ für variable Schrittweiten
- Optimale Nullstabilität (2.21)
- Superkonvergenz $p = s$ für konstante Schrittweiten
- $L(\alpha)$-Stabilität mit möglichst großem α
- kleine Fehlerkonstante für $\sigma = 1$

$$\text{err} = \left\| \frac{1}{(p+1)!} c^{p+1} - \frac{1}{(p+1)!} B\,(c-1)^{p+1} - \frac{1}{p!} G c^p \right\|_2 . \tag{3.16}$$

Der zu normierende Vektor entspricht dabei dem Fehlerterm von $\mathcal{O}(h^{p+1})$ des Residuums in (2.11).

- kleine Werte für γ

Die Knoten $c_i, i = 1, 2, \ldots, s$ sind freie Parameter. Für gegebene c_i gilt:

1. aus der optimalen Nullstabilität (2.21) folgen die $g_{ij}, j < i$, also G_0
2. aus der Superkonvergenz erhält man mit der Matrix G_0 maximal s mögliche Werte für γ und damit G
3. (2.16) liefert dann für beliebiges σ_m die Koeffizientenmatrix B_m, das Verfahren hat somit die Konvergenzordnung $p = s - 1$

Für gegebene Knoten c_i bekommt man also maximal s verschiedene superkonvergente Verfahren. Außerdem sind durch diese Konstruktion die ersten drei geforderten Eigenschaften automatisch erfüllt. Wir müssen jetzt solche c_i und dazugehörige Werte für γ (resultierend aus der Superkonvergenz) finden, die für die letzten drei Eigenschaften möglichst gute Ergebnisse liefern. Dazu verwenden wir einen Evolutionsalgorithmus [5] zur Suche, der sich wie folgt beschreiben lässt:

1. Erzeuge eine zufällig gleichverteilte Startpopulation P von n_p Eltern von Knotenvektoren $c \in \mathbb{R}^s$ mit $c_s = 1$ und $c_i \in [a, b]$, z.B. in Matlab mit Hilfe der gleichverteilten Zufallsfunktion rand.

$$P(1 : s-1, 1 : n_p) = a + (b - a)\,\mathrm{rand}(s - 1, n_p)$$
$$P(s, 1 : n_p) = 1$$

2. Berechne die Eigenschaften α, err und γ von jedem Knotenvektor in P und bewerte diese mit Hilfe einer Optimierungsfunktion $f(\alpha, \mathrm{err}, \gamma) : \mathbb{R}^3 \to \mathbb{R}$.

3. Die Hauptschleife:
   wähle eine Startintervalllänge h_c

   a) Berechne zu jedem Elter $P(:, j), j = 1, 2, \ldots, n_p$, n_c Kinder $C(:, k), k = (j - 1)n_c + 1, \ldots, jn_c$ durch gleichverteilte zufällige Wahl der Knoten $C(i, k)$ in den Intervallen $[P(i, j) - \frac{h_c}{2}, P(i, j) + \frac{h_c}{2}]$ für $i = 1, 2, \ldots, s-1$ und $C(s, k) = 1$. Dies kann man in Matlab wieder wie folgt realisieren:

$$C(1 : s-1, (j-1)n_c + 1 : jn_c) = P(1 : s-1, j) - \frac{h_c}{2} + h_c\,\mathrm{rand}(s - 1, n_c)$$
$$C(s, (j-1)n_c + 1 : jn_c) = 1$$

   b) Berechne die Eigenschaften α, err und γ aller Kinder in C.

   c) Bewerte die berechneten Eigenschaften aus 3b mit Hilfe der Optimierungsfunktion $f(\alpha, \mathrm{err}, \gamma)$, d.h., jedes Kind bekommt eine Punktzahl zugeordnet.

   d) Aus allen $n = n_p(n_c + 1)$ Individuen in P und C werden n_p Individuen mit der höchsten Punktzahl ausgewählt und diese bilden die nächste Elterngeneration. Die aktuelle Intervalllänge h_c wird mit einem festen Faktor $0 < \lambda < 1$ verkleinert, $h_c = \lambda h_c$. Gehe zurück zu 3a und wiederhole die Schritte 3a bis 3d solange, bis h_c kleiner als eine vorgegebene minimale Intervalllänge h_{min} ist.

4. Die letzte Elterngeneration ist das Ergebnis des Algorithmus.

Wir haben mit Hilfe des angegebenen Algorithmus für $s = 3, 4, 5$ Verfahren gesucht und anschließend getestet. Dabei haben sich die in Tabelle 3.1 angegebenen Methoden, welche bereits in [4] untersucht und getestet wurden, als gut geeignet erwiesen.

In Abbildung 3.1 stellen wir die Stabilitätsgebiete dieser drei Peer-Verfahren dar. Den Rand des Stabilitätsgebietes erhält man, indem man die Eigenwertgleichung der Stabilitätsmatrix $M(z)$ gegeben durch (2.31) in ein Eigenwertproblem bezüglich z überführt und für die Eigenwerte von $M(z)$ mit Betrag eins die zugehörigen Werte für z bestimmt. Die Eigenwertgleichung von $M(z)$ ist gegeben durch

$$(I_s - zG)^{-1}Bx = \lambda x.$$

Diese Gleichung überführen wir in ein Eigenwertproblem bezüglich z. Wir erhalten

$$(\lambda G)^{-1}(\lambda I_s - B)x = zx. \tag{3.17}$$

Nun setzt man für λ in (3.17) Werte auf dem Einheitskreis ein, d.h., man wählt $\lambda = \mathrm{e}^{i\varphi}$ mit $\varphi \in [0, 2\pi)$ und berechnet für diese Werte die Eigenwerte z der Matrix $(\lambda G)^{-1}(\lambda I_s - B)$. Die berechneten Eigenwerte bilden die sogenannte Wurzelortskurve, welche den Rand des Stabilitätsgebietes enthält.

Bemerkung 3.8
In Tabelle 3.1 haben wir Peer-Methoden angegeben die nicht A-stabil sind. Es existieren aber A-stabile Peer-Verfahren mit $s = 3, 4, 5$, welche jedoch größere Fehlerkonstanten und größere Werte für γ haben.

3.3 Fragen der Implementierung

Bestimmung der Startwerte mit ROWMAP

Das Peer-Verfahren (2.2) ist eine Zweischritt-Methode und benötigt zur Berechnung der neuen Stufenwerte Y_m die alten Stufenwerte Y_{m-1} aus dem vorherigen Zeitschritt. Das bedeutet für den Startschritt, es werden s Startwerte $Y_{0,1}, Y_{0,2}, \ldots, Y_{0,s}$ benötigt. Durch das Anfangswertproblem (2.1) ist aber nur ein Startwert $y(t_0) = y_0$ gegeben. Die restlichen $s - 1$ Startwerte berechnen wir mit ROWMAP. Dazu müssen wir zuerst die Zeitpunkte $t_{0,i}$ bestimmen, zu denen die Startwerte $Y_{0,i}$ gehören. Die Peer-Methode führt immer einen Zeitschritt mit der Schrittweite h_{m-1} von t_{m-1} nach t_m aus. Für $m = 0$ ergibt sich daraus $t_1 = t_0 + h_0$, wobei der fiktive Zeitpunkt t_0 sich von dem des gegebenen Anfangswertes unterscheidet, wir bezeichnen ihn deshalb mit $\widetilde{t_0}$. Die Startschrittweite des Peer-Verfahrens h_0 bezeichnen wir mit h_p, damit ergibt sich

$$t_{0,i} = \widetilde{t_0} + c_i h_p. \tag{3.18}$$

	$s = 3$ PeerKry3	$s = 4$ PeerKry4	$s = 5$ PeerKry5
c_1	0.4385371847140350	0.1661225026730741	0.2068377401453823
c_2	0.8743710492192502	0.4145497896735533	0.3951241118982431
c_3	1.0000000000000000	0.7042604619720084	0.6199266734460809
c_4		1.0000000000000000	0.8406000177315648
c_5			1.0000000000000000
g_{21}	0.4358338645052150	0.2484272870004789	0.1882863717528655
g_{31}	0.4805420905198220	0.2243553795746857	0.1664873086357274
g_{41}		0.2112962998724116	0.1510411365150871
g_{51}			0.1531895778101022
g_{32}	0.0809207247661426	0.3137825797242480	0.2466016246649778
g_{42}		0.3138914292536178	0.2590889022811201
g_{52}			0.2234013037887930
g_{43}		0.3086897682008952	0.2236322387899814
g_{53}			0.2999378263874648
g_{54}			0.1166335518682632
γ	0.1869928069686800	0.1205215848722439	0.0947726533677875
α	86.1°	83.2°	75.7°
err	0.0099	0.0021	0.00046
$\|B\|_\infty$	8.46	1.61	4.85

Tabelle 3.1: Koeffizienten superkonvergenter einfach impliziter Peer-Methoden

Einer der Zeitpunkte $t_{0,i}$ soll der gegebenen Anfangszeit t_0 entsprechen. Damit keine Anfangswerte zu Zeitpunkten die kleiner als t_0 sind, berechnet werden müssen, soll gelten:

$$t_0 = \min_{i=1,2,\ldots,s} t_{0,i} = \min_{i=1,2,\ldots,s} \widetilde{t_0} + c_i h_p = \widetilde{t_0} + c_{min} h_p$$
$$\text{mit} \qquad c_{min} = \min_{i=1,2,\ldots,s} c_i$$

Wir erhalten also $\widetilde{t_0} = t_0 - c_{min} h_p$ und können jetzt $\widetilde{t_0}$ in (3.18) ersetzen. Wir erhalten

$$t_{0,i} = t_0 + (c_i - c_{min}) h_p. \tag{3.19}$$

Damit der Einfluss der Berechnung der Startwerte auf die weitere Rechnung mit der Peer-Methode so gering wie möglich bleibt, wollen wir, dass ROWMAP genau einen Zeitschritt mit der Schrittweite h_{Start} ausgehend von t_0 für den größten der Zeitpunkte

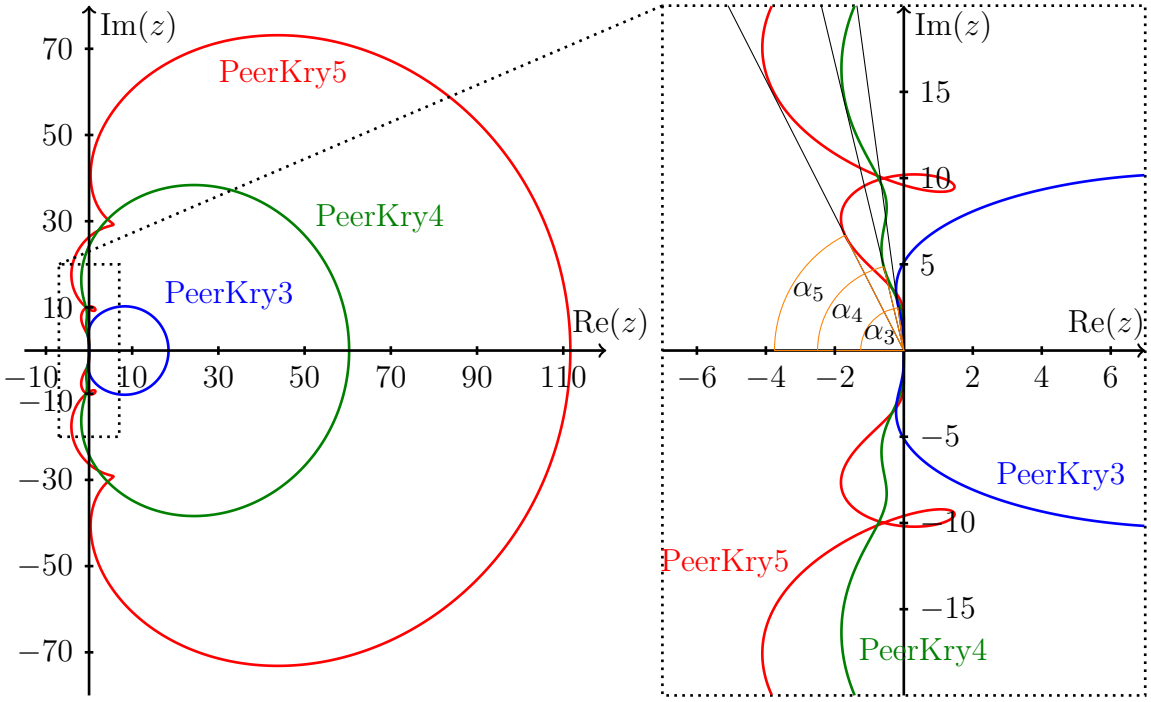


Abbildung 3.1: Stabilitätsgebiete der einfach impliziten Peer-Verfahren PeerKry3, PeerKry4 und PeerKry5. Die Methoden sind L(α)-stabil mit den im rechten Bild eingezeichneten Winkeln $\alpha_3 = 86.1°$, $\alpha_4 = 83.2°$ und $\alpha_5 = 75.7°$

$t_{0,i}$ macht.

$$t_0 + h_{start} = \max_{i=1,2,\ldots,s} t_{0,i} = \max_{i=1,2,\ldots,s} t_0 + (c_i - c_{min})h_p = t_0 + (c_{max} - c_{min})h_p$$

mit $$c_{max} = \max_{i=1,2,\ldots,s} c_i$$

$$\Rightarrow \qquad h_p = \frac{h_{Start}}{c_{max} - c_{min}}$$

Jetzt können wir noch h_p in (3.19) ersetzen und erhalten

$$t_{0,i} = t_0 + \frac{c_i - c_{min}}{c_{max} - c_{min}} h_{Start}. \tag{3.20}$$

Die Startwerte werden mit ROWMAP mit einer schärferen Toleranz von

$$atol_{ROWMAP} = \max(0.01 \cdot atol_{Peer}, 10^{-12}), \quad rtol_{ROWMAP} = \max(0.01 \cdot rtol_{Peer}, 10^{-12})$$

berechnet, $atol_{Peer}$ und $rtol_{Peer}$ sind dabei die für die Peer-Methode verwendeten Toleranzen. Wir führen mit ROWMAP ausgehend von t_0 genau einen akzeptierten Zeitschritt aus und lassen uns die verwendete Schrittweite und die Lösung zurückgeben. Die zurückgegebene Schrittweite ergibt dann h_{Start} und die Lösung gehört zum größten der Zeitpunkte $t_{0,i}$ mit c_{max}. Der Anfangswert y_0 wird dem Zeitpunkt $t_{0,i}$ der zu c_{min} gehört zugeordnet. Die restlichen $s - 2$ Startwerte werden mit ROWMAP ausgehend von t_0 mit den Schrittweiten $(c_i - c_{min})/(c_{max} - c_{min})h_{Start}$ berechnet.

	PeerKry3	PeerKry4	PeerKry5
b_1	-0.2882496313666936	1.2951093310625457	-0.7434180486076123
b_2	1.2882496313666936	-3.4264772596721884	2.5484337050065160
b_3		3.1313679286096430	-3.7318303324971054
b_4			2.9268146760982012

Tabelle 3.2: Koeffizienten zur Berechnung der eingebetteten Lösung für die drei einfach impliziten Verfahren PeerKry3, PeerKry4 und PeerKry5

Schrittweitensteuerung

Die Schrittweitensteuerung wird mit Hilfe einer eingebetteten Lösung $\widetilde{Y}_{m,s}$ realisiert. Die Berechnung von $\widetilde{Y}_{m,s}$ erfolgt durch Extrapolation. Dazu wird ein Vektor-Interpolationspolynom $p(t) = [p_1(t), p_2(t), \ldots, p_n(t)]$ zu den Punkten

$$(t_{m,1}, Y_{m,1}), (t_{m,2}, Y_{m,2}), \ldots, (t_{m,s-1}, Y_{m,s-1})$$

berechnet. Die Vergleichslösung ergibt sich dann aus

$$\widetilde{Y}_{m,s} = p(t_{m,s}) = \sum_{i=1}^{s-1} b_i Y_{m,i}.$$

Die Koeffizienten b_i lassen sich leicht mit Hilfe der Lagrange-Polynome berechnen. Die Lagrange-Darstellung des Polynoms $p(t)$ ist gegeben durch

$$p(t) = \sum_{i=1}^{s-1} L_i(t) Y_{m,i} \qquad \text{mit} \qquad L_i(t) = \prod_{\substack{j=1 \\ j\neq i}}^{s-1} \frac{t - t_{m,j}}{t_{m,i} - t_{m,j}}, \quad i = 1, 2, \ldots, s-1$$

Jetzt werten wir die Lagrange-Polynome an der Stelle $t_{m,s}$ aus und erhalten

$$b_i = L_i(t_{m,s}) = \prod_{\substack{j=1 \\ j\neq i}}^{s-1} \frac{t_{m,s} - t_{m,j}}{t_{m,i} - t_{m,j}} = \prod_{\substack{j=1 \\ j\neq i}}^{s-1} \frac{t_m + h_m - t_m - c_j h_m}{t_m + c_i h_m - t_m - c_j h_m} = \prod_{\substack{j=1 \\ j\neq i}}^{s-1} \frac{1 - c_j}{c_i - c_j},$$

für $i = 1, 2, \ldots, s-1$. Die Koeffizienten b_i sind also nur von den Knoten c_i abhängig und lassen sich aus diesen sehr leicht berechnen. Für die drei Krylov-Peer Methoden PeerKry3,PeerKry4 und PeerKry5 haben wir diese berechnet und in Tabelle 3.2 angegeben.
Den Fehler schätzen wir in der üblichen Art und Weise mit

$$\text{est} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(\frac{(\widetilde{Y}_{m,s})_i - (Y_{m,s})_i}{atol + rtol\,|(Y_{m-1,s})_i|} \right)^2}. \tag{3.21}$$

atol und *rtol* sind die vorgegebene absolute und relative Genauigkeit. Die neue Schrittweite wird dann wie folgt berechnet

$$h_{new} = h_m \min\left(2, \max(0.2, 0.8 \cdot \text{est}^{\frac{-1}{p}})\right), \tag{3.22}$$

p ist dabei die Ordnung des Verfahrens.

Prädiktor

Damit wir im Newton-Verfahren (2.9) die Näherungslösung $Y_{m,i}^1$ berechnen können, benötigen wir einen Startwert $Y_{m,i}^0$. Dafür gibt es viele Möglichkeiten, wir wollen hier einen Ansatz verwenden, der möglichst viele bereits berechnete Informationen nutzt und somit einen guten Startwert liefert. Da wir sequentielle Peer-Verfahren betrachten, bei denen in der aktuellen Stufe bereits berechnete Funktionswerte aus dem aktuellen Schritt eingehen, wollen wir einen Prädiktor verwenden, der ebenfalls die bereits berechneten Funktionswerte benutzt. Der verwendete Prädiktor hat für die i-te Stufe die folgende Form

$$Y_{m,i}^0 = \sum_{j=1}^{s} \widehat{b}_{ij} Y_{m-1,j} + h_m \sum_{j=1}^{i-1} \widehat{g}_{ij} F_{m,j}. \tag{3.23}$$

Ein Prädiktor dieser Art wurde für Peer-Verfahren zuerst in [36] untersucht. Für skalare Probleme ergibt sich die kompakte Schreibweise

$$Y_m^0 = \widehat{B}_m Y_{m-1} + h_m \widehat{G} F_m, \tag{3.24}$$

mit den Koeffizientenmatrizen $\widehat{B}$ und $\widehat{G}$. Die Matrix $\widehat{G}$ ist eine strenge untere Dreiecksmatrix. Wir fordern, dass der Prädiktor die Ordnung $s-1$ haben soll, d.h. $Y_m - Y_m^0 = \mathcal{O}(h^s)$, dann kann man die Matrix $\widehat{B}$ mit Hilfe der Ordnungsbedingungen aus der Matrix $\widehat{G}$ bestimmen. Analog zur Herleitung von (2.15) erhalten wir

$$V_0 = \widehat{B}_m V_1 S_m^{-1} + \widehat{G} V_0 D F_0^\top, \tag{3.25}$$

mit den Matrizen aus (2.14). Ist diese Gleichung erfüllt, besitzt der Prädiktor für beliebige Schrittweiten die Ordnung $s-1$. Sind die Knoten c_i paarweise verschieden, so sind die Matrizen V_0 und V_1 regulär und wir können (3.25) nach

$$\widehat{B}_m = (V_0 - \widehat{G} V_0 D F_0^\top) S_m V_1^{-1} \tag{3.26}$$

umstellen. Somit ist die Matrix $\widehat{B}_m$ für jedes $\widehat{G}$ und für beliebiges σ_m durch (3.26) bestimmt. Die Matrix $\widehat{G}$ kann jetzt beliebig gewählt werden. Wir wollen $\widehat{G}$ wie in [36] bestimmen. Dazu müssen wir zunächst den Nordsieck-Vektor

$$y^{[m]} = V_1^{-1} Y_{m-1} = \begin{pmatrix} y(t_m) \\ h_{m-1} y'(t_m) \\ \vdots \\ \frac{h_{m-1}^{s-1}}{(s-1)!} y^{(s-1)}(t_m) \end{pmatrix} + \mathcal{O}(h_{m-1}^s) \tag{3.27}$$

einführen. Diesen setzen wir in (3.24) ein und erhalten

$$\begin{aligned} Y_m^0 &= \widehat{B}_m V_1 y^{[m]} + h_m \widehat{G} F_{m,j} = \widehat{\Theta} S_m y^{[m]} + h_m \widehat{G} F_{m,j} \\ \text{mit} \qquad \widehat{\Theta} &= V_0 - \widehat{G} V_0 D F_0^\top. \end{aligned} \tag{3.28}$$

Wir bestimmen die Matrix $\widehat{G}$ unter der Bedingung, dass die Frobeniusnorm der skalierten Matrix $\widehat{\Theta}$ minimal wird, d.h.,

$$\left\|\widehat{\Theta}D_f\right\|_F \to \min \qquad \text{mit} \qquad D_f = \operatorname{diag}\left(1, 1, \frac{1}{2!}, \frac{1}{3!}, \ldots, \frac{1}{(s-1)!}\right).$$

Mit (3.28) erhalten wir

$$\left\|\widehat{\Theta}D_f\right\|_F = \left\|V_0 D_f - \widehat{G}V_0 D F_0^\top D_f\right\|_F \to \min. \tag{3.29}$$

Für gegebene Knoten c_i sucht man jetzt diejenige Matrix $\widehat{G}$, so dass die Frobeniusnorm in (3.29) minimal wird. Das führt auf ein lineares Ausgleichsproblem für die Koeffizienten $\widehat{g}_{ij}$ mit $j = 1, 2, \ldots, s-1$ und $i = j+1, j+2, \ldots, s$. Um dieses herzuleiten, benötigen wir den folgenden Operator.

Definition 3.9 [vect$(\cdot)$**]**
Sei $A \in \mathbb{R}^{s,s}$, wir definieren den Operator vect$\colon \mathbb{R}^{s,s} \to \mathbb{R}^{s\cdot s}$ *durch*

$$\operatorname{vect}(A) := \sum_{j=1}^{s}\sum_{i=1}^{s} a_{ij} e_{k(i,j)}, \qquad k(i,j) = (j-1)s + i$$

Dabei sind die Vektoren e_k s^2-dimensionale Einheitsvektoren.

Der Operator vect erzeugt einen Vektor v_A, der alle Elemente von A enthält. Der Vektor ergibt sich dabei aus den Spalten von A, die untereinander angeordnet werden

$$v_A = (a_{11}, a_{21}, \ldots, a_{s1}, a_{12}, a_{22}, \ldots, a_{s2}, a_{13}, a_{23}, \ldots, a_{ss})^\top.$$

Die Matrix $\widehat{G}$ ist eine strenge untere Dreiecksmatrix, die sich wie folgt in ihre Elemente zerlegen lässt

$$\widehat{G} = \sum_{j=1}^{s-1}\sum_{i=j+1}^{s} \widehat{g}_{ij} E_{ij} \qquad \text{mit} \qquad E_{ij} = e_i e_j^\top \in \mathbb{R}^{s,s}.$$

Diese Zerlegung setzen wir in (3.29) ein und erhalten

$$\left\| V_0 D_f - \sum_{j=1}^{s-1}\sum_{i=j+1}^{s} \widehat{g}_{ij} e_i e_j^\top V_0 D_f F_0^\top \right\|_F \to \min$$

Daraus kann man jetzt das lineare Ausgleichsproblem

$$\|Ax - b\|_2 \to \min, \tag{3.30}$$

erzeugen, mit

$$x = (\widehat{g}_{21}, \widehat{g}_{31}, \ldots, \widehat{g}_{s1}, \widehat{g}_{32}, \widehat{g}_{42}, \ldots, \widehat{g}_{s2}, \widehat{g}_{43}, \widehat{g}_{53}, \ldots, \widehat{g}_{s,s-1})^\top \quad \text{und} \quad b = \operatorname{vect}(V_0 D_f).$$

Die ℓ-te Spalte von A ergibt sich dabei aus

$$Ae_\ell = \operatorname{vect}\left(e_i e_j^\top V_0 D_f F_0^\top\right) \qquad \text{mit} \qquad \ell = (i-1) + (j-1)(s-1) - \frac{(j-1)j}{2}$$

Für gegebene Knoten c_i können wir mit (3.30) zuerst x und damit $\widehat{G}$ und schließlich mit (3.26) $\widehat{B}_m$ berechnen. Für die drei Krylov-Peer Methoden PeerKry3,PeerKry4 und Peer-Kry5 haben wir $\widehat{G}$ in Tabelle 3.3 angegeben.

	PeerKry3	PeerKry4	PeerKry5
$\widehat{g}_{21}$	0.8739363601379309	0.4173897839175595	0.3944355830316005
$\widehat{g}_{31}$	0.8589765039122383	0.1651295614765928	0.2687561117109394
$\widehat{g}_{41}$		0.4927828853168685	0.5837572805490611
$\widehat{g}_{51}$			0.5143425950232470
$\widehat{g}_{32}$	0.1410234960877617	0.5387989102421881	0.3508845549385570
$\widehat{g}_{42}$		-0.2102950292666084	-0.3261779079210321
$\widehat{g}_{52}$			-0.1045955037674921
$\widehat{g}_{43}$		0.7175121439497394	0.5830218812575633
$\widehat{g}_{53}$			0.2774411211068372
$\widehat{g}_{54}$			0.3128117876374074

Tabelle 3.3: Koeffizienten $\widehat{G}$ zur Berechnung des Prädiktors für die drei einfach impliziten Verfahren PeerKry3, PeerKry4 und PeerKry5

Abbruchkriterien

Für implizite Krylov-Peer-Verfahren sind einige Abbruchkriterien notwendig. Wir benötigen zunächst einen Test, um das Newton-Verfahren (2.9) abzubrechen. Die aktuelle Approximation $Y_{m,i}^{k+1}$ im $(k+1)$-ten Newtonschritt wird als $Y_{m,i}$ akzeptiert, wenn

$$\max_{j=1,2,\ldots,n} \frac{\left|(\Delta Y_{m,i}^{k})_j\right|}{atol + rtol\,|(Y_{m-1,s})_j|} \leq 0.1 \tag{3.31}$$

erfüllt ist. Dabei sind *atol* und *rtol* die absolute und die relative Toleranz der Fehlerschätzung (3.21) der Schrittweitensteuerung. Ist der Abbruchtest nach 10 Iterationsschritten immer noch nicht erfüllt, setzen wir $Y_{m,i} = Y_{m,i}^{10}$ und lassen die Schrittweitensteuerung entscheiden, ob der Wert genau genug ist oder nicht. Um unnötige Iterationsschritte zu vermeiden, wenn davon auszugehen ist, dass das Newton-Verfahren wahrscheinlich nicht mehr konvergiert, führen wir einen zweiten Test

$$\frac{\|\Delta Y_{m,i}^{k}\|_\infty}{\|\Delta Y_{m,i}^{k-1}\|_\infty} > \Delta tol \tag{3.32}$$

durch. Ist dieses Kriterium erfüllt, wiederholen wir den Integrationsschritt mit der halbierten Schrittweite $h_{new} = 0.5 h_m$. Damit der Integrationsschritt nur im Ausnahmefall wiederholt werden muss, nehmen wir einen relativ großen Wert für Δtol, wir setzen $\Delta tol = 10$.

Wir verwenden noch ein weiteres Abbruchkriterium. Wenn die Norm des Residuums $g(Y_{m,i})$ in (2.8) für die aktuelle Approximation $Y_{m,i}^{k}$ der folgenden Bedingung

$$\|g(Y_{m,i}^{k})\|_{2,n} > 1$$

genügt, wiederholen wir ebenfalls den Integrationsschritt mit halber Schrittweite. Dabei verwenden wir die gewichtete $l_{2,n}$-Norm, die für Vektoren $v = (v_i)_{i=1}^{n} \in \mathbb{R}^n$ wie folgt

definiert ist

$$\|v\|_{2,n} := \frac{\|v\|_2}{\sqrt{n}}.$$

Zur Lösung der linearen Gleichungssysteme (2.9) im Newton-Verfahren verwenden wir das Krylov Unterraum-Verfahren FOM. Die mit FOM berechnete Näherungslösung von (2.9) wird akzeptiert, wenn das Residuum (3.15) der Bedingung

$$\frac{\|res\|_{2,n}}{atol} \leq ktol \qquad \text{mit} \qquad ktol = \min(\vartheta, \max(0.1\vartheta, \vartheta 10^{\frac{2}{3}(6+\log_{10}(atol))})),$$

genügt. Der Wert ϑ hängt dabei von der Anzahl der Stufen der Peer-Methode ab, für $s = 3$ ist $\vartheta = 10^{-1}$ und für $s = 4, 5$ ist $\vartheta = 10^{-2}$. Die Toleranz *ktol* liegt im Intervall $[0.1\vartheta, \vartheta]$. Im Bild rechts sieht man den Verlauf von *ktol* in Abhängigkeit von der Toleranz *atol*. Es hat sich als günstig erwiesen *ktol* sukzessive zu verkleinern, wenn *atol* den Wert 10^{-6} unterschreitet. Unterhalb von $atol = 10^{-7.5}$ bleibt die Toleranz *ktol* dann konstant auf 0.1ϑ.

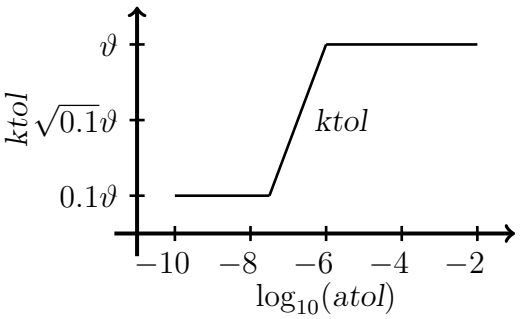


Abbildung 3.2: *ktol* in Abhängigkeit von *atol*

Die maximale Krylovdimension setzen wir auf 20. Wird dieses Maximum erreicht, wird die berechnete Näherungslösung akzeptiert und auf $\Delta Y^k_{m,i}$ gesetzt. Um unnötigen Aufwand zu vermeiden, haben wir auch hier einen zusätzlichen Abbruchtest eingebaut. Der Integrationsschritt wird mit der Schrittweite $h_{new} = 0.5h_m$ wiederholt, wenn das Residuum die Bedingung $\|res\|_{2,n} > 1$ erfüllt.

Auftreten der Jacobi-Matrix

Ein Vorteil von Krylov-Methoden ist, dass die Jacobi-Matrix J nicht explizit benötigt wird, sondern nur in Form von Matrix-Vektor-Produkten Jv auftritt. Dieses Produkt kann leicht durch einen Differenzenquotienten der Form

$$Jv = f_y\left(t_{m,i}, Y^k_{m,i}\right) v = \frac{1}{\delta}\left(f\left(t_{m,i}, Y^k_{m,i} + \delta v\right) - f\left(t_{m,i}, Y^k_{m,i}\right)\right) + \mathcal{O}\left(\delta \|v\|^2\right) \qquad (3.33)$$

mit Hilfe von nur zwei Funktionsaufrufen approximiert werden. Der Funktionswert $f\left(t_{m,i}, Y^k_{m,i}\right)$ muss dabei nicht zusätzlich berechnet werden, da er sowieso im Newton-Verfahren verwendet wird. Ein weiterer Vorteil der Vorschrift (3.33) ist, dass das Produkt mit der Jacobi-Matrix an der Stelle $Y^k_{m,i}$ approximiert wird, d.h., mit FOM wird nicht die Lösung des vereinfachten Newton-Verfahrens approximiert, sondern die des normalen Newton-Verfahrens, was zu einer besseren Konvergenz des Newton-Verfahrens führen könnte.

4 Peer-AMF-Methoden

Wir wollen in diesem Kapitel die impliziten Peer-Verfahren mit einer weiteren sehr effizienten Methode zur Lösung von sehr großen ($n \gtrsim 10000$) linearen Gleichungssystemen kombinieren. Das Verfahren, das dabei verwendet wird, nennt sich **A**pproximierende **M**atrix-**F**aktorisierung (AMF). Diese Art von Verfahren geht zurück auf klassische Splitting-Methoden, welche zuerst von Peaceman und Rachford [33] und später von Beam und Warming [1] sowie D'Yakonov [13] verwendet wurden. Im Falle der Semidiskretisierung von partiellen Differentialgleichungen mit Hilfe der Linienmethode versteht man unter AMF eine Zerlegung der Jacobi-Matrix in d Matrizen mit einfacher Struktur (oft tridiagonal), wie es z.B. in [23, 48] angewendet wurde.

Die Berechnung der Stufenwerte $Y_{m,i}$ erfolgt mit Hilfe des Newton-Verfahrens (2.9). Hierbei tritt in den linearen Gleichungssystemen die Matrix $(I_n - h_m g_{ii} T_i)$ auf, wobei T_i eine Approximation an die Jacobi-Matrix $f_y(t_{m,i}, Y^k_{m,i})$ ist. Das Hauptproblem bei der Lösung von sehr großen linearen Gleichungssystemen ist die Berechnung der LU-Zerlegung. Um die Anzahl der LU-Zerlegungen zu reduzieren, verwenden wir zum einen das vereinfachte Newton-Verfahren, das bedeutet, wir benutzen in jeder Stufe die Matrix $T_i = J_m = f_y(t_{m-1,s}, Y_{m-1,s})$. Des Weiteren werden wir uns wieder auf einfach implizite Peer-Verfahren konzentrieren, da bei diesen $g_{ii} = \gamma$ gilt. Somit tritt in allen Stufen die gleiche Matrix $(I_n - h_m g_{ii} T_i) = (I_n - h_m \gamma J_m)$ auf und wir müssen für einen kompletten Integrationsschritt nur eine LU-Zerlegung berechnen.

Wir benötigen jetzt zwar nur eine LU-Zerlegung, aber dennoch ist bei sehr großen linearen Gleichungssystem die Berechnung dieser sehr aufwendig. Deshalb versucht man, die Struktur der Matrix $(I_n - h_m \gamma J_m)$ zu vereinfachen und dabei kommt die AMF-Methode zum Einsatz. Wie der Name schon sagt, wird die Matrix $(I_n - h_m \gamma J_m)$ faktorisiert und da dies nicht exakt geschieht, wird die Matrix mit Hilfe der Faktorisierung approximiert. Dieses Vorgehen führt auf eine Folge von linearen Gleichungssystemen sehr einfacher Struktur, bei denen die Berechnung der LU-Zerlegungen sehr effizient realisiert werden kann.

Wir werden uns in diesem Kapitel zunächst mit den theoretischen Grundlagen beschäftigen. Im zweiten Abschnitt werden wir die AMF-Methode mit den Peer-Verfahren kombinieren, spezielle Peer-AMF-Verfahren konstruieren und auf Stabilitätseigenschaften eingehen. Im letzten Abschnitt befassen wir uns mit Fragen der Implementierung.

4.1 Theoretische Grundlagen

Zur Berechnung der Stufenwerte $Y_{m,i}$ müssen wir die in (2.9) auftretenden linearen Gleichungssysteme lösen. Wenn wir einfach implizite Peer-Verfahren betrachten und das vereinfachte Newton verwenden, haben diese die Form

$$\begin{aligned} (I_n - h_m\gamma J_m)\,\Delta Y_{m,i}^k &= w_i - Y_{m,i}^k + h_m\gamma f\left(t_{m,i}, Y_{m,i}^k\right) \\ Y_{m,i}^{k+1} &= Y_{m,i}^k + \Delta Y_{m,i}^k \end{aligned}, \qquad k = 0, 1, 2, \ldots \tag{4.1}$$

mit $J_m = f_y(t_{m-1,s}, Y_{m-1,s})$. Bei der AMF-Methode geht man zunächst davon aus, dass man die Jacobi-Matrix in d-Summanden aufspalten kann, es soll gelten

$$J_m = \sum_{j=1}^{d} J_{m,j}. \tag{4.2}$$

Die Matrizen $J_{m,j}$ sind dabei von sehr einfacher Struktur. Jetzt bildet man die Matrix Π_m, welche die Matrix der linearen Gleichungssysteme in (4.1) approximiert

$$\Pi_m = \prod_{j=1}^{d}(I_n - \gamma h_m J_{m,j}) \tag{4.3}$$

Die Matrix $(I_n - h_m\gamma J_m)$ in (4.1) wird durch Π_m ersetzt und wir erhalten das Newton-Verfahren in der folgenden Form

$$\begin{aligned} \Pi_m \Delta Y_{m,i}^k &= w_i - Y_{m,i}^k + h_m\gamma f\left(t_{m,i}, Y_{m,i}^k\right) \\ Y_{m,i}^{k+1} &= Y_{m,i}^k + \Delta Y_{m,i}^k \end{aligned}, \qquad k = 0, 1, 2, \ldots \tag{4.4}$$

Man kann die Matrix Π_m noch anders beschreiben, es gilt der folgende Zusammenhang

$$\begin{aligned} \Pi_m = \prod_{j=1}^{d}(I_n - \gamma h_m J_{m,j}) &= (I_n - \gamma h_m \widetilde{J}_m) = (I_n - \gamma h_m J_m) + \mathcal{O}(h_m^2), \\ \text{mit} \quad \widetilde{J}_m = \sum_{j=1}^{d} J_{m,j} &- \gamma h_m \sum_{1\le j_1<j_2\le d} J_{m,j_1}J_{m,j_2} + \gamma^2 h_m^2 \sum_{1\le j_1<j_2<j_3\le d} J_{m,j_1}J_{m,j_2}J_{m,j_3} \\ &\pm \cdots + (-1)^{d-1}\gamma^{d-1}h_m^{d-1}J_{m,1}J_{m,2}\cdots J_{m,d} \\ &= J_m + \mathcal{O}(h_m) \end{aligned} \tag{4.5}$$

Jetzt kann man (4.4) als vereinfachtes Newton-Verfahren mit der Matrix $\widetilde{J}_m$ interpretieren, indem man die Matrix J_m in (4.1) durch die Matrix $\widetilde{J}_m$ aus (4.5) ersetzt.

Bemerkung 4.1
Da die Konsistenzordnung der Peer-Methode bei Startwerten entsprechend hoher Ordnung unabhängig von J_m ist, besitzt die Peer-AMF-Methode die gleiche Konsistenzordnung. Allerdings beeinflusst das Ersetzen von J_m durch $\widetilde{J}_m$ die Konvergenz des Newton-Verfahrens und die Stabilität.

4.2 Konstruktion spezieller Methoden

Stabilität

Zur Untersuchung der Stabilität von Methoden, welche AMF verwenden, wird die Testgleichung

$$y' = (\lambda_1 + \lambda_2 + \cdots + \lambda_d)y \tag{4.6}$$

betrachet (vgl. [23, 19]). Wenn man das Peer-Verfahren (2.2) mit konstanter Schrittweite auf dieses Problem anwendet, so erhält man

$$Y_m = BY_{m-1} + h\lambda GY_m \tag{4.7}$$

und

$$Y_m = (I_s - \xi G)^{-1}BY_{m-1} = M(\xi)Y_{m-1} \tag{4.8}$$

mit $\lambda = \sum_{j=1}^{d} \lambda_j$, $\xi = h\lambda$ und der Stabilitätsmatrix $M(\xi) = (I_s - \xi G)^{-1}B$. Wir bezeichnen die exakte Lösung von (4.7) mit Y^*, es gilt

$$0 = Y^* - \xi GY^* - BY_{m-1}. \tag{4.9}$$

Wir lösen (4.7) mit dem Newton-Verfahren, wir erhalten

$$(I_s - h\lambda G)(Y_m^{k+1} - Y_m^k) = \xi GY_m^k + BY_{m-1} - Y_m^k.$$

Wir untersuchen jetzt die Konvergenz des Newton-Verfahrens mit AMF für die Testgleichung (4.6). Wenn die Iteration mit AMF konvergiert, erhält man die Stabilität der exakten Lösung Y^* der impliziten Peer-Methode. Dazu bilden wir wie in (4.3) die AMF-Matrix

$$\Pi_m = \prod_{j=1}^{d}(I_s - z_j G)$$

mit $z_j = h\lambda_j$ und ersetzen die Matrix $(I_s - h\lambda G)$ im Newton-Verfahren. Wir erhalten die AMF-Iteration

$$\Pi_m(Y_m^{k+1} - Y_m^k) = \xi GY_m^k + BY_{m-1} - Y_m^k,$$

bzw.

$$\Pi_m Y_m^{k+1} = \Pi_m Y_m^k + \xi GY_m^k + BY_{m-1} - Y_m^k. \tag{4.10}$$

Wir definieren mit $\Delta_k = Y_m^k - Y^*$ den Fehler der aktuellen Iterierten Y_m^k zur exakten Lösung des Peer-Verahrens und leiten eine Rekursion für den Fehler her, es gilt

$$\begin{aligned}
\Pi_m\Delta_{k+1} &= \Pi_m Y_m^{k+1} - \Pi_m Y^* \\
&\overset{(4.10)}{=} \Pi_m Y_m^k - \Pi_m Y^* + \xi GY_m^k - Y_m^k + BY_{m-1} \\
&\overset{(4.9)}{=} \Pi_m Y_m^k - \Pi_m Y^* + \xi GY_m^k - Y_m^k + BY_{m-1} + Y^* - \xi GY^* - BY_{m-1} \\
&= \Pi_m(Y_m^k - Y^*) + \xi G(Y_m^k - Y^*) - (Y_m^k - Y^*).
\end{aligned}$$

Wir erhalten die folgende Fehlerrekursion

$$\Pi_m \Delta_{k+1} = \Pi_m \Delta_k + \xi G \Delta_k - \Delta_k. \tag{4.11}$$

Wir stellen die Iteration nach Δ_{k+1} um und erhalten die Iterationsmatrix $R(z_1, z_2, \ldots, z_d)$

$$\begin{aligned} \Pi_m \Delta_{k+1} &= (\Pi_m - I + \xi G)\Delta_k \\ \Delta_{k+1} &= (I - \Pi_m^{-1}(I - \xi G))\Delta_k = R(z_1, z_2, \ldots, z_d)\Delta_k. \end{aligned} \tag{4.12}$$

Damit das Newton-Verfahren mit AMF unabhängig von den Startwerten gegen die exakte Lösung Y^* konvergiert, muss der Spektralradius der Iterationsmatrix kleiner eins sein. Dazu betrachten wir die Eigenwerte von $R(z_1, z_2, \ldots, z_d)$

$$\det(R(z_1, z_2, \ldots, z_d) - \zeta I) = \det(I - \Pi_m^{-1}(I - \xi G) - \zeta I).$$

Da Π_m und $G = \gamma I + G_0$ Dreiecksmatrizen sind und somit auch Π_m^{-1} eine Dreiecksmatrix ist, gilt

$$\det(R(z_1, z_2, \ldots, z_d) - \zeta I) = \det(\mathrm{diag}(1 - \frac{1 - \gamma\xi}{\prod_{j=1}^{d}(1 - \gamma z_j)}) - \zeta I)$$

Somit besitzt $R(z_1, z_2, \ldots, z_d)$ den s-fachen Eigenwert

$$\zeta = 1 - \frac{1 - \gamma\xi}{\prod_{j=1}^{d}(1 - \gamma z_j)}. \tag{4.13}$$

Wir wollen zuerst eine Bedingung herleiten, die eine Einschränkung an die Konvergenz für betragsmäßig große Werte von $h\lambda_j$ ist. Dazu betrachten wir den Fall, dass alle $z_j = r\mathrm{e}^{i\varphi}$ sind und damit $\xi = d \cdot r\mathrm{e}^{i\varphi}$ ist. Dabei ist

$$\varphi = \arg(z) \in [0, 2\pi) \qquad \text{und} \qquad r = |z|\,. \tag{4.14}$$

Wir erhalten

$$\zeta = 1 - \frac{1 - \gamma d r\mathrm{e}^{i\varphi}}{(1 - \gamma r\mathrm{e}^{i\varphi})^d} = 1 - \frac{\frac{1}{r} - \gamma d\mathrm{e}^{i\varphi}}{r^{d-1}(\frac{1}{r} - \gamma\mathrm{e}^{i\varphi})^d}$$

Wir setzen $r = 1/x$ und untersuchen den Fall $x \to 0$, es ergibt sich

$$\zeta = 1 - x^{d-1}\frac{x - \gamma d\mathrm{e}^{i\varphi}}{(x - \gamma\mathrm{e}^{i\varphi})^d}. \tag{4.15}$$

Wir entwickeln die Funktion $f(x) = (x - d\gamma\mathrm{e}^{i\varphi})/(x - \gamma\mathrm{e}^{i\varphi})^d$ im Punkt $x_0 = 0$, es gilt

$$f(x) = (-1)^{d-1}\frac{d}{\gamma^{d-1}}\mathrm{e}^{(1-d)i\varphi} + \mathcal{O}(x).$$

Wir setzen $f(x)$ in (4.15) ein und erhalten

$$\zeta = 1 + (-1)^d\frac{d}{\gamma^{d-1}}\mathrm{e}^{(1-d)i\varphi}x^{d-1} + \mathcal{O}(x^d).$$

Da wir uns für den Spektralradius von $R(z)$ interessieren, betrachten wir den Betrag von ζ, es gilt für $x \to 0$

$$|\zeta|^2 = 1 + 2(-1)^d \frac{d}{\gamma^{d-1}} x^{d-1} \cos((d-1)\varphi) + \mathcal{O}(x^d). \tag{4.16}$$

Soll die Iteration (4.10) gegen die exakte Lösung Y^* konvergieren, so müssen die Eigenwerte von $R(z)$ vom Betrag kleiner als eins seien, d.h., in (4.16) muss $|\zeta|^2 < 1$ gelten. Für den Winkel α der A(α)-Stabilität gilt $\alpha = \pi - \varphi$. Wir unterscheiden zwei Fälle

1. Fall *d* ist gerade:

$$|\zeta|^2 < 1 \quad \Leftrightarrow \quad \cos((d-1)\varphi) < 0 \quad \Leftrightarrow \quad \cos((d-1)\pi - (d-1)\alpha) < 0$$

da d gerade ist, ist $(d-1)$ ungerade und es muss gelten

$$\cos((d-1)\pi - (d-1)\alpha) < 0 \quad \Leftrightarrow \quad (d-1)\alpha < \frac{\pi}{2} \quad \Leftrightarrow \quad \alpha < \frac{\pi}{2(d-1)}$$

2. Fall *d* ist ungerade:

$$|\zeta|^2 < 1 \quad \Leftrightarrow \quad \cos((d-1)\varphi) > 0 \quad \Leftrightarrow \quad \cos((d-1)\pi - (d-1)\alpha) > 0$$

da d ungerade ist, ist $(d-1)$ gerade und es muss gelten

$$\cos((d-1)\pi - (d-1)\alpha) > 0 \quad \Leftrightarrow \quad (d-1)\alpha < \frac{\pi}{2} \quad \Leftrightarrow \quad \alpha < \frac{\pi}{2(d-1)}$$

Wir erhalten den folgenden

Satz 4.2
Wenn die AMF-Iteration (4.10) *für den Fall, dass alle* $z_j = z \in \mathbb{C}$ *sind, für alle* $|\arg(z) - \pi| \leq \alpha$ *mit* $\arg(z)$ *gegeben durch* (4.14) *und für beliebige Startwerte* $Y_m^0 \neq Y^*$ *konvergiert, so ist*

$$\alpha \leq \frac{\pi}{2(d-1)}. \tag{4.17}$$

Dabei ist d die Dimension des AMF-Splittings.

Folgerung 4.3
Ein Peer-AMF-Verfahren kann höchstens A(α)-stabil sein, mit

$$\alpha \leq \frac{\pi}{2(d-1)}.$$

Mit wachsendem d verschlechtert sich die Stabilität. Die gleiche Schranke für den α-Winkel der A(α)-Stabilität wurde für Rosenbrock-AMF-Methoden in [24] hergeleitet. Wir haben in Abbildung 4.1 den Konvergenzbereich der Methode PeerAMF4 für $d = 2$ und $d = 3$ für den Spezialfall $z_j = z,\ j = 1, 2, \ldots, d$ dargestellt. Die Eigenwerte (4.13) der Iterationsmatrix $R(z_1, z_2, \ldots, z_d) = R(z)$ haben dann die Form

$$\zeta = 1 - \frac{1 - \gamma dz}{(1 - \gamma z)^d}. \tag{4.18}$$

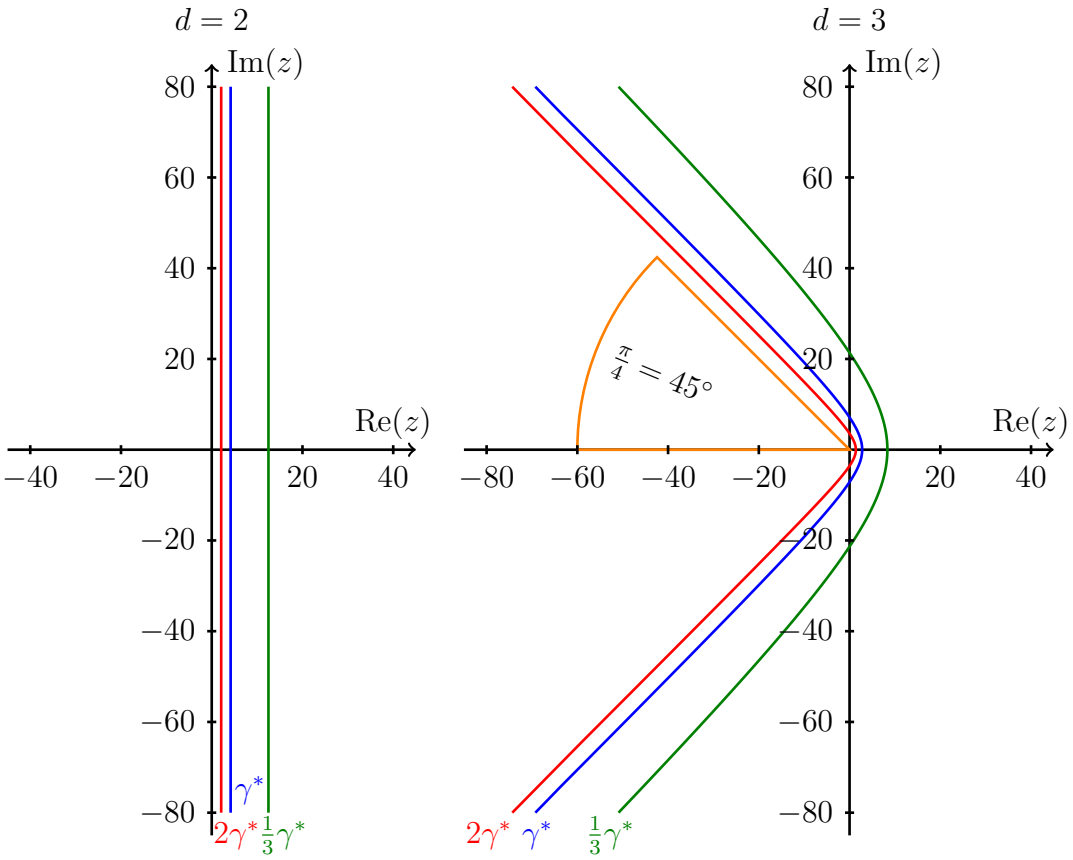


Abbildung 4.1: Im Bild ist der Rand des Konvergenzbereiches der Iterationsmatrix $R(z_1, z_2, \ldots, z_d)$ für den Fall $z_j = z, j = 1, 2, \ldots, d$ dargestellt. Wir haben den Rand für $d = 2$ (links) und für $d = 3$ (rechts) jeweils für drei verschiedene Werte für γ gezeichnet, $\gamma^*, 2\gamma^*$ und $1/3\gamma^*$. Der Parameter $\gamma^* \approx 0.1205$ gehört dabei zur 4-stufigen Peer-Methode PeerAMF4. Die Eigenwerte von $R(z)$ und damit die Lage des Randes sind nur von γ abhängig, wie man an (4.13) erkennen kann. Die Iteration (4.12) konvergiert jeweils im Bereich links vom Rand. Man sieht, dass der Konvergenzbereich für kleine Werte γ größer und für große Werte kleiner wird.

Man kann die Verschlechterung der Konvergenz und damit der Stabilität mit wachsendem d erkennen. Den Rand des Konvergenzbereiches erhält man, indem man (4.18) in ein Nullstellenproblem eines Polynoms in z überführt. Wir erhalten

$$0 = -\zeta + \zeta d\gamma z + \sum_{k=2}^{d} (-1)^k \binom{d}{k} \gamma^k z^k (1 - \zeta). \tag{4.19}$$

Nun setzt man für ζ in (4.19) Werte auf dem Einheitskreis mit Betrag eins ein, d.h., man wählt $\zeta = \mathrm{e}^{i\varphi}, \varphi \in [0, 2\pi)$ und berechnet für diese Werte die Nullstellen z, welche dann auf dem Rand des Konvergenzbereiches liegen.

Stabilitätsmatrix der Newton-AMF-Iteration

Wir wollen jetzt die Stabilität der Newton-AMF-Iteration untersuchen in Abhängigkeit von den Startwerten und der Anzahl der Newton-Schritte Wir betrachten wieder die Testgleichung (4.6) und setzen zur Vereinfachung $\lambda_i = \lambda$ für alle i. Wir erhalten somit

$$y' = (\lambda + \lambda + \cdots + \lambda)y = d\lambda y. \tag{4.20}$$

Wenn man die Peer-Methode (2.2) auf skalare Probleme anwendet, lässt sich die i-te Stufe des Verfahrens mit Hilfe des i-ten s-dimensionalen Einheitsvektors e_i wie folgt schreiben

$$\begin{aligned} Y_{m,i} &= e_i^\top B Y_{m-1} + h_m e_i^\top G_0 \widetilde{F}_m^{i-1} + h_m \gamma F_{m,i} \\ 0 &= Y_{m,i} - e_i^\top B Y_{m-1} - h_m e_i^\top G_0 \widetilde{F}_m^{i-1} - h_m \gamma F_{m,i}. \end{aligned} \tag{4.21}$$

Dabei ist $\widetilde{F}_m^{i-1}$ ein s-dimensionaler Vektor, der die bis zur Stufe $i-1$ berechneten Funktionswerte der Stufenwerte $Y_{m,j},\ j = 1, 2, \ldots, i-1$ und sonst Nullen enthält. Analog definiert man $\widetilde{Y}_m^{i-1}$:

$$\begin{aligned} \widetilde{Y}_m^{i-1} &= \left(Y_{m,1}, Y_{m,2}, \ldots, Y_{m,i-1}, 0, \ldots, 0\right)^\top, \\ \widetilde{F}_m^{i-1} &= \left(F_{m,1}, F_{m,2}, \ldots, F_{m,i-1}, 0, \ldots, 0\right)^\top. \end{aligned}$$

Wir wenden das Newton-Verfahren auf (4.21) an und erhalten

$$(I - h_m \gamma J)(Y_{m,i}^{k_i+1} - Y_{m,i}^{k_i}) = e_i^\top B Y_{m-1} + h_m e_i^\top G_0 \widetilde{F}_m^{i-1} + h_m \gamma F_{m,i}^{k_i} - Y_{m,i}^{k_i}.$$

Dabei ist $J = f_y(t_{m,i}, Y_{m,i}^{k_i})$ die Jacobi-Matrix. Bei Anwendung auf (4.20) ergibt sich mit $z = h_m \lambda$

$$(1 - \gamma dz)(Y_{m,i}^{k_i+1} - Y_{m,i}^{k_i}) = e_i^\top B Y_{m-1} + e_i^\top dz G_0 \widetilde{Y}_m^{i-1} + dz \gamma Y_{m,i}^{k_i} - Y_{m,i}^{k_i}.$$

Weiterhin ersetzen wir $(1 - \gamma dz)$ durch die AMF-Matrix $(1 - \gamma z)^d$ und definieren

$$\alpha_d := (1 - \gamma z)^d, \qquad \xi_d := \alpha_d + d\gamma z - 1, \qquad \beta_d := \frac{\xi_d}{\alpha_d}. \tag{4.22}$$

Damit erhalten wir in der i-ten Stufe der Newton-AMF-Iteration

$$Y_{m,i}^{k_i+1} = \frac{1}{\alpha_d} e_i^\top B Y_{m-1} + \frac{1}{\alpha_d} e_i^\top dz G_0 \widetilde{Y}_m^{i-1} + \beta_d Y_{m,i}^{k_i}.$$

Als nächstes benötigen wir einen Prädiktor für $k_i = 0$. Wir wählen einen sehr allgemeinen in der Form

$$Y_{m,i}^0 = e_i^\top \left(\widehat{B} Y_{m-1} + h_m \widehat{A} F_{m-1} + \widehat{R} \widetilde{Y}_m^{i-1} + h_m \widehat{G} \widetilde{F}_m^{i-1} \right) \tag{4.23}$$

$$\overset{(4.20)}{=} e_i^\top \left(\widehat{B} Y_{m-1} + dz \widehat{A} Y_{m-1} + \widehat{R} \widetilde{Y}_m^{i-1} + dz \widehat{G} \widetilde{Y}_m^{i-1} \right) \tag{4.24}$$

Zusammen mit diesem Prädiktor ergibt sich für $k_i = 1, 2, 3, \ldots$

$$
\begin{aligned}
Y^1_{m,i} &= \frac{e_i^\top}{\alpha_d}\Big[\left(B + \xi_d\widehat{B} + dz\xi_d\widehat{A}\right) Y_{m-1} + \left(dzG_0 + \xi_d\widehat{R} + dz\xi_d\widehat{G}\right) \widetilde{Y}^{i-1}_m \Big] \\
Y^2_{m,i} &= \frac{e_i^\top}{\alpha_d}\Big[\left(B + \beta_d B + \beta_d\xi_d\widehat{B} + dz\beta_d\xi_d\widehat{A}\right) Y_{m-1} \\
&\qquad + \left(dzG_0 + \beta_d dzG_0 + \beta_d\xi_d\widehat{R} + dz\beta_d\xi_d\widehat{G}\right) \widetilde{Y}^{i-1}_m \Big] \\
Y^3_{m,i} &= \frac{e_i^\top}{\alpha_d}\Big[\left\{\left(1 + \beta_d + \beta_d^2\right) B + \beta_d^2\xi_d\widehat{B} + dz\beta_d^2\xi_d\widehat{A}\right\} Y_{m-1} \\
&\qquad + \left\{dz\left(1 + \beta_d + \beta_d^2\right) G_0 + \beta_d^2\xi_d\widehat{R} + dz\beta_d^2\xi_d\widehat{G}\right\} \widetilde{Y}^{i-1}_m \Big] \\
&\qquad \vdots \\
Y^{k_i}_{m,i} &= \frac{e_i^\top}{\alpha_d}\Big[\left\{v_i B + w_i\left(\widehat{B} + dz\widehat{A}\right)\right\} Y_{m-1} + \left\{dzv_i G_0 + w_i\left(\widehat{R} + dz\widehat{G}\right)\right\} \widetilde{Y}^{i-1}_m \Big] \\
\text{mit} \qquad & v_i = 1 + \beta_d + \cdots + \beta_d^{k_i-1}, \qquad w_i = \beta_d^{k_i-1}\xi_d. \qquad (4.25)
\end{aligned}
$$

Mit den Vorfaktoren v_i und w_i bilden wir die Diagonalmatrizen

$$
V = \operatorname{diag}(v_1, v_2, \ldots, v_s), \qquad W = \operatorname{diag}(w_1, w_2, \ldots, w_s) \qquad (4.26)
$$

und definieren damit die Matrizen

$$
A_{m-1} = VB + W\left(\widehat{B} + dz\widehat{A}\right), \qquad \overline{A}_m = dzVG_0 + W\left(\widehat{R} + dz\widehat{G}\right). \qquad (4.27)
$$

Nun lässt sich die i-te Stufe wie folgt schreiben

$$
Y^{k_i}_{m,i} = \frac{e_i^\top}{\alpha_d}\left[A_{m-1}Y_{m-1} + \overline{A}_m\widetilde{Y}^{i-1}_m\right].
$$

Wir veranschaulichen jetzt für $s = 3$, wie man die Stufen nacheinander berechnet. Der neue Stufenvektor Y_m ergibt sich aus dem alten Y_{m-1}, wir erhalten

$$
\begin{aligned}
Y_m &= \begin{pmatrix} Y_{m,1} \\ Y_{m,2} \\ Y_{m,3} \end{pmatrix} \\
&= \begin{pmatrix}
\frac{e_1^\top}{\alpha_d}A_{m-1}Y_{m-1} \\
\frac{e_2^\top}{\alpha_d}\left[A_{m-1}Y_{m-1} + \overline{A}_m\begin{pmatrix}\frac{e_1^\top}{\alpha_d}A_{m-1}Y_{m-1} \\ 0 \\ 0\end{pmatrix}\right] \\
\frac{e_3^\top}{\alpha_d}\left[A_{m-1}Y_{m-1} + \overline{A}_m\begin{pmatrix}\frac{e_1^\top}{\alpha_d}A_{m-1}Y_{m-1} \\ \frac{e_2^\top}{\alpha_d}\left[A_{m-1}Y_{m-1} + \overline{A}_m\begin{pmatrix}\frac{e_1^\top}{\alpha_d}A_{m-1}Y_{m-1} \\ 0 \\ 0\end{pmatrix}\right] \\ 0\end{pmatrix}\right]
\end{pmatrix}.
\end{aligned}
$$

Da die Matrix $\overline{A}_m$ nur strenge untere Dreiecksmatrizen $G_0, \widehat{R}$ und $\widehat{G}$ enthält, lässt sich diese komplizierte Verkettung wie folgt in Matrixschreibweise formulieren

$$\begin{aligned} Y_m &= \frac{1}{\alpha_d}\left[A_{m-1}Y_{m-1} + \frac{1}{\alpha_d}\overline{A}_m\left(A_{m-1}Y_{m-1} + \frac{1}{\alpha_d}\overline{A}_m A_{m-1}Y_{m-1}\right)\right] \\ &= \frac{1}{\alpha_d}\left(A_{m-1}Y_{m-1} + \frac{1}{\alpha_d}\overline{A}_m A_{m-1}Y_{m-1} + \frac{1}{\alpha_d^2}\overline{A}_m^2 A_{m-1}Y_{m-1}\right) \\ &= \frac{1}{\alpha_d}\left(I + \frac{1}{\alpha_d}\overline{A}_m + \frac{1}{\alpha_d^2}\overline{A}_m^2\right)A_{m-1}Y_{m-1}. \end{aligned}$$

Wenn man dies für s Stufen verallgemeinert, erhält man mit $A_m = \frac{1}{\alpha_d}\overline{A}_m$

$$Y_m = \frac{1}{\alpha_d}\left(I + A_m + \cdots + A_m^{s-1}\right)A_{m-1}Y_{m-1}.$$

mit der Stabilitätsmatrix der Newton-AMF-Iteration

$$M_{AMF}(z, d, k_1, k_2, \ldots, k_s) = \frac{1}{\alpha_d}\left(I + A_m + \cdots + A_m^{s-1}\right)A_{m-1}. \tag{4.28}$$

Mit Hilfe dieser Stabilitätsmatrix können wir jetzt Stabilitätsgebiete in Abhängigkeit vom gewählten Startwert (4.23), der Anzahl der Newton-Schritte und der Dimension des AMF-Splittings berechnen. Dazu verwenden wir einen Algorithmus, der auf der sogenannten *Pseudo-arclength-continuation* [27] basiert. Dabei berechnet man ausgehend von einem Punkt $z \in \mathbb{C}$, der zum Rand des Stabilitätsgebietes gehört einen weiteren Punkt $z_1 \in \mathbb{C}$, der wieder zum Rand gehört usw. Den neuen Punkt z_1 findet man, indem man die Nullstelle $x^* \in \mathbb{R}$ der Funktion

$$F(z, x) = \varrho(M_{AMF}(z + v + x \cdot v_\perp)) - 1 \tag{4.29}$$

berechnet, wobei $\varrho(M_{AMF}(z))$ der Spektralradius von $M_{AMF}(z)$ ist. Die den komplexen Zahlen v und $v_\perp$ entsprechenden Vektoren im $\mathbb{R}^2$ sind orthogonal zueinander. Hat man die Nullstelle x^* von $F(z, x)$ gefunden, ergeben sich der neue Punkt z_1 und die neue Suchrichtung v durch

$$z_1 = z + v + x^* \cdot v_\perp, \qquad v = h\frac{z_1 - z}{|z_1 - z|},$$

wobei h eine vorgegebene Schrittweite ist. Die Zahl $v_\perp$ ergibt sich durch $v_\perp = \mathrm{Im}(v) - \mathrm{Re}(v)i$. Danach setzen wir $z = z_1$ und berechnen den nächsten Punkt. Die Bestimmung des Randes des Stabilitätsgebietes beginnen wir im Punkt $z = 0$, da dieser Punkt zum Rand gehört, weil $M_{AMF}(0, d, k_1, k_2, \ldots, k_s) = B$ ist und die Peer-Methode nullstabil und präkonsistent ist. Als erste Suchrichtung wählen wir $v = h \cdot i$. Des Weiteren verwenden wir noch eine Art Schrittweitensteuerung, die sich wie folgt beschreiben lässt

$$\begin{aligned} a &= \min(\max(\frac{x_{opt}}{|x^*|}, a_{min}), a_{max}), \qquad a_{min} = 0.2, \qquad a_{max} = 1.5, \qquad x_{opt} = 0.05 \\ h &= \min(\max(h \cdot a, h_{min}), h_{max}), \qquad h_{min} = 10^{-14}, \qquad h_{max} = 10^{-2} \\ \mathrm{err} &= \frac{|x^*|}{x_{opt}}. \end{aligned}$$

Der Schritt wird akzeptiert, wenn err < 1.01 gilt. Außerdem wird ein Schritt nicht wiederholt, wenn $h = h_{min}$ gilt. Es kann passieren, dass kein x^* gefunden werden kann, da in der vorgegeben Suchrichtung keine Nullstelle existiert. Falls dies auftritt, drehen wir die die Zahlen v und $v_\perp$ um 30° um den Nullpunkt. Um die Richtung festzulegen, in die die Vektoren gedreht werden, unterscheiden wir vier Fälle:

1. Der Punkt $z + v$ ist stabil und der Punkt $z + 0.01v_\perp$ ist stabil, dann wird gegen den Uhrzeigersinn gedreht.
2. Der Punkt $z + v$ ist stabil und der Punkt $z + 0.01v_\perp$ ist instabil, dann wird im Uhrzeigersinn gedreht.
3. Der Punkt $z + v$ ist instabil und der Punkt $z + 0.01v_\perp$ ist stabil, dann wird im Uhrzeigersinn gedreht.
4. Der Punkt $z + v$ ist instabil und der Punkt $z + 0.01v_\perp$ ist instabil, dann wird gegen den Uhrzeigersinn gedreht.

Die Zahlen v und $v_\perp$ werden außerdem nach diesem Schema gedreht, falls für die berechnete Lösung $|x^*| > 100$ gilt.

Um Stabilitätsgebiete zu berechnen, bei denen die Zahlen z auf dem Rand sehr große Beträge ($|z|$) > 1000 annehmen, verwenden wir eine Möbiustransformation, welche die linke komplexe Halbebene auf einen Kreis mit Mittelpunkt $\mu = -0.5$ und Radius $r = 0.5$ abbildet. Die Transformation lässt sich durch folgende Gleichung beschreiben

$$w = \frac{z}{1-z}, \qquad\qquad z = \frac{w}{1+w}.$$

Wir berechnen den Rand zunächst in w-Koordinaten und transformieren ihn danach in z-Koordinaten. Die Stabilitätsmatrix $M_{AMF}(z)$ werten wir dann immer im Punkt $z = w/(1 + w)$ aus.

Wie wir bereits gesehen haben, hängt die Stabilitätsmatrix der Newton-AMF-Iteration (4.28) vom Startwert (4.23) ab. Wir untersuchen, wie sich verschiedene Startwerte auf die Stabilität auswirken. Dabei betrachten wir die folgenden Startwerte:

$\boldsymbol{Y_{m-1,s}}$: $Y^0_{m,i} = Y_{m-1,s}$ für $i = 1, 2, \dots s$, das bedeutet $\widehat{A} = \widehat{R} = \widehat{G} = 0$ und $e_i^\top \widehat{B} = e_s^\top$ für $i = 1, 2, \dots, s$.

$\boldsymbol{Y_{m,i-1}}$: $Y^0_{m,1} = Y_{m-1,s}$ und $Y^0_{m,i} = Y_{m,i-1}$ für $i = 2, 3, \dots s$. Das bedeutet $\widehat{A} = \widehat{G} = 0$, $e_1^\top \widehat{B} = e_s^\top$, $e_i^\top \widehat{B} = 0^\top$ für $i = 2, 3, \dots, s$, $e_1^\top \widehat{R} = 0^\top$, $e_i^\top \widehat{R} = e_{i-1}^\top$ für $i = 2, 3, \dots, s$.

$\boldsymbol{\widehat{B}Y_{m-1}}$: $Y^0_{m,i} = p_{m-1}(t_{m,i})$ für $i = 1, 2, \dots, s$. Dabei ist $p_{m-1}(t)$ das Interpolationspolynom zu den Punkten $(t_{m-1,1}, Y_{m-1,1}), (t_{m-1,2}, Y_{m-1,2}), \dots,$ $(t_{m-1,s}, Y_{m-1,s})$ das bedeutet $\widehat{A} = \widehat{R} = \widehat{G} = 0$ und $\widehat{B} = V_0 V_1^{-1}$.

$\boldsymbol{\widehat{B}Y_{m-1} + \widehat{R}Y_m}$: $Y^0_{m,i} = p_{m,i}(t_{m,i})$ für $i = 1, 2, \dots, s$. Dabei sind $p_{m,i}(t)$ Interpolationspolynome zu den Punkten $(t_{m-1,i}, Y_{m-1,i}), \dots, (t_{m-1,s}, Y_{m-1,s})$, $(t_{m,1}, Y_{m,1}), \dots, (t_{m,i-1}, Y_{m,i-1})$. Das bedeutet $\widehat{A} = \widehat{G} = 0$, $\widehat{B}$ ist eine obere Dreiecksmatrix und $\widehat{R}$ ist eine strenge untere Dreiecksmatrix.

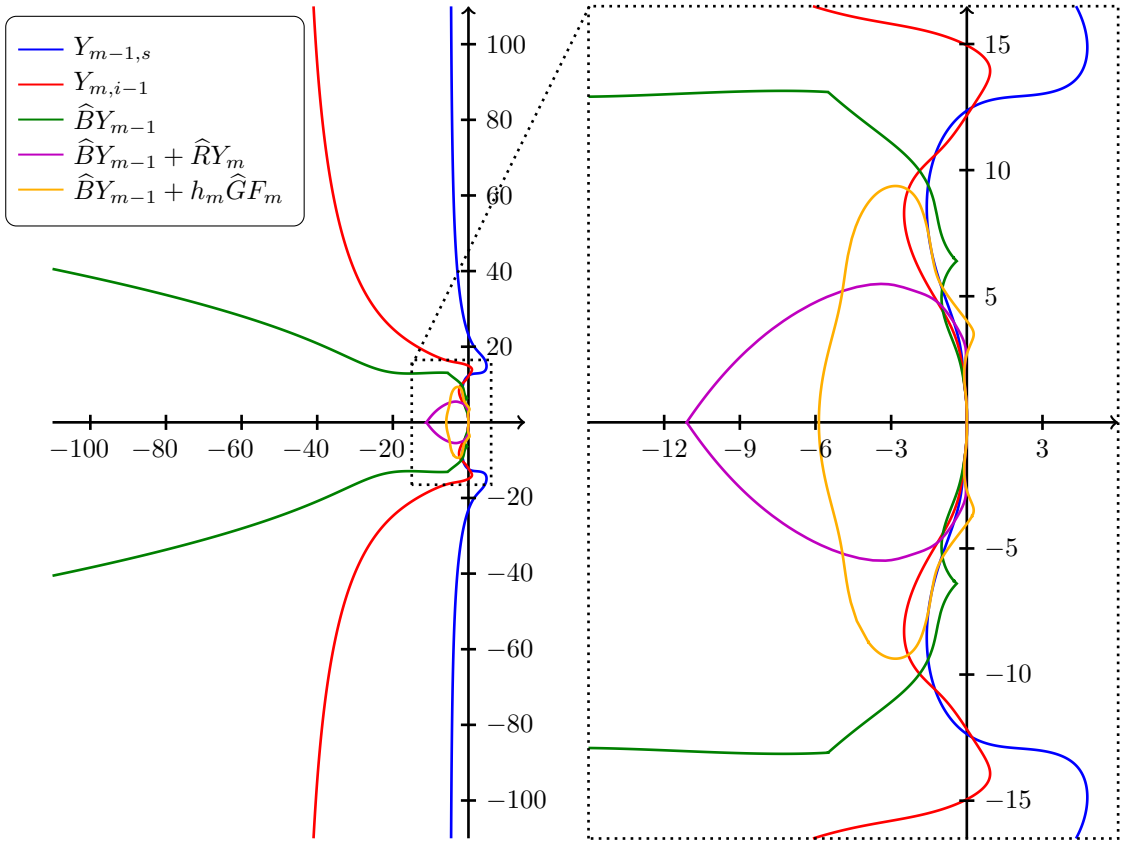


Abbildung 4.2: Es sind für $d = 2$ die Stabilitätsgebiete der Newton-AMF-Iteration für die dreistufige Peer-AMF-Methode PeerAMF3 für unterschiedliche Startwerte und fester Anzahl von Iterationsschritten dargestellt. Der Winkel der $A(\alpha)$-Stabilität dieses Verfahrens beträgt 86.5°. Für die ersten beiden Startwerte beträgt der Winkel der $A(\alpha)$-Stabilität der Newton-AMF-Iteration 78.3° bzw. 50.5°, für den dritten Startwert geht der Winkel gegen Null und für die beiden letzten Startwerte ist er nicht definiert.

$\boldsymbol{\widehat{B}Y_{m-1} + h_m\widehat{G}F_m}$ **:** $Y_m^0 = \widehat{B}Y_{m-1} + h_m\widehat{G}F_{m-1}$, vgl. (3.23), das bedeutet $\widehat{A} = \widehat{R} = 0$, $\widehat{B} = (V_0 - \widehat{G}V_0DF_0^\top)V_1^{-1}$ und $\widehat{G}$ ist eine strenge untere Dreiecksmatrix.

Wir haben für $d = 2$ und $k_i = 2$ für $i = 1, 2, \ldots, s$ die Stabilitätsgebiete für die angegeben Startwerte für das Peer-Verfahren PeerAMF3 berechnet und in der Abbildung 4.2 grafisch dargestellt. Wie man sieht sind die Stabilitätsgebiete vom Startwert abhängig. Mit dem einfachsten Startwert erhält man das größte Stabilitätsgebiet. Startwerte hoher Ordnung sind kaum stabil, verwendet man Stufenwerte oder Ableitungen aus dem aktuellen Zeitschritt, ist die Iteration nur in einem kleinem Gebiet stabil.

Bemerkung 4.4 [Anzahl der Newton-Schritte]
Da das Stabilitätsgebiet auch von der Anzahl der Newton-Schritte abhängt, haben wir auch dies untersucht. Für den ersten Startwert ändert sich das Stabilitätsgebiet nur leicht, es wird im Bereich der Null größer. Bei den letzten beiden Startwerten vergrößern sich die Stabilitätsgebiete mit zunehmender Anzahl von Newton-Schritten, allerdings sind zu viele Newton-Schritte nötig, um ein wirklich großes Stabilitätsgebiet zu erhalten,

	$s = 3$ PeerAMF3	$s = 4$ PeerAMF4	$s = 5$ PeerAMF5
c_1	0.4166758867822329	0.1921699030327529	0.2068377401453823
c_2	0.7986481474343004	0.4050665992470763	0.3951241118982431
c_3	1.0000000000000000	0.7486745001472902	0.6199266734460809
c_4		1.0000000000000000	0.8406000177315648
c_5			1.0000000000000000
g_{21}	0.3819722606520676	0.2128966962143225	0.1882863717528613
g_{31}	0.4295781440181508	0.1074150155097919	0.1664873086357186
g_{41}		0.0189350233043571	0.1510411365150738
g_{51}			0.1531895778100870
g_{32}	0.1537459691996164	0.4490895816047438	0.2466016246649800
g_{42}		0.5783370557083746	0.2590889022811269
g_{52}			0.2234013037887999
g_{43}		0.2105580179545139	0.2236322387899817
g_{53}			0.2999378263874698
g_{54}			0.1166335518682617
γ	0.1737324157139342	0.1199814936779696	0.0947726533677840
α	86.5°	82.8°	75.7°
err	0.0098	0.0026	0.00046
$\|B\|_\infty$	5.96	2.20	4.85

Tabelle 4.1: Koeffizienten superkonvergenter einfach impliziter Peer-Methoden. Die Koeffizienten von PeerAMF5 sind mit denen von PeerKry5 aus Tabelle 3.1 identisch.

das Verfahren wäre somit sehr uneffektiv.

Fazit: Wir verwenden daher bei den numerischen Tests den Startwert $Y_{m,i-1}$.

Koeffizienten

Wie am Anfang dieses Kapitels bereits beschrieben, konzentrieren wir uns auch bei den Peer-AMF-Methoden auf einfach implizite Verfahren. Die Koeffizienten der Peer-AMF-Methoden berechnen wir mit dem gleichen Algorithmus wie in Abschnitt 3.2. Bei der Kombination der Peer-Methoden mit AMF haben sich die in Tabelle 4.1 angegebenen Methoden als gut geeignet erwiesen.

	PeerAMF3	PeerAMF4	PeerAMF5
b_1	-0.5271373691429072	1.2620228797706832	-0.7434180486076123
b_2	1.5271373691429071	-2.7753932072294454	2.5484337050065160
b_3		2.5133703274587624	-3.7318303324971054
b_4			2.9268146760982012

Tabelle 4.2: Koeffizienten zur Berechnung der eingebetteten Lösung für die drei einfach impliziten Verfahren PeerAMF3, PeerAMF4 und PeerAMF5

4.3 Fragen der Implementierung

Da wir uns mit diesem Thema schon einmal in Abschnitt 3.3 beschäftigt haben, können wir einige Aussagen von dort übernehmen. Die Bestimmung der Startwerte und die Schrittweitensteuerung werden ohne Änderung aus Abschnitt 3.3 übernommen. Die Koeffizienten zur Schrittweitensteuerung der Peer-AMF-Verfahren findet man in Tabelle 4.2.

Prädiktor

Wir haben in Abschnitt 4.2 Bedingungen für die Konvergenz der AMF-Iteration bez. der Testgleichung (4.6) hergeleitet. Diese garantieren die Konvergenz für beliebige Startwerte. Für die Konvergenzgeschwindigkeit und bei nichtlinearen Problemen ist aber die Wahl geeigneter Startwerte wichtig. Wie wir bei der Untersuchung der Stabilität der Newton-AMF-Iteration gesehen haben, sind nur Prädiktoren geringer Ordnung ausreichend stabil. Wir haben uns deshalb für den Startwert $Y_{m,i-1}$

$$Y_{m,i}^0 = \begin{cases} Y_{m-1,s}, & i = 1 \\ Y_{m,i-1}, & \text{sonst.} \end{cases}$$

entschieden, welcher sich auch in umfangreichen Tests als geeignet herausgestellt hat.

Abbruchkriterien

Das Abbruchkriterium für die Newton-Iteration ist das gleiche wie bei den Krylov-Peer-Verfahren. Die aktuelle Approximation $Y_{m,i}^{k+1}$ im $(k+1)$-ten Newtonschritt wird als $Y_{m,i}$ akzeptiert, wenn die Bedingung (3.31)

$$\max_{j=1,2,\ldots,n} \frac{\left|(\Delta Y_{m,i}^k)_j\right|}{atol + rtol\,|(Y_{m-1,s})_j|} \leq 0.1$$

erfüllt ist. Ist der Abbruchtest nach 10 Iterationsschritten immer noch nicht erfüllt, setzen wir $Y_{m,i} = Y_{m,i}^{10}$ und lassen die Schrittweitensteuerung entscheiden, ob der Wert

genau genug ist oder nicht. Wir wollen auch hier unnötige Iterationsschritte vermeiden und verwenden wie für Krylov-Peer-Verfahren den Test (3.32)

$$\frac{\|\Delta Y_{m,i}^{k}\|_{\infty}}{\|\Delta Y_{m,i}^{k-1}\|_{\infty}} > \Delta tol,$$

wobei wir hier $\Delta tol = 0.5$ setzen. Dies hängt damit zusammen, dass die AMF-Iteration (4.4) nur sehr langsam konvergiert, und wenn sich $\Delta Y_{m,i}^{k}$ gegenüber $\Delta Y_{m,i}^{k-1}$ nur sehr wenig ändert, ist davon auszugehen, dass die aktuelle Näherung Y_m^k kaum noch genauer wird. Ist die Bedingung (3.32) erfüllt, brechen wir die Newton-Iteration ab und setzen $Y_{m,i} = Y_{m,i}^{k+1}$.

5 Numerische Tests

In diesem Kapitel wollen wir die in Kapitel 3 und 4 konstruierten Peer-Methoden ausführlichen numerischen Tests unterziehen. Die Koeffizienten dieser Methoden findet man in Tabelle 3.1 und Tabelle 4.1. In Abschnitt 5.1 werden wir die Verfahren zunächst an steifen Systemen geringer Dimension testen und dann in Abschnitt 5.2 auf große steife Systeme anwenden. Dazu haben wir die Methoden in Matlab implementiert. Alle Rechnungen wurden auf dem Universitäts-Rechner „pi“ des Instituts für Mathematik an der Martin-Luther-Universität Halle-Wittenberg durchgeführt. Der Rechner „pi“ ist mit zwei Intel-Xeon X5365 (3GHz) Prozessoren ausgestattet und läuft mit dem 64-Bit-Betriebssystem Ubuntu 12.04.3. Es wurde die Matlab-Version 7.13.0.564 (R2011b) verwendet.

5.1 Steife Systeme geringer Dimension

In diesem Abschnitt testen wir die Peer-Methoden an steifen Systemen geringer Dimension, welche häufig als Test-Beispiele für steife Systeme dienen, z.B. in [21]. Dabei gehen wir in Abschnitt 5.1.1 zunächst auf die Test-Beispiele ein und diskutieren dann im Abschnitt 5.1.2 die numerischen Ergebnisse.

5.1.1 Test-Beispiele

Oregonator

Der Oregonator ist ein Modell zur Beschreibung oszillierender chemischer Reaktionen. Er wurde für die Belousov-Zhabotinsky-Reaktion entwickelt. Wir haben ihn [20] bzw. [21] entnommen. Er lässt sich als ein dreidimensionales Differentialgleichungssystem darstellen, welches eine periodische Lösung besitzt. Das entsprechende Anfangswertproblem lautet

$$\begin{aligned} y'(t) &= \begin{pmatrix} 77.27\left(y_2 + y_1\left(1 - 8.375 \cdot 10^{-6} y_1 - y_2\right)\right) \\ \frac{1}{77.27}\left(y_3 - (1 + y_1)\, y_2\right) \\ 0.161\left(y_1 - y_3\right) \end{pmatrix} \\ y(0) &= (1, 2, 3)^\top \in \mathbb{R}^3, \qquad t \in [0, 360]\,. \end{aligned} \tag{5.1}$$

Hires

Dieses Testbeispiel wurde [21] entnommen und ist ausführlich in [29] beschrieben. Es beschreibt die Reaktion einer Pflanze auf den Einfluss von Strahlung mit hoher Lichtintensität (engl.: ***H**igh **I**rradiance **Res**ponse kurz Hires*). Es ergibt sich eine chemische Reaktion von acht Reaktanten. Dies führt auf ein steifes Differentialgleichungssystem mit acht Komponenten. Man erhält das folgende Anfangswertproblem

$$
\begin{aligned}
y'(t) &= \begin{pmatrix}
-1.71y_1 + 0.43y_2 + 8.32y_3 + 0.0007 \\
1.71y_1 - 8.75y_2 \\
-10.03y_3 + 0.43y_4 + 0.035y_5 \\
8.32y_2 + 1.71y_3 - 1.12y_4 \\
-1.745y_5 + 0.43y_6 + 0.43y_7 \\
-280y_6y_8 - 0.69y_4 + 1.71y_5 - 0.43y_6 + 0.69y_7 \\
280y_6y_8 - 1.81y_7 \\
-280y_6y_8 + 1.81y_7
\end{pmatrix} \\
y(0) &= (1,0,0,0,0,0,0,0.0057)^\top \in \mathbb{R}^8, \qquad t \in [0, 321.8122]\,.
\end{aligned}
\tag{5.2}
$$

Plate

Dieses Beispiel wurde [21] entnommen und beschreibt das Verhalten einer rechteckigen Platte (engl.: *plate*) unter dem Einfluss eines darüberfahrenden Fahrzeugs. Dabei simuliert die Funktion $f(x,y,t)$ die auf die Platte wirkende Kraft. Dieser Vorgang lässt sich mit Hilfe einer partiellen Differentialgleichung vierter Ordnung, welche den biharmonischen Operator Δ^2 verwendet, beschreiben

$$
\begin{aligned}
u_{tt} + \omega u_t + \sigma \Delta^2 u = f(x,y,t), \qquad & (x,y) \in \Omega = [0,2] \times [0,4/3]\,, \\
& t \in [0,7]\,.
\end{aligned}
\tag{5.3}
$$

Hierbei ist $\omega = 1000$ der Reibungsparameter und $\sigma = 100$ der Parameter zur Steuerung der Steifheit. Die Anfangsbedingungen lauten

$$u(x,y,0) = 0 \quad \text{und} \quad u_t(x,y,0) = 0.$$

Die Randbedingungen sind gegeben durch

$$u(x,y,t) = 0 \quad \text{und} \quad \Delta u(x,y,t) = 0 \quad \text{für} \quad (x,y) \in \partial\Omega.$$

Die partielle Differentialgleichung wird auf einem äquidistanten Gitter

$$(x_j, y_i) = (j\Delta x, i\Delta x), \quad \Delta x = 2/9, \quad i = 1,2,\ldots,5\,, \quad j = 1,2,\ldots,8\,,$$

der Größe 8×5 im Ort diskretisiert. Dabei wird der biharmonische Operator mit Hilfe des 13-Punkt-Differenzensterns

$$
\Delta^2_{\Delta x} = \frac{1}{\Delta x^4}
\begin{bmatrix}
 & & 1 & & \\
 & 2 & -8 & 2 & \\
1 & -8 & 20 & -8 & 1 \\
 & 2 & -8 & 2 & \\
 & & 1 & &
\end{bmatrix}
$$

diskretisiert. Dies ergibt ein gewöhnliches Differentialgleichungssystem zweiter Ordnung der Dimension 40. Eine Überführung in ein System erster Ordnung ergibt ein System der Dimension 80.

Die von den vier Rädern des Fahrzeugs ausgehende Kraft $f(x, y, t)$ wird mit Hilfe zweier Gaußkurven simuliert, welche sich in x-Richtung bewegen.

$$f(x,y,t) = \begin{cases} 200\left(\mathrm{e}^{-5(t-x-2)^2} + \mathrm{e}^{-5(t-x-5)^2}\right), & \text{für } y \in \{y_2, y_4\} \\ 0, & \text{sonst} \end{cases}$$

5.1.2 Numerische Ergebnisse

In diesem Abschnitt möchten wir die konstruierten Peer-Methoden mit den Matlab Standardintegratoren Ode15s und Ode23s vergleichen. Dabei wird für die Lösung der linearen Gleichungssysteme bei allen Integratoren die LU-Zerlegung verwendet, da die Dimension der Beispiele sehr gering ist.

- Ode15s – NDF-Methode (engl.: *numerical differentiation formula*) mit Ordnungssteuerung $p = 1, 2, \ldots, 5$, optional kann man das BDF-Verfahren (engl.: *backward differentiation formula*), auch als Gear's method bekannt, verwenden, welches aber im Normalfall weniger effizient ist, [44].
- Ode23s – basiert auf einer Rosenbrock-Methode der Ordnung 2. Da das Verfahren Ordnung 2 hat, kann es nur für grobe Toleranzen effizienter als Ode15s sein, [44].

Bei den Peer-Verfahren unterscheiden wir fünf Methoden, zwei dreistufige (PeerKry3, PeerAMF3), zwei vierstufige (PeerKry4, PeerAMF4) und ein fünfstufiges (PeerKry-AMF5), welches sowohl als Krylov-Methode als auch als AMF-Methode in Abschnitt 5.2 verwendet wird. Die entsprechenden Koeffizienten und Eigenschaften der Peer-Verfahren findet man in den Tabellen 3.1 (Seite 36) und 4.1 (Seite 54).

Wir haben die Beispiele aus dem vorherigen Abschnitt mit den aufgeführten Verfahren gerechnet. Dabei wurden die absolute (*atol*) und relative (*rtol*) Fehler-Toleranz des jeweiligen Integrators auf $atol = rtol = tol$ gesetzt. Wir haben die Toleranzen

$$tol = 10^{-2}, 10^{-2.5}, \ldots, 10^{-8}.$$

verwendet. Die bei den Test-Rechnungen erzielten Ergebnisse haben wir in Abbildung 5.1 grafisch veranschaulicht. In den Diagrammen vergleichen wir die Genauigkeit der berechneten Lösung zum Endzeitpunkt des jeweiligen Integrationsintervalls mit der dafür benötigten Rechenzeit in Sekunden. Dabei haben wir die Rechenzeit und die berechnete Genauigkeit logarithmisch skaliert dargestellt. Den Fehler der berechneten Lösung y haben wir mit der gewichteten Norm

$$\text{error} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - yref_i}{1 + |yref_i|}\right)^2} \tag{5.4}$$

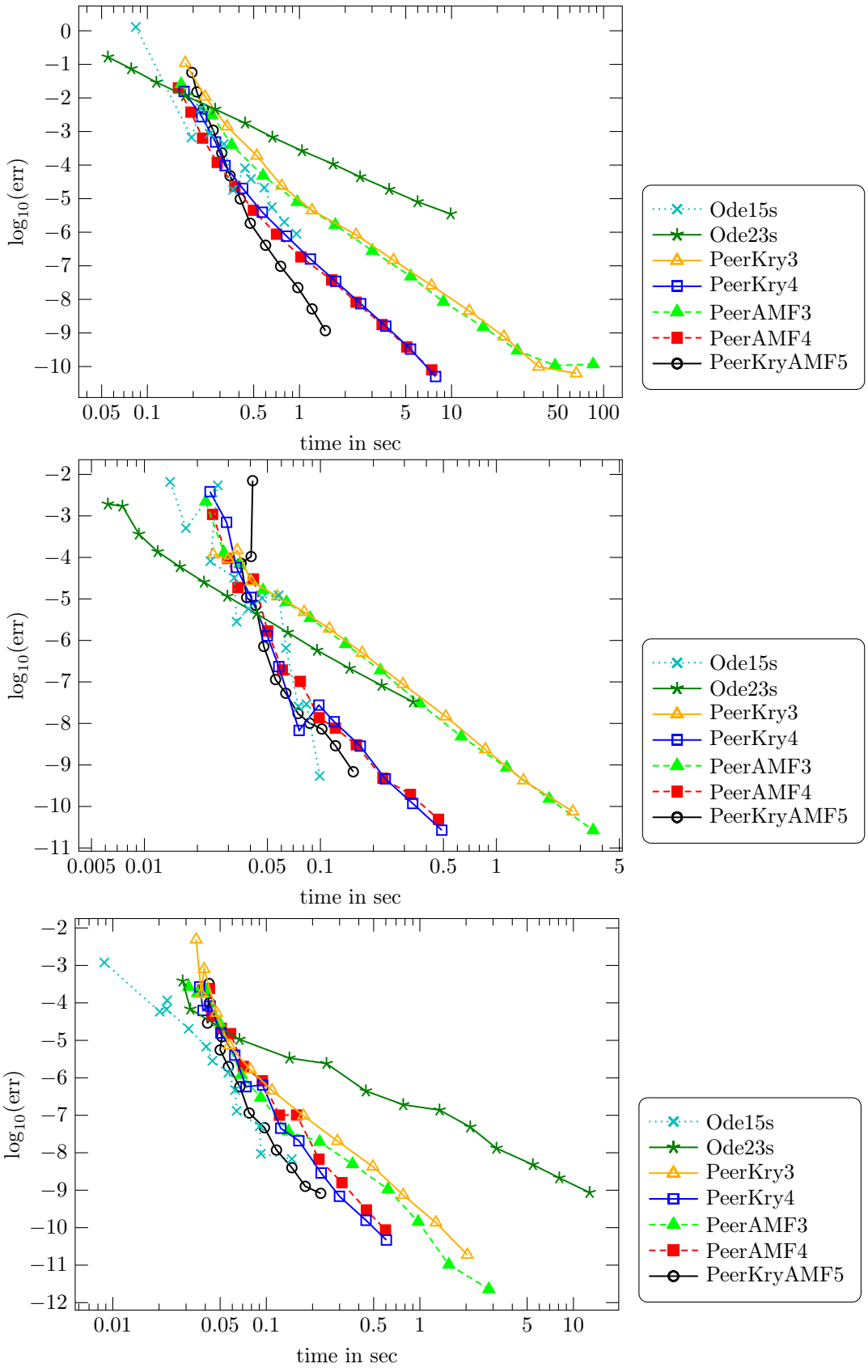


Abbildung 5.1: Ergebnisse für Oregonator (oben), Hires (Mitte) und Plate (unten)

bestimmt. Die jeweilige Vergleichslösung *yref* wurde mit Ode15s mit hoher Genauigkeit berechnet.

Die $s-1$ zusätzlichen Startwerte zu den Zeitpunkten $t_{0,i}$ (vgl. (3.20)) werden mit Ode15s mit einer schärferen Toleranz von

$$atol_{Ode15s} = 0.01 \cdot atol_{Peer}, \qquad rtol_{Ode15s} = 0.01 \cdot rtol_{Peer}$$

berechnet, $atol_{Peer}$ und $rtol_{Peer}$ sind dabei die für die Peer-Methode verwendeten Toleranzen. Im Newton-Verfahren verwenden wir denselben Abbruchtest (3.31) wie für die Krylov-Peer-Verfahren und die Peer-AMF-Methoden. Ist der Abbruchtest nach 10 Iterationsschritten immer noch nicht erfüllt, setzen wir $Y_{m,i} = Y^{10}_{m,i}$ und lassen die Schrittweitensteuerung entscheiden. Die Schrittweitensteuerung ist ebenfalls dieselbe, d.h., die neue Schrittweite wird nach Formel (3.22) bestimmt. Als Prädiktor für das Newton-Verfahren verwenden wir den gleichen wie für die Krylov-Peer-Methoden, der Startwert $Y^0_{m,i}$ wird demzufolge nach Formel (3.23) berechnet. Die Jacobi-Matrix und die LU-Zerlegung werden in jedem Schritt neu berechnet. Bei dem Beispiel Plate wird die Jacobi-Matrix nur einmal berechnet, da sie konstant ist.

Ergebnisse für Oregonator

Für grobe Toleranzen ist Ode23s am besten geeignet, es ist bei gleichem Fehler bis zu vier mal so schnell wie die Peer-Methoden. Jedoch liefert es für Toleranzen von $10^{-4.5}$ bis 10^{-8} bei gleicher Rechenzeit viel schlechtere Ergebnisse. Bei 10^{-8} ist der Fehler um drei Zehnerpotenzen schlechter als bei den dreistufigen Peer-Verfahren und fünf Zehnerpotenzen schlechter als bei den vierstufigen Peer-Verfahren. Ode15s ist für 10^{-2} und $10^{-2.5}$ schneller als die Peer-Verfahren, für feinere Toleranzen ist es jedoch langsamer als PeerKry4, PeerAMF4 und PerrKryAMF5. Bei den Peer-Methoden liefert PeerAMF4 für grobe Toleranzen die besten Ergebnisse. Das fünfstufige Peer-Verfahren ist ab 10^{-5} am schnellsten. Die dreistufigen Peer-Methoden sind für feine Toleranzen deutlich schlechter geeignet als die vier- (drei- bis viermal soviel Zeit) und fünfstufigen (fünf- bis mehr als zehnmal soviel Zeit) Methoden. Bei den dreistufigen Peer-Verfahren ist PeerAMF3 für grobe Toleranzen etwas besser als PeerKry3, für feine Toleranzen sind sie nahezu gleich gut. Bei den vierstufigen ist ebenfalls PeerAMF4 für grobe Toleranzen besser geignet als PeerKry4, auch hier sind beide für feine Toleranzen ähnlich gut.

Ergebnisse für Hires

Auch bei diesem Beispiel liefert Ode23s für die Toleranzen 10^{-2} bis 10^{-5} die besten Ergebnisse, jedoch ist es hier auch für feine Toleranzen besser als die dreistufigen Peer-Verfahren, aber schlechter als Ode15s und die vier- bzw. fünfstufigen Peer-Verfahren. Ode15s liefert für 10^{-8} das beste Ergebnis, ist aber für alle anderen Toleranzen ähnlich gut wie die vier- und fünfstufigen Peer-Verfahren. Nur für 10^{-2} und $10^{-2.5}$ ist Ode15s noch besser als die Peer-Verfahren. Bei den Peer-Verfahren sind die drei- und vierstufigen Methoden für grobe Toleranzen am besten. Für feine Toleranzen ist das fünfstufige Peer-Verfahren etwas besser als die vierstufigen Methoden. Die dreistufigen Methoden

sind jedoch für feine Toleranzen fast zehn mal langsamer als die vierstufigen Methoden. PeerKry3 und PeerAMF3 sind für dieses Beispiel gleich gut geeignet. Die vierstufigen Methoden sind ebenfalls nahezu gleich gut.

Ergebnisse für Plate

Für dieses Beispiel ist Ode15s für alle Toleranzen am besten geeignet, was daran liegt, dass es die LU-Zerlegung über mehrere Schritte konstant halten kann. Da die Dimension dieses Beispiels mit 80 deutlich größer ist als bei Hires und Oregonator, fallen diese Einsparungen deutlich ins Gewicht. Speziell bei 10^{-2} ist es dreimal so schnell wie alle anderen Codes. Bei den Peer-Methoden könnte man ebenfalls LU-Zerlegungen einsparen, dies wird aber in dieser Arbeit nicht betrachtet. Das Verfahren Ode23s ist nur für die Toleranzen 10^{-2} und $10^{-2.5}$ gut geeignet, für schärfere Toleranzen ist es deutlich schlechter als alle anderen Verfahren. Bis zu einer Toleranz von ca. 10^{-4} liefern alle Peer-Methoden ähnlich gute Ergebnisse, für $10^{-4.5}$ bis 10^{-8} ist das fünfstufige Verfahren wieder am besten geeignet. Etwas schlechter sind die vierstufigen Verfahren und am schlechtesten sind für diese Toleranzen wieder die dreistufigen Methoden. Bei den vierstufigen Peer-Verfahren ist für dieses Beispiel PeerKry4 für alle Toleranzen besser als PeerAMF4. Bei den dreistufigen Methoden ist es genau umgekehrt, hier ist PeerAMF3 etwas besser als PeerKry3.

Allgemein

Insgesamt kann man sagen, dass die vier- und fünfstufigen Peer-Verfahren vergleichbare Ergebnisse liefern wie Ode15s. Alle Peer-Methoden arbeiten robust und liefern zuverlässige Resultate, welche den Toleranzvorgaben sehr gut entsprechen. Durch Konstanthalten der Jacobi-Matrix und der LU-Zerlegung bietet sich weiteres Einsparpotential, was aber in dieser Arbeit nicht weiter untersucht wird.

5.2 Große steife Systeme

In diesem Abschnitt wollen wir die Krylov-Peer-Verfahren und die Peer-AMF-Methoden auf steife Systeme großer Dimension anwenden. Dazu stellen wir in Abschnitt 5.2.1 zuerst die Test-Beispiele vor und gehen dann in Abschnitt 5.2.2 auf die numerischen Ergebnisse ein. Wir haben die verwendeten Beispiele schon einmal ausführlich in [3] diskutiert.

5.2.1 Test-Beispiele

Zweidimensionaler Brusselator mit Diffusion

Der zweidimensionale Brusselator mit Diffusion ist wie folgt definiert

$$\begin{aligned} u_t &= 1 + u^2 v - (B+1)u + \alpha\,(u_{xx} + u_{yy}) + f(x,y,t) \\ v_t &= -u^2 v + Bu + \alpha(v_{xx} + v_{yy}), \qquad (x,y) \in \Omega = [0,1]^2, \quad t \in [0, t_{end}]. \end{aligned} \tag{5.5}$$

Wir betrachten zwei Versionen dieses Beispiels, beide mit homogenen Neumann Randbedingungen

$$\frac{\partial u}{\partial \mathbf{n}} = 0 \quad \frac{\partial v}{\partial \mathbf{n}} = 0 \quad \text{für} \quad x = 0, 1 \quad \text{und} \quad y = 0, 1.$$

Version 1: [20, S.248-249], [21, S.6]

- $t_{end} = 1, \quad B = 3, \quad \alpha = 0.02$
- Anfangsbedingungen: $u(x,y,0) = 0.5 + y, \quad v(x,y,0) = 1 + 5x$
- $f\,(x,y,t) = 0$
- Anzahl der Gitterpunkte je Raumrichtung: $M = 100$, Dimension des resultierenden Systems gewöhnlicher Differentialgleichungen: $n = 2M^2 = 20000$

Version 2: [21, S.151-152]

- $t_{end} = 11.5, \quad B = 3.4, \quad \alpha = 0.1$
- Anfangsbedingungen: $u(x,y,0) = 22y\,(1-y)^{3/2}, \quad v(x,y,0) = 27x\,(1-x)^{3/2}$
- $f\,(x,y,t) = \begin{cases} 5 & \text{falls} \quad (x-0.3)^2 + (y-0.6)^2 \leq 0.1^2 \ \text{ und } \ t \geq 1.1 \\ 0 & \text{sonst} \end{cases}$
- Anzahl der Gitterpunkte je Raumrichtung: $M = 256$, Dimension des resultierenden Systems gewöhnlicher Differentialgleichungen: $n = 2M^2 = 131072$

Version 2 ist damit deutlich anspruchsvoller (größere Steifheit, höhere Dimension). Wir diskretisieren die partielle Differentialgleichung auf einem äquidistanten Gitter

$$(x_j, y_i) = ((j-1)\Delta x, (i-1)\Delta x), \quad \Delta x = \frac{1}{M-1}, \quad 1 \leq i, j \leq M. \tag{5.6}$$

Wir bezeichnen

$$U_{ij} = U_{ij}(t) = u(x_j, y_i, t)\,, \quad V_{ij} = V_{ij}(t) = v(x_j, y_i, t) \quad \text{und} \quad f_{ij} = f(x_j, y_i, t),$$

und verwenden für die zweiten Ortsableitungen den zentralen Differenzenquotient zweiter Ordnung. Am Rand benötigen wir zusätzliche virtuelle Punkte $x_0 = y_0 = -\Delta x$, $x_{M+1} = y_{M+1} = 1 + \Delta x$ für die Randbedingungen, (nur Approximationen erster Ordnung in diesen Gitterpunkten). Wir erhalten ein semi-diskretisiertes System

$$y' = F_{diff}(y) + F_{react}(t, y)$$

mit

$$y = (U_{1,1}, U_{2,1}, \ldots, U_{M,1}, U_{1,2}, U_{2,2}, \ldots, U_{M,M}, V_{1,1}, V_{2,1}, \ldots, V_{M,1}, V_{1,2}, V_{2,2}, \ldots, V_{M,M})$$

der Dimension $2M^2$. Für jeden Gitterpunkt ergibt die Diskretisierung des Reaktionsteils

$$F^{ij}_{reakt} = \begin{pmatrix} 1 + U_{ij}^2 V_{ij} - (B+1)\, U_{ij} + f_{ij}(t) \\ -U_{ij}^2 V_{ij} + B U_{ij} \end{pmatrix}, \quad J^{ij}_{reakt} = \begin{pmatrix} \alpha_{ij} & \beta_{ij} \\ -1 - \alpha_{ij} & -\beta_{ij} \end{pmatrix} \tag{5.7}$$

mit $\alpha_{ij} = -(B+1) + 2U_{ij}V_{ij}$ und $\beta_{ij} = U_{ij}^2$. Für AMF verwenden wir ein Drei-Term-Splitting (vgl. [19]) der Jacobi-Matrix

$$J = J_r + J_x + J_y.$$

Die erste Matrix J_r resultiert aus dem Reaktionsteil J_{reakt} (5.7) und die anderen beiden Matrizen beziehen sich auf die Diskretisierung der Diffusion in x- und y-Richtung. Mit dieser Zerlegung reduziert sich das lineare Gleichungssystem $(I - \gamma h J_r)w = b$ auf die Lösung von M^2 Systemen der Dimension 2 in jedem Gitterpunkt (x_i, y_j). Die Matrix J_r hat die folgende Form:

$$J_r = \sum_{j=1}^{M} \sum_{i=1}^{M} J^{ij}_{reakt} \otimes E_{(j-1)M+i} \quad \text{mit} \quad E_k = \operatorname{diag}(e_k), \tag{5.8}$$

wobei $e_k \in \mathbb{R}^{M^2}$ der k-te Einheitsvektor der Länge M^2 ist. Die Matrix J_y ist gegeben durch

$$J_y = I_{2M} \otimes J_0, \quad J_0 = \frac{\alpha}{\Delta x^2} \begin{pmatrix} -2 & 2 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 2 & -2 \end{pmatrix} \in \mathbb{R}^{M,M}, \tag{5.9}$$

wobei I_{2M} die $2M$-dimensionale Einheitsmatrix ist. Die linearen Gleichungssysteme der Form $(I - \gamma h J_y)w = b$ reduzieren sich auf $2M$ tridiagonale Systeme der Dimension M, welche getrennt voneinander gelöst werden können.

$$(I - \gamma h J_0)\, U_j = b_u^j, \quad (I - \gamma h J_0)\, V_j = b_v^j, \quad 1 \le j \le M,$$

mit $U_j = (U_{ij})_{i=1}^M$ und $V_j = (V_{ij})_{i=1}^M$.

Zuletzt betrachten wir die linearen Gleichungssysteme der Form $(I - \gamma\tau J_x)w = b$ mit $J_x = I_2 \otimes (J_0 \otimes I_M)$. Auch diese reduzieren sich zu $2M$ tridiagonalen Systemen der Dimension M mit derselben Matrix J_0 wie für J_y jedoch mit umgekehrter Reihenfolge der Indizes:

$$(I - \gamma\tau J_0)\, U_i = b_u^i, \quad (I - \gamma\tau J_0)\, V_i = b_v^i, \quad 1 \le i \le M,$$

jetzt mit $U_i = (U_{ij})_{j=1}^M$ und $V_i = (V_{ij})_{j=1}^M$.

Radiation-Diffusion

Dieses Problem ist ein System zweier stark nichtlinearer Diffusionsgleichungen mit sehr steifem Reaktionsterm [31], [24, S.441]. Die Strahlungsenergie $E(x,y,t)$ und die Materialtemperatur $T(x,y,t)$ werden durch ein gekoppeltes partielles Differentialgleichungssystem beschrieben. Dieses hat die Form:

$$\begin{aligned} E_t &= \nabla\cdot(D_1\nabla E) + \sigma\left(T^4 - E\right) \\ T_t &= \nabla\cdot(D_2\nabla T) - \sigma\left(T^4 - E\right) \end{aligned}, \quad (x,y)\in\Omega=[0,1]^2, \quad t\in[0,3] \tag{5.10}$$

mit $\sigma = \dfrac{Z^3}{T^3}, \quad D_1 = \dfrac{1}{3\sigma + \frac{\|\nabla E\|_2}{E}}, \quad D_2 = kT^{\frac{5}{2}}, \quad k = 5\cdot 10^{-3},$

wobei $Z(x,y) = \begin{cases} Z_0 & \text{wenn } \left|x-\frac{1}{2}\right| \le \frac{1}{6} \text{ und } \left|y-\frac{1}{2}\right| \le \frac{1}{6} \\ 1 & \text{sonst.} \end{cases}$

$Z(x,y)$ ist die atomare Massenzahl. Für $Z_0 \neq 1$ ist das Material inhomogen, wir betrachten $Z_0 = 1$ und $Z_0 = 10$. Für $Z_0 = 10$ hat der nichtlineare Quellterm in (5.10) einen Sprung, wodurch die Berechnung der Lösung anspruchsvoller wird.

Die Anfangswerte sind räumlich konstant,

$$E(x,y,0) = 10^{-5}, \quad T(x,y,0) = E(x,y,0)^{1/4}$$

und die Randbedingungen sind gegeben durch

$$\frac{1}{4}E - \frac{1}{6\sigma}E_x = 1 \text{ bei } x=0, \quad \frac{1}{4}E + \frac{1}{6\sigma}E_x = 0 \text{ bei } x=1, \quad T_x = 0 \text{ bei } x = 0,1,$$

zusammen mit homogenen Neumann Randbedingungen für E und T bei $y = 0,1$.

Die Ortsdiskretisierung ist realisiert auf einem Gitter der Form

$$(x_j, y_i) = \left((j-\frac{1}{2})\Delta x, (i-\frac{1}{2})\Delta x\right), \quad \Delta x = \frac{1}{M}, \quad 1\le i,j\le M, \tag{5.11}$$

wobei die Gitterpunkte die Mittelpunkte von Zellen sind. Die zweiten Ableitungen im Ort werden mit dem konservativen zentralen Differenzenquotienten zweiter Ordnung approximiert [24]. Wir bezeichnen

$$E_{ij} = E_{ij}(t) = E(x_j, y_i, t) \quad \text{und} \quad T_{ij} = T_{ij}(t) = T(x_j, y_i, t),$$

und erhalten ein semi-diskretisiertes Anfangswertproblem der Dimension $2M^2$

$$y' = F_{diff}(y) + F_{reakt}(t,y)$$

mit

$$y = (E_{1,1}, E_{2,1}, \ldots, E_{M,1}, E_{1,2}, E_{2,2}, \ldots, E_{M,M}, T_{1,1}, T_{2,1}, \ldots, T_{M,1}, T_{1,2}, T_{2,2}, \ldots, T_{M,M}).$$

In jedem Gitterpunkt ergibt sich der folgende nichtlineare Reaktionsterm

$$F^{ij}_{reakt} = \frac{Z^3_{ij}}{T^3_{ij}}\left(T^4_{ij} - E_{ij}\right)\begin{pmatrix}1\\-1\end{pmatrix}, \quad J^{ij}_{reakt} = Z^3_{ij}\begin{pmatrix}-\alpha_{ij} & \beta_{ij}\\ \alpha_{ij} & -\beta_{ij}\end{pmatrix}$$

mit $\alpha_{ij} = 1/T_{ij}^3$ und $\beta_{ij} = 1 + 3E_{ij}/T_{ij}^4$.

Für AMF verwenden wir ein Drei-Term-Splitting $J = J_r + J_x + J_y$ analog zum Brusselator. Genau wie dort müssen wir für den Reaktionsteil M^2 lineare Gleichungssysteme der Dimension 2 lösen. Beim Diffusionsteil reduzieren wir die Bandbreite der Matrizen J_x, J_y derart, dass nur noch $2M$ tridiagonale Systeme der Dimension M gelöst werden müssen, vgl. [34].

Combustion

Das letzte Test-Beispiel ist ein nichtlineares dreidimensionales Problem aus der Verbrennungstheorie [46]:

$$\begin{aligned} c_t &= \Delta c - Dce^{-\delta/T} \\ L\,T_t &= \Delta T + \alpha Dce^{-\delta/T}, \end{aligned} \quad (x,y) \in \Omega = [0,1]^3, \quad t \in [0, 0.3]. \tag{5.12}$$

Dabei verwenden wir $L = 0.9$, $\alpha = 1$, $\delta = 20$, $D = \frac{Re^\delta}{\alpha\delta}$ und $R = 5$. Die Anfangsbedingungen sind gegeben durch

$$c\,(x, y, z, 0) = T\,(x, y, z, 0) = 1.$$

Auf dem Rand von Ω haben wir Neumann-Randbedingungen bei $x = 0, y = 0, z = 0$ und Dirichlet-Randbedingungen

$$c\,(x, y, z, t) = T\,(x, y, z, t) = 1 \quad \text{bei} \quad x = 1, y = 1, z = 1\,.$$

Wir diskretisieren dieses Problem auf einem äquidistantem dreidimensionalen Gitter der Form

$$(x_j, y_i, z_k) = \left((j - \frac{1}{2})\Delta x, (i - \frac{1}{2})\Delta x, (k - \frac{1}{2})\Delta x\right), \quad \Delta x = \frac{1}{M + \frac{1}{2}}, \quad 1 \le i, j, k \le M. \tag{5.13}$$

Wir verwenden für die Diskretisierung der Diffusion wieder den zentralen Differenzenquotienten. Wir bezeichnen

$$C_{ijk} = c(x_j, y_i, z_k, t) \quad \text{and} \quad T_{ijk} = T(x_j, y_i, z_k, t)$$

und erhalten ein semi-diskretisiertes Anfangswertproblem der Dimension $2M^3$ der Form

$$y' = F_{diff}(y) + F_{reakt}(y) \tag{5.14}$$

mit

$$\begin{aligned} y = (&C_{1,1,1}, C_{2,1,1}, \ldots,\ C_{M,1,1}, C_{1,2,1} \ldots, C_{M,M,1}, C_{1,1,2}, \ldots, C_{M,M,M} \\ &T_{1,1,1}, T_{2,1,1}, \ldots, T_{M,1,1}, T_{1,2,1} \ldots, T_{M,M,1}, T_{1,1,2}, \ldots, T_{M,M,M})^\top. \end{aligned}$$

In jedem Gitterpunkt erhalten wir den folgenden Reaktionsterm

$$F^{ijk}_{reakt}(C_{ijk}, T_{ijk}) = \mu C_{ijk} \begin{pmatrix} -1 \\ \alpha/L \end{pmatrix}, \quad J^{ijk}_{reakt}(C_{ijk}, T_{ijk}) = \mu \begin{pmatrix} -1 & -\nu \\ \alpha/L & \nu\alpha/L \end{pmatrix} \tag{5.15}$$

mit $\mu = D\mathrm{e}^{-\delta/T_{ijk}}$ und $\nu = C_{ijk}\delta/T^2_{ijk}$.

Für AMF verwenden wir eine Zerlegung der Jacobi-Matrix in vier Terme $J = J_r + J_x + J_y + J_z$, wobei J_r die Diskretisierung des Reaktionsteils (5.15) enthält und J_x, J_y, J_z zum Diffusionsteil in die entsprechende Richtung gehören. Wie bei den vorherigen Beispielen reduzieren sich die Systeme $(I - \gamma\tau J_r)w = b$ auf M^3 Systeme der Dimension 2. Beim Diffusionsteil reduzieren sich die Systeme bezüglich J_x, J_y, J_z jeweils auf $2M^2$ tridiagonale Systeme der Dimension M.

Bemerkung 5.1
Für die AMF-Methoden verwenden wir in allen Stufen dieselben Matrizen J_x, J_y, J_z im Punkt $(t_{m-1,s}, Y_{m-1,s})$ und die entsprechenden LU-Zerlegungen. Da die linearen Gleichungssysteme bezüglich J_r nur die Lösung von Systemen der Dimension 2 erfordern, berechnen wir in jedem Newton-Schritt die Matrix J_r neu.

5.2.2 Numerische Ergebnisse

Wir möchten nun die Krylov-Peer-Verfahren und die Peer-AMF-Methoden an den im vorherigen Abschnitt beschriebenen Beispielen testen und mit den Integratoren ROWMAP, EXP4, RadauAMF und Ode15s vergleichen. Dazu geben wir zunächst eine kurze Beschreibung der Vergleichsverfahren an.

- ROWMAP – Ist ein Krylov-W-Code, der auf den ROW-Methoden der Ordnung vier des Codes ROS4 von Hairer und Wanner [21] basiert. Wir verwenden für unsere Test die Koeffizienten der vierstufigen Methode GRK4T [26], welche sich bei den numerischen Tests in [2] und [51] als gut geeignet erwiesen hat. Die auftretenden linearen Gleichungssysteme werden mit Hilfe eines speziellen mehrfachen Arnoldi-Prozesses [39, 50] gelöst. ROWMAP wurde in Matlab [2] und Fortran [51] implementiert und kann auf der Software-Seite der Arbeitsgruppe Numerik am Institut für Mathematik der Martin-Luther-Universität Halle-Wittenberg heruntergeladen werden (*http://numerik.mathematik.uni-halle.de/forschung/software/*). Wir verwenden die Version vom 25 Mai 2009.

- EXP4 – Gehört zur Klasse der exponentiellen W-Methoden und wurde von Hochbruck, Lubich und Selhofer konstruiert. Ausführliche Betrachtungen zur Herleitung, Stabilität und Konsistenz findet man in [22]. Bei diesem Integrator werden Krylov-Techniken zur Approximation der φ_1-Funktionen verwendet

  $$\varphi_1(h\gamma J)v, \quad \varphi_1(z) = (e^z - 1)/z, \quad J = f_y(t_m, u_m). \tag{5.16}$$

  Der Matlab-Code wurde von der Software-Seite der Arbeitsgruppe Numerik an der Mathematischen Fakultät des Karlsruher Institut für Technologie heruntergeladen (*http://na.math.kit.edu/research/software.php*). Wir verwenden die Version vom 6 September 2000.

- RadauAMF – Basiert auf einer Radau-IIA Methode mit zwei Stufen (vgl. z.B. [21]). Dieses Verfahren ist eine Kollokationsmethode, welches die klassische Ordnung drei und die Stufenordnung zwei besitzt. Weiterhin hat dieses Verfahren sehr gute Stabilitätseigenschaften für steife Systeme, da es L-stabil und B-stabil ist. Außerdem ist das Verfahren steif genau ($b_i = a_{s,i}$) und da die Verfahrensmatrix A regulär ist gilt $R_0(\infty) = 0$ (vgl. z.B. [47]). Ein großer Nachteil der Radau-IIA Methode ist, dass A vollbesetzt ist und außerdem konjugiert komplexe Eigenwerte besitzt. Das bedeutet, es müssen im Newton-Verfahren lineare Gleichungssysteme der Dimension $s \cdot n$ gelöst werden. Mit Hilfe eines Ansatzes von Butcher [7] kann man durch eine Ähnlichkeitstransformation die Dimension der linearen Gleichungssysteme auf Block-Systeme der Dimension $2 \cdot n$ verringern. Dies ist aber dennoch bei sehr großen steifen Systemen zu aufwendig. Deshalb ersetzt man im Newton-Verfahren die Matrix $(I - h_m(A \otimes J))$ durch eine Matrix $(I - h_m(T \otimes J))$, wobei T die Form

  $$T = \gamma S(I_s - L)^{-1} S^{-1}$$

  hat und einige zusätzliche Bedingungen erfüllen muss (vgl. [34]). Die Matrix T hat den großen Vorteil, dass sie nur einen reellen s-fachen Eigenwert γ besitzt. Durch algebraische Umformungen im Newton-Verfahren werden die Stufen entkoppelt. Das bedeutet, man muss pro Newton-Schritt s lineare Gleichungssysteme der Dimension n lösen. Außerdem erhält man für jede Stufe die Iterationsmatrix $(I - h_m\gamma J)$, wodurch man pro Integrationsschritt nur eine LU-Zerlegung benötigt, was bei großen Systemen vorteilhaft ist. Jetzt verwendet man noch die in Abschnitt 4.1 beschriebene Approximierende Matrix-Faktorisierung, indem man $(I - h_m\gamma J)$ durch die Matrix Π_m aus (4.3) ersetzt. Dadurch reduziert sich der Aufwand noch einmal sehr stark.

  Als Prädiktor des Stufenvektors Y_m im Newton-Verfahren wird $Y_m^0 = \mathbb{1} \otimes y_m$ verwendet. Dieser hat zwar eine geringe Ordnung, ist aber aus Stabilitätsgründen vorteilhaft [34]. Damit das gesamte Verfahren mit all seinen Vereinfachungen die klassische Ordnung drei der zugrundeliegenden zweistufigen Radau-IIA Methode erreicht, sind drei Iterationen im Newton-Verfahren erforderlich (vgl. [34] und [19]). Diese Eigenschaft wird auch im Programmcode umgesetzt, es gibt im Newton-Verfahren keinen Abbruchtest, sondern es wird in jedem Zeitschritt eine feste Anzahl von drei Newton-Iterationen durchgeführt. Der von uns verwendete Code wurde mit Hilfe der Autoren von [19] aus Fortran in Matlab übertragen.

- Ode15s – siehe Abschnitt 5.1.2. Bei großen steifen Systemen mit dünn-besetzter Jacobi-Matrix (engl.: *sparse Matrix*) ist es für die Effizienz von Ode15s sehr wichtig, das man die Jacobi-Matrix im sogenannten *sparse*-Format bereitstellt. Wir haben dies, für die in Abschnitt 5.2.1 aufgeführten Beispiele, ausgenutzt.

Bei den Peer-Verfahren unterscheiden wir sechs Methoden, drei Krylov-Peer-Verfahren und drei Peer-AMF-Verfahren jeweils mit drei, vier und fünf Stufen. Wir bezeichen die Verfahren mit,

$$\text{PeerKry}s, \qquad \text{PeerAMF}s, \qquad s = 3, 4, 5,$$

wobei s die Anzahl der Stufen darstellt. Die entsprechenden Koeffizienten und Eigenschaften der Peer-Verfahren findet man in den Tabellen 3.1 (Seite 36) und 4.1 (Seite 54), wobei die Koeffizienten der Verfahren PeerKry5 und PeerAMF5 identisch sind.

Wir haben die Beispiele aus dem vorherigen Abschnitt mit den aufgeführten Verfahren gerechnet. Dabei wurden die absolute (*atol*) und relative (*rtol*) Fehler-Toleranz des jeweiligen Integrators auf $atol = rtol = tol$ gesetzt. Wir haben die Toleranzen

$$tol = 10^{-2}, 10^{-2.5}, \ldots, 10^{-8}$$

verwendet. Die bei den Test-Rechnungen erzielten Ergebnisse haben wir in den Abbildungen 5.2 bis 5.7 grafisch veranschaulicht. In den Diagrammen vergleichen wir die Genauigkeit der berechneten Lösung zum Endzeitpunkt des jeweiligen Integrationsintervalls mit der dafür benötigten Rechenzeit in Sekunden. Dabei haben wir die Rechenzeit und die berechnete Genauigkeit logarithmisch skaliert dargestellt. Den Fehler der berechneten Lösung y haben wir nach Formel (5.4) bestimmt. Die jeweilige Vergleichslösung *yref* wurde mit VODPK, RadauAMF, Ode45 oder Ode15s mit hoher Genauigkeit bestimmt. Für jedes Testbeispiel stellen wir die Ergebnisse in jeweils drei Diagrammen dar. Zuerst vergleichen wir die Krylov-Peer-Verfahren und Peer-AMF-Verfahren jeweils unter sich. Danach stellen wir die Methoden PeerKry4 und PeerAMF4 den Integratoren ROWMAP, EXP4, RadauAMF und Ode15s gegenüber.

Details zur Implementierung (Startwerte, Schrittweitensteuerung, Prädiktor, Abbruchkriterien) der Krylov-Peer-Verfahren und der Peer-AMF-Methoden findet man in den Abschnitten 3.3 und 4.3.

Bei allen Krylov-Methoden tritt die Jacobi-Matrix nur in Form von Matrix-Vektor-Produkten auf. Dafür verwenden wir in diesen Integratoren die Jacobi-Matrix nicht explizit, sondern approximieren die Produkte durch Differenzenquotienten. Bei den Krylov-Peer-Verfahren werden diese Produkte in der Newton-Iteration nach Formel (3.33) berechnet, dabei wird der Funktionswert $f(t_{m,i}, Y^k_{m,i})$ verwendet und damit das normale Newton-Verfahren approximiert. Bei den beiden anderen Krylov-Methoden ROWMAP und EXP4 wird dagegen der Funktionswert $f(t_m, y_m)$ benutzt. Das Produkt Jv wird nach

$$Jv = f_y\left(t_m, y_m\right) v = \frac{1}{\delta}\left(f\left(t_m, y_m + \delta v\right) - f\left(t_m, y_m\right)\right) + \mathcal{O}\left(\delta \left\|v\right\|^2\right) \tag{5.17}$$

berechnet.

In Tabelle 5.1 haben wir die Dimensionen der gewöhnlichen Differentialgleichungen der Test-Beispiele angegeben.

Ergebnisse für den Brusselator Version 1

Vergleich der Krylov-Peer-Verfahren: Für grobe Toleranzen erzielt PeerKry3 die besten Ergebnisse, es ist für 10^{-2} und $10^{-2.5}$ doppelt so schnell wie PeerKry5. PeerKry4 rechnet für 10^{-3} bis $10^{-4.5}$ am schnellsten bei gleichem Fehler. Ab einer Toleranz von 10^{-4} benötigt PeerKry3 die längsten Rechenzeiten. PeerKry5 ist ab einer Toleranz von

Beispiel	Gitterpunkte je Raumrichtung	Dimension
Brusselator Version 1	100	20000
Brusselator Version 2	256	131072
Radiation $Z_0 = 0$	200	80000
Radiation $Z_0 = 10$	200	80000
Combustion	40	128000
Combustion	80	1024000

Tabelle 5.1: Dimensionen der gewöhnlichen Differentialgleichungen nach Semidiskretisierung der entsprechenden partiellen Differentialgleichung

$10^{-5.5}$ am besten geeignet, es ist in diesem Bereich teilweise zehnmal so schnell wie PeerKry3 und fast doppelt so schnell wie PeerKry4.

Vergleich der Peer-AMF-Methoden: Für grobe Toleranzen ist PeerAMF3 am besten geeignet, ab einer Toleranz von $10^{-3.5}$ benötigt es bei gleicher Genauigkeit jedoch die längsten Rechenzeiten. PeerKry5 rechnet für grobe Toleranzen am längsten, liefert aber für feine Toleranzen am schnellsten ein Ergebnis. PeerKry4 liegt für grobe und feine Toleranzen genau zwischen PeerKry3 und PeerKry5.

Vergleich von Krylov-, AMF-Verfahren und Ode15s: Für dieses Beispiel rechneten alle Verfahren für alle Toleranzen erfolgreich und es wurden insgesamt nur sehr kurze Rechenzeiten benötigt. RadauAMF brauchte für grobe Toleranzen sogar nur weniger als eine Sekunde. Allgemein ist RadauAMF für grobe Toleranzen am besten geeignet, es ist ungefähr doppelt so schnell bei gleichem Fehler wie PeerAMF4. Für feinere Toleranzen liefert es allerdings bei gleicher Rechenzeit schlechtere Ergebnisse als PeerAMF4. Der Fehler von RadauAMF ist für die Toleranzen 10^{-6} bis 10^{-8} um mehr als eine Zehnerpotenz schlechter als bei PeerAMF4. PeerKry4 ist für feine Toleranzen am besten geeignet, es ist teilweise 10s schneller bei gleichem Fehler als PeerAMF4. Für grobe Toleranzen ist PeerKry4 allerdings etwas schlechter als PeerAMF4. ROWMAP und EXP4 liefern für grobe Toleranzen ähnliche Ergebnisse, sind aber für alle Toleranzen schlechter als die AMF-Verfahren und PeerKry4. Ab einer Toleranz von 10^{-6} ist EXP4 schlechter geeignet für dieses Beispiel als ROWMAP und Ode15s, es ist mehr als 10s langsamer bei gleicher Genauigkeit. ODE15s benötigt für grobe Toleranzen die längste Rechenzeit. Erst ab einer Toleranz von $10^{-5.5}$ ist es besser als EXP4 und ab $10^{-7.5}$ besser als ROWMAP. Für eine Toleranz von 10^{-8} ist es sogar besser als RadauAMF.

Ergebnisse für den Brusselator Version 2

Vergleich der Krylov-Peer-Verfahren: Bis zu einer Toleranz von 10^{-6} gibt es kaum Unterschiede zwischen den Verfahren. Nur PeerKry5 hat für die Toleranzen 10^{-3} bis 10^{-5} ein paar Schwierigkeiten und rechnet mehr als 100s langsamer als PeerKry3 und PeerKry4. Das dreistufige Verfahren knickt ab 10^{-6} deutlich ein und benötigt für 10^{-8} mehr als die dreifache Rechenzeit der beiden anderen Methoden. Für $10^{-7.5}$ und 10^{-8}

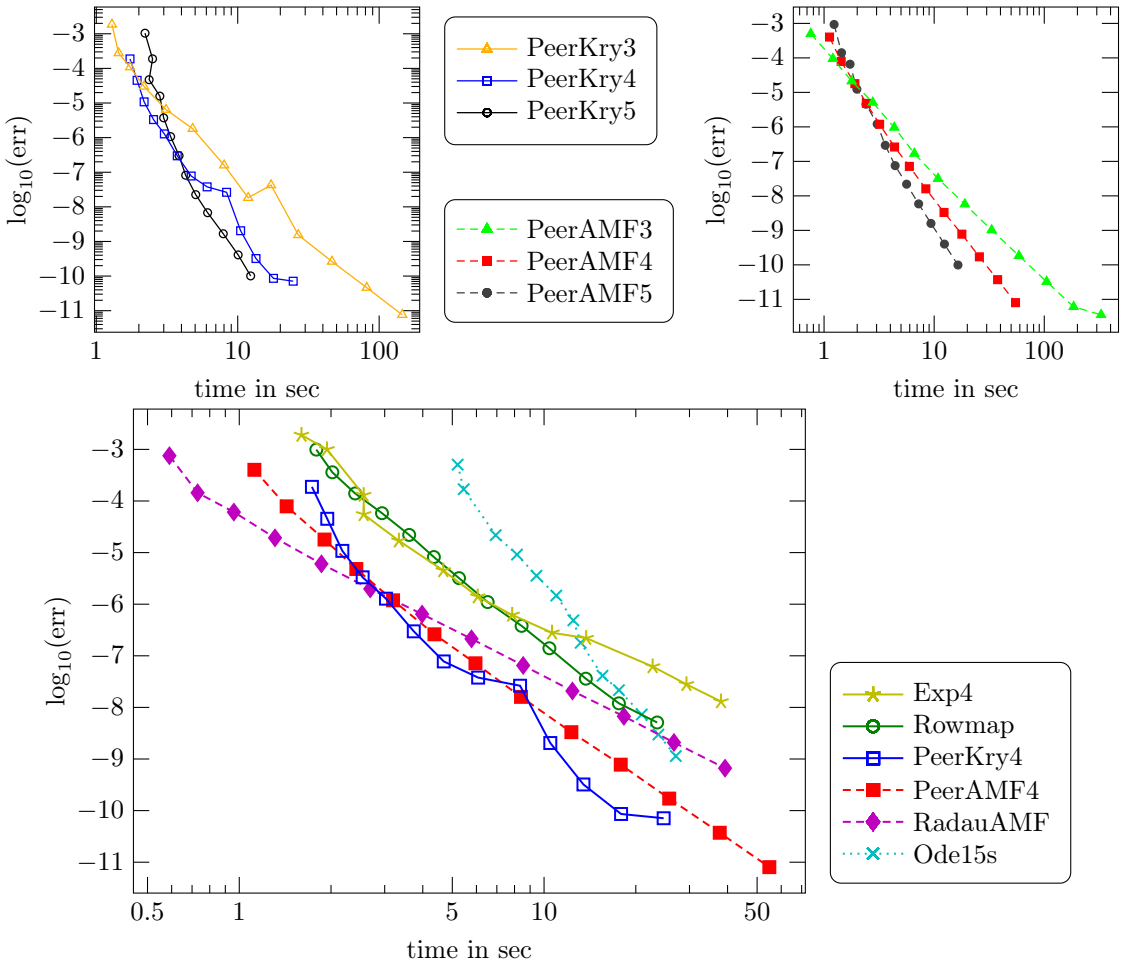


Abbildung 5.2: Ergebnisse für den zweidimensionalen Brusselator Version 1 der Dimension 20000

liefert PeerKry5 die besten Ergebnisse wärend bei PeerKry4 für 10^{-8} der Fehler wieder größer wird.

Vergleich der Peer-AMF-Methoden: Unter diesen Verfahren ist das dreistufige für dieses Beispiel am schlechtesten geeignet. Es rechnet für alle Toleranzen langsamer als die beiden anderen Methoden und benötigt bei feinen Toleranzen sogar mehr als die siebenfache Rechenzeit. Für grobe Toleranzen liefert PeerAMF4 die besten Ergebnisse, für feine Toleranzen hingegen PeerKry5. Leider stagniert bei PeerAMF5 ab einer Toleranz von 10^{-7} der Fehler. PeerKry4 erreicht die höchste Genauigkeit.

Vergleich von Krylov-, AMF-Verfahren und Ode15s: Dieses Beispiel wurde von allen Integratoren für sämtliche geforderte Toleranzen erfolgreich integriert. Insgesamt kann man sagen, dass die AMF Verfahren für dieses Problem für alle Toleranzen am besten geeignet sind, sie liefern bei gleichem Fehler am schnellsten ein Ergebnis. Die Kurven der beiden AMF-Verfahren liegen dicht beieinander, es gibt kaum Unterschiede. RadauAMF benötigt für 10^{-2} und $10^{-2.5}$ die geringste Rechenzeit, es ist bei gleicher Genauigkeit mehr als 20 mal so schnell wie Ode15s, 50 mal so schnell wie PeerKry4 und 100 mal so schnell wie ROWMAP und EXP4. Selbst bei einer Toleranz von $10^{-7.5}$

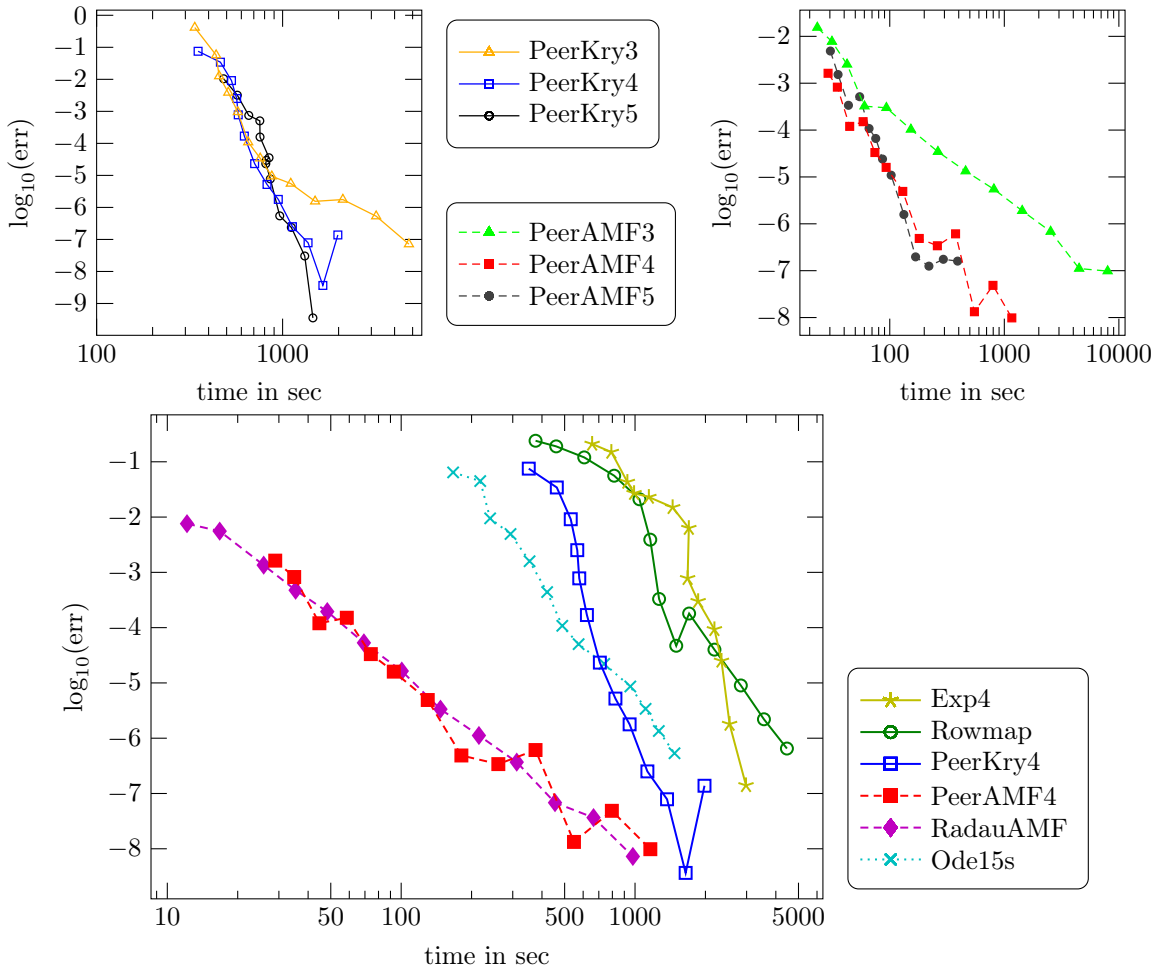


Abbildung 5.3: Ergebnisse für den zweidimensionalen Brusselator Version 2 der Dimension 131072

sind die AMF-Verfahren noch 500s schneller als PeerKry4. PeerAMF4 hat bei feinen Toleranzen ein paar Schwierigkeiten, der Fehler wird mal etwas schlechter und dann wieder besser. Ode15s liefert für grobe Toleranzen bis 10^{-6} bessere Ergebnisse als PeerKry4, es ist teilweise sogar mehr als doppelt so schnell. Allerdings ist Ode15s für feine Toleranzen ab $10^{-6.5}$ schlechter geeignet als PerrKry4. EXP4 ist für grobe Toleranzen am schlechtesten, liefert jedoch für $10^{-7.5}$ und 10^{-8} bessere Ergebnisse als ROWMAP, das somit für feine Toleranzen am schlechtesten geeignet ist.

Ergebnisse für Combustion mit $M = 40$

Vergleich der Krylov-Peer-Verfahren: PeerKry3 ist für die Toleranzen 10^{-2} und $10^{-2.5}$ am besten geeignet. Ab $10^{-3.5}$ liefert das Verfahren schlechtere Ergebnisse als zuvor, ehe es ab 10^{-5} wieder besser wird. Für 10^{-6} wächst der Fehler bei diesem Verfahren um eine Zehnerpotenz an und wird danach nur sehr langsam wieder genauer. Dabei werden außerdem sehr lange Rechenzeiten benötigt. PeerKry4 ist für grobe Toleranzen etwas schlechter als PeerKry3, ist aber für die Toleranzen 10^{-4} bis 10^{-8} am

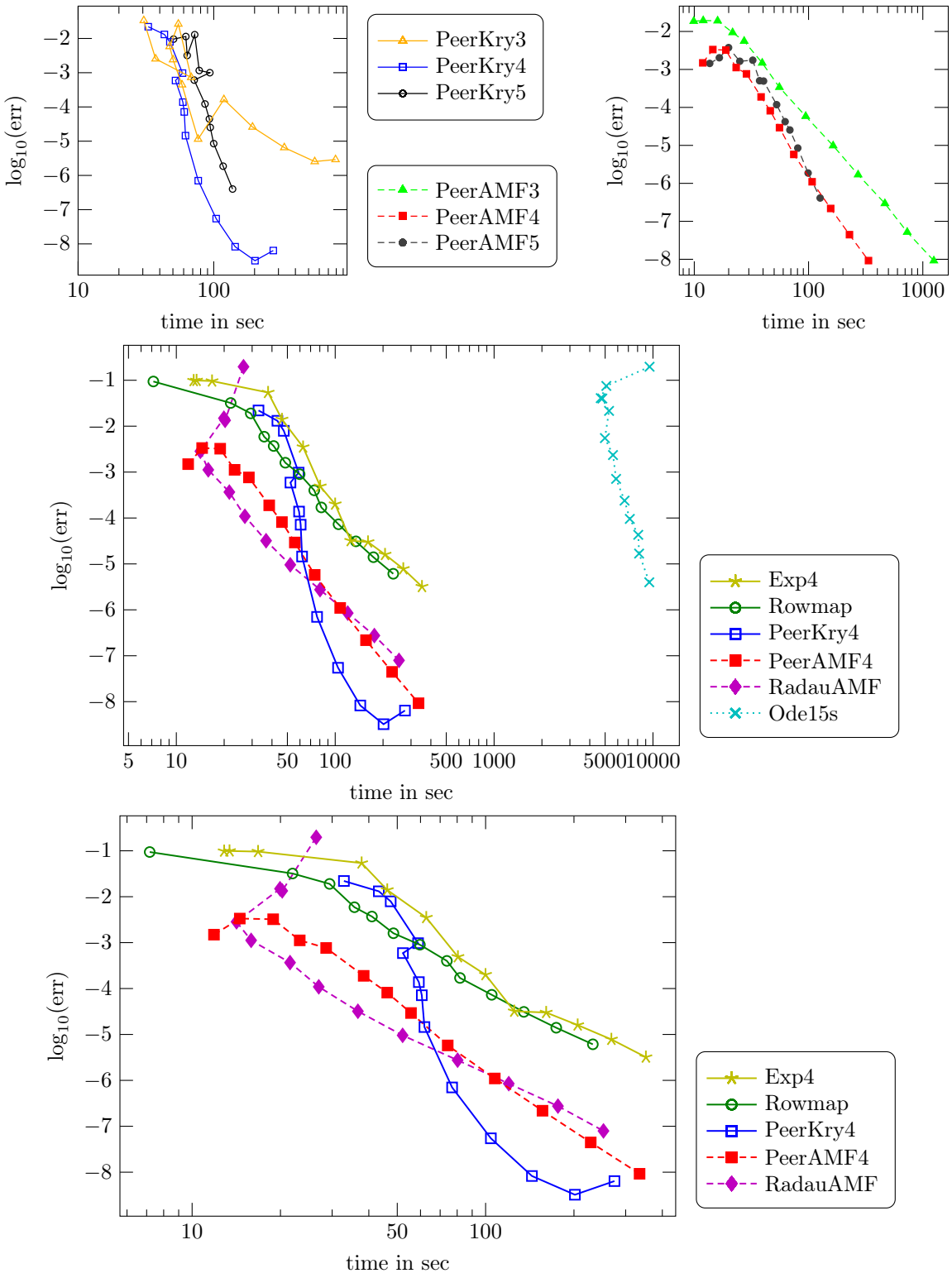


Abbildung 5.4: Ergebnisse für das dreidimensionale Combustion Problem der Dimension 128000, einmal mit Ode15s (Mitte) und einmal ohne Ode15s (unten)

besten geeignet und erreicht für $10^{-7.5}$ die höchste Genauigkeit. PeerKry5 erzielt bis 10^{-6} die schlechtesten Ergebnisse und ist danach aber besser als PeerKry3.

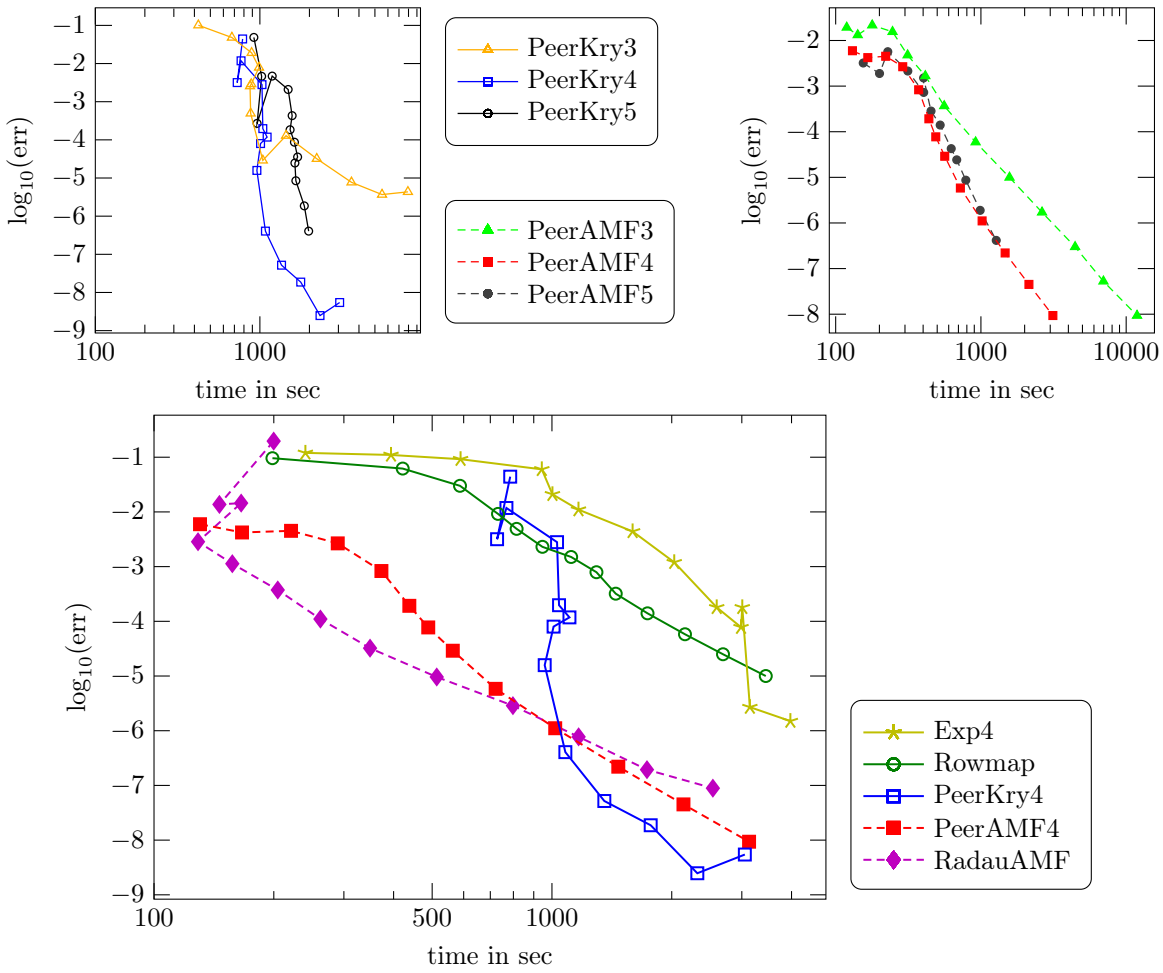


Abbildung 5.5: Ergebnisse für das dreidimensionale Combustion Problem der Dimension 1024000

Vergleich der Peer-AMF-Methoden: PeerAMF3 liefert bei diesem Beispiel durchweg die schlechtesten Ergebnisse, für feine Toleranzen benötigt es sogar die dreifache Rechenzeit. PeerAMF4 ist für fast alle Toleranzen am besten geeignet, nur für $10^{-2.5}$ und 10^{-3} ist der Fehler schlechter als bei PeerAMF5. Bei PeerAMF5 stagniert der Fehler für die Toleranzen 10^{-2} bis 10^{-4} und erreicht danach bis 10^{-8} nur eine Genauigkeit von $10^{-6.5}$.

Vergleich von Krylov-, AMF-Verfahren und Ode15s: Auch hier lieferten alle Integratoren für alle Toleranzen Ergebnisse. Wir haben allerdings für den Gesamtvergleich zwei Diagramme erstellt, da Ode15s für dieses Beispiel relativ schlecht geeignet ist und sehr lange Rechenzeiten benötigt. Es ist durchgängig mehr als 100 mal langsamer als der Rest der Verfahren. Dies ist auf die Lösung der linearen Gleichungssysteme per LU-Zerlegung zurückzuführen, da die Bandbreite der Jacobi-Matrix ohne Umformungen $2 \cdot M^3$ beträgt und das Band aufgrund der Dreidimensionalität des Beispiels mit mehr Elementen besetzt ist als bei einem 2D-Problem. Beim Vergleich der restlichen Methoden sind die AMF-Verfahren wieder für grobe Toleranzen am besten geeignet. Allerdings hat RadauAMF für die Toleranzen 10^{-2} bis 10^{-3} einige Schwierigkeiten und

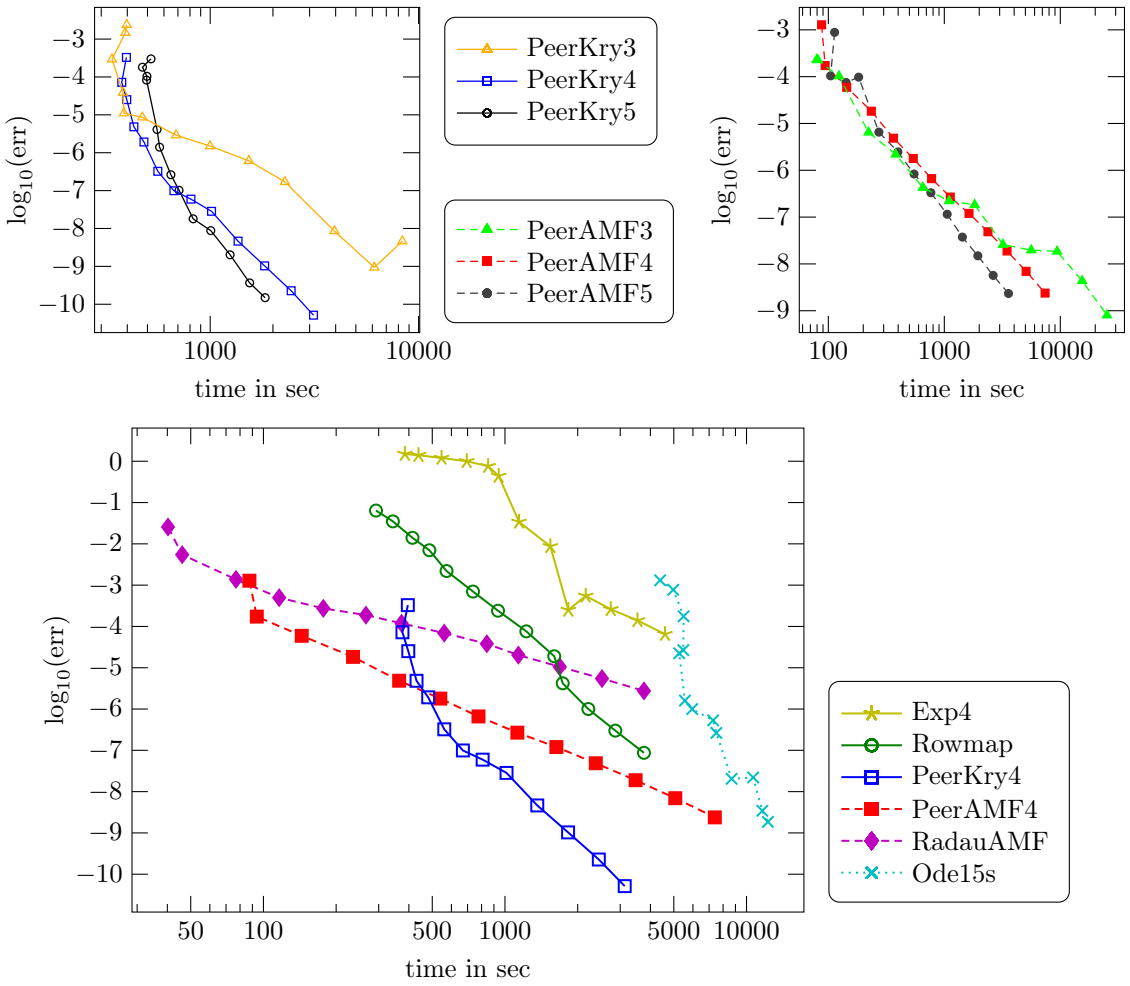


Abbildung 5.6: Ergebnisse für das zweidimensionale Radiation-Diffusion Problem der Dimension 80000 und dem Parameter $Z_0 = 1$

erzielt für 10^{-2} sogar das schlechteste Ergebnis. Für die Toleranzen 10^{-4} bis 10^{-6} rechnet es allerdings am schnellsten, ist jedoch danach schlechter als PeerKry4, was für feine Toleranzen die geringste Rechenzeit benötigt. PeerKry4 ist für die Toleranzen 10^{-6} bis 10^{-8} am besten geeignet und ist teilweise doppelt so schnell wie PeerAMF4. PeerAMF4 ist für 10^{-4} besser als RadauAMF, von 10^{-3} bis 10^{-6} ist es jedoch schlechter und danach wieder etwas besser. EXP4 erzielt für alle Toleranzen bei gleicher Rechenzeit die geringste Genauigkeit. ROWMAP ist für 10^{-2} am schnellsten und ist bis 10^{-5} besser als PeerKry4, was bei gleicher Genauigkeit etwas mehr Zeit benötigt. Ab 10^{-6} ist ROWMAP allerdings deutlich schlechter als PeerKry4, außerdem ist es bei gleicher Rechenzeit für alle Toleranzen ca. zwei Zehnerpotenzen schlechter in der Genauigkeit als die AMF-Methoden.

Ergebnisse für Combustion mit $M = 80$

Vergleich der Krylov-Peer-Verfahren: Wir erhalten hier fast genau die gleichen Ergebnisse wie für die kleinere Variante dieses Beispiels, nur dass die Verfahren ca. die

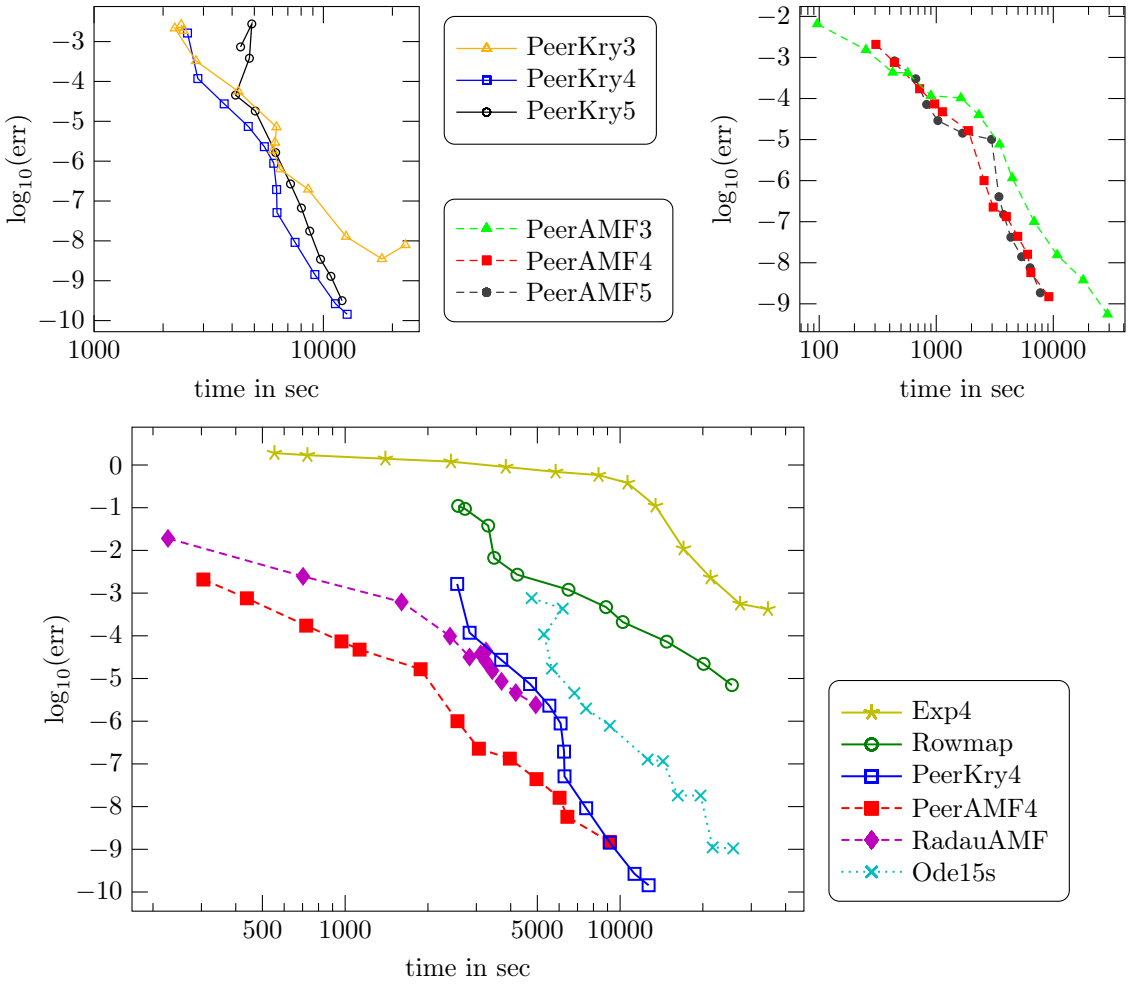


Abbildung 5.7: Ergebnisse für das zweidimensionale Radiation-Diffusion Problem der Dimension 80000 und dem Parameter $Z_0 = 10$

zehnfache Rechenzeit benötigen. PeerKry3 ist für 10^{-2} und $10^{-2.5}$ am besten geeignet, danach ist es etwas schlechter als PeerKry4 und für 10^{-4} bis $10^{-5.5}$ wieder besser. Bei 10^{-6} wird der Fehler bei diesem Verfahren etwas schlechter und danach nur kaum besser. PeerKry4 ist ab $10^{-5.5}$ bis 10^{-8} am besten geeignet und erreicht die höchste Genauigkeit. PeerKry5 rechnet für grobe Toleranzen am längsten und ist ab $10^{-5.5}$ besser als PeerKry3, aber benötigt bei gleichem Fehler doppelt so viel Rechenzeit wie PeerKry4.

Vergleich der Peer-AMF-Methoden: Auch hier sind die Ergebnisse zu Combustion $M = 40$ sehr ähnlich und die Verfahren rechnen zehn mal so lange. PeerAMF3 ist wieder für alle Toleranzen am schlechtesten. PeerAMF5 ist nur für 10^{-2} und $10^{-2.5}$ am besten geeignet, danach liefert PeerAMF4 die besten Ergebnisse.

Vergleich von Krylov-, AMF-Verfahren und Ode15s: Für dieses Problem konnte mit Ode15s kein Ergebnis erzielt werden, das liegt ganz einfach an der Dimension des Systems gewöhnlicher Differentialgleichungen von 1024000. Eine LU-Zerlegung benötigt zu viel Speicherplatz und ließ sich bei einem Test nicht einmal auf einem Rechner

mit 64GB Arbeitsspeicher realisieren. Alle anderen Methoden hatten aber bei allen Toleranzen keine Probleme. Insgesamt erhalten wir bei diesem Beispiel sehr ähnliche Ergebnisse wie für die kleinere Variante, nur dass die Unterschiede zwischen den Verfahren etwas größer sind. Die AMF-Verfahren sind für grobe Toleranzen deutlich schneller oder genauer als der Rest der Verfahren. RadauAMF hat für die Toleranzen 10^{-2} bis 10^{-3} ein paar Schwierigkeiten, liefert für $10^{-3.5}$ bis $10^{-6.5}$ aber die besten Ergebnisse. Es ist in diesem Bereich teilweise mehr als eine Zehnerpotenz genauer als PeerAMF4 bei gleicher Rechenzeit und ca. neunmal so schnell wie Krylov-Methoden. Für 10^{-7} bis 10^{-8} ist jedoch PeerAMF4 etwas besser geeignet und liefert den kleineren Fehler. PeerKry4 ist für grobe Toleranzen schlechter als die AMF-Methoden und für 10^{-2} bis $10^{-3.5}$ sogar weniger gut als ROWMAP. Für die Toleranzen 10^{-6} bis 10^{-8} ist es allerdings am schnellsten bei gleicher Genauigkeit, es benötigt teilweise mehr als 500s weniger als PeerAMF4. EXP4 ist für fast alle Toleranzen am schlechtesten geeignet und erzielt bei gleicher Rechenzeit die geringste Genauigkeit. Nur für $10^{-7.5}$ und 10^{-8} ist es etwas besser als ROWMAP.

Ergebnisse für Radiation mit $Z_0 = 1$

Vergleich der Krylov-Peer-Verfahren: PeerKry3 ist für die Toleranzen 10^{-2} bis 10^{-4} am besten geeignet, danach knickt das Verfahren ein und der Fehler wird nur sehr langsam besser. Ab 10^{-4} liefert PeerKry4 die besten Ergebnisse. PeerKry5 erzielt für grobe Toleranzen die schlechtesten Ergebnisse, rechnet aber ab 10^{-6} am schnellsten, es ist ca. doppelt so schnell wie PeerKry4 und mehr als fünfmal so schnell wie PeerKry3. PeerKry4 erreicht für 10^{-8} die höchste Genauigkeit.

Vergleich der Peer-AMF-Methoden: PeerAMF3 liefert für grobe Toleranzen die besten Ergebnisse, nur für $10^{-2.5}$ ist es schlechter als PeerAMF5. PeerAMF5 ist ab 10^{-6} am besten geeignet und rechnet mehr als doppelt so schnell wie PeerAMF3 und PeerAMF4. Für $10^{-7.5}$ und 10^{-8} ist PeerAMF5 sogar mehr als fünfmal schneller als PeerAMF3. PeerAMF3 erreicht die höchste Genauigkeit.

Vergleich von Krylov-, AMF-Verfahren und Ode15s: Bei diesem Beispiel rechneten alle Integratoren für alle Toleranzen erfolgreich. Auch hier sind die AMF-Methoden für grobe Toleranzen wieder am besten geeignet. RadauAMF ist für 10^{-2} bis 10^{-3} am schnellsten, in diesem Bereich ist es zehnmal so schnell wie PeerKry4 und 20-mal so schnell wie ROWMAP. Danach liefert PeerAMF4 bei gleicher Rechenzeit das genauere Ergebnis. Ab $10^{-2.5}$ ist der Fehler von PeerAMF4 gegenüber RadauAMF ca. eine Zehnerpotenz besser und für feine Toleranzen sogar mehr als zwei Zehnerpotenzen. Ab 10^{-4} liefert PeerKry4 die besten Resultate, erst ist es bei gleicher Rechenzeit rund eine Zehnerpotenz besser als PeerAMF4 und am Ende sogar mehr als zwei Zehnerpotenzen genauer. ROWMAP ist deutlich besser geeignet als EXP4 und Ode15s, es liefert bei gleicher Rechenzeit einem um ca. zwei Zehnerpotenzen besseren Fehler als EXP4. Außerdem rechnet es bei gleichem Fehler ca. fünfmal so schnell wie Ode15s. Ab $10^{-6.5}$ ist es sogar genauer als RadauAMF, jedoch ist es für alle Toleranzen um zwei Zehnerpotenzen schlechter in der Genauigkeit als PeerKry4. Ode15s benötigt insgesamt die längsten Rechenzeiten, für grobe Toleranzen ist es 100-mal langsamer als die AMF-Methoden. Für feine Toleranzen ist PeerKry4 mehr als zehnmal schneller als Ode15s.

Ergebnisse für Radiation mit $Z_0 = 10$

Vergleich der Krylov-Peer-Verfahren: PeerKry4 und PeerKry5 können für 10^{-2} kein Ergebnis erzielen. PeerKry4 ist ab 10^{-3} für alle Toleranzen am besten geeignet. PeerKry3 rechnet für 10^{-2} am schnellsten. PeerKry5 ist für grobe Toleranzen am schlechtesten geeignet, erst ab einer Toleranz von $10^{-5.5}$ ist es besser als PeerKry3. PeerKry3 knickt ab einer Toleranz von $10^{-6.5}$ ein und wird danach nur langsam genauer, für 10^{-8} wird der Fehler sogar wieder größer.

Vergleich der Peer-AMF-Methoden: Nur PeerAMF5 kann für 10^{-2} kein Ergebnis erzielen. PeerAMF3 ist für grobe Toleranzen am besten geeignet, ab einer Toleranz von $10^{-4.5}$ rechnet es allerdings am längsten. PeerAMF4 und PeerAMF5 liefern sehr ähnliche Ergebnisse, mal ist PeerAMF5 besser, danach wieder PeerAMF4. Für feine Toleranzen liegen die Kurven direkt übereinander. Beide Verfahren rechnen in diesem Bereich mehr als doppelt so schnell wie PeerAMF3.

Vergleich von Krylov-, AMF-Verfahren und Ode15s: Dieses Beispiel ist das anspruchsvollste und es werden somit auch die längsten Rechenzeiten benötigt. Gleich drei Integratoren hatten Schwierigkeiten bei groben Toleranzen: PeerKry4, RadauAMF und ROWMAP konnten für 10^{-2} kein Ergebnis erzielen, ROWMAP rechnete zusätzlich auch für $10^{-2.5}$ nicht erfolgreich. Bei allen anderen Integratoren gab es für alle Toleranzen keine Probleme. Auch bei diesem Beispiel sind wieder die AMF-Methoden am besten geeignet, wobei PeerAMF4 bei gleichen Rechenzeiten um mehr als eine Zehnerpotenz besser in der Genauigkeit ist als RadauAMF. Bei den Krylov-Methoden ist PeerKry4 am besten geeignet und es erreicht bei feinen Toleranzen ähnliche Rechenzeiten wie die AMF-Methoden. Anders als bei der Variante mit $Z_0 = 1$ ist Ode15s hier besser geeignet als ROWMAP, es ist teilweise drei Zehnerpotenzen genauer als ROWMAP, es benötigt aber bei gleicher Genauigkeit mehr als doppelt soviel Zeit wie PeerKry4. ROWMAP ist aber auch hier wieder besser in der Genauigkeit als EXP4, es ist teilweise fast drei Zehnerpotenzen genauer. EXP4 rechnet speziell für grobe Toleranzen sehr ungenau und erreicht in diesem Bereich nur einen Fehler in der Größenordnung von 10^0.

5.2.3 Diskussion

Insgesamt kann man sagen, dass die AMF-Methoden für solche Probleme sehr effizient sind, bei denen die LU-Zerlegung der Teilmatrizen des AMF-Splittings der Jacobi-Matrix leicht berechnet werden kann. Dies war bei allen drei betrachten Beispielen der Fall. In unseren Tests konnten wir mit den beiden AMF-Methoden PeerAMF4 und RadauAMF, speziell in dem für MOL-Probleme interessanten niedrigen Genauigkeitsbereich, exzellente Ergebnisse erzielen. Die beiden Verfahren lieferten sehr ähnliche Ergebnisse, für Combustion war RadauAMF besser und für Radiation PeerAMF4. Jedoch erfordern diese Methoden eine genaue Kenntnis der Jacobi-Matrix und die einzelnen Matrizen der Zerlegung müssen selbst implementiert werden. Wenn man die linearen Gleichungssysteme möglichst effizient lösen möchte, muss man auch dies am besten selbst umsetzen.

Im Gegensatz dazu sind Krylov-Methoden für den Nutzer einfacher zu handhaben, sie benötigen nur die rechte Seite und verwenden die Jacobi-Matrix nicht explizit, sondern nur in Form von Matrix-Vektor-Produkten, welche einfach zu approximieren sind. Wenn man für ein Beispiel eine hohe Genauigkeit bezüglich der Zeit benötigt, kann man die Verwendung von Peer-Methoden mit Krylov empfehlen. Sie benötigen bei hohen Genauigkeit und damit sehr kleinen Schrittweiten nur sehr geringe Krylov-Dimensionen und eine geringe Anzahl von Newton-Schritte, dadurch sind sie in diesem Bereich sehr effizient. Bei geringen Genauigkeiten treten jedoch wegen der großen Schrittweiten hohe Krylov-Dimensionen und eine größere Anzahl von Newton-Schritten auf, was die Methoden ineffizient werden lässt. Unter den Krylov-Verfahren lieferten die Peer-Methoden die besten Ergebnisse. Die anderen Krylov-Methoden erfüllten teilweise die vorgegeben Genauigkeitsforderungen nur ungenügend. Hier zeigt sich deutlich der Vorteil der hohen Stufenordnung der Peer-Methoden. Im Gegensatz zu ROWMAP und EXP4 tritt keine Ordnungsreduktion auf.

Bemerkung 5.2 [Vorkonditionierung]
Um die Effizienz der Krylov-Peer-Verfahren weiter zu verbessern, kann man Vorkonditionierer verwenden. Auch wir haben dies versucht. Wir haben uns dabei für sogenannte Rechts-Vorkonditionierer entschieden, weil diese das Residuum im Arnoldi-Prozess nicht beeinflussen und es deshalb weiterhin ohne Zusatzaufwand nach Formel (3.15) *berechnet werden kann. Das Ziel der Vorkonditionierung ist die Verbesserung der Konvergenz des Krylov-Unterraum-Verfahrens FOM, d.h. eine Verringerung der Krylovdimensionen. Allerdings muss man dafür zusätzliche lineare Gleichungssysteme der Dimension des Differentialgleichungssystems lösen. Des Weiteren treten relativ viele solcher Gleichungssysteme auf, im Newton-Schritt je eins pro Krylovdimension und ein zusätzliches. Das bedeutet, der Vorkonditionierer sollte eine einfache Struktur haben, damit der Zusatzaufwand gering bleibt. Allerdings muss er so komplex sein, damit noch eine Verbesserung der Konvergenz von FOM erreicht wird.*

Wir haben uns für zwei Typen von Vorkonditionierern entschieden, zum einen verwendeten wir die AMF-Matrizen und zum anderen sogenannte (2×2)*-Block-Diagonal Matrizen, welche die Hauptdiagonale der Jacobi-Matrix und den Reaktionsteil enthalten. Die zweite Variante haben wir aus [19] entnommen, wo sie schon für VODPK verwendet wurde. Wir mussten feststellen, dass eine Steigerung der Effizienz mit Hilfe der Vorkonditionierung nicht so ohne weiteres möglich ist. Es macht nur dann Sinn, wenn ohne Vorkonditionierung relativ hohe Krylovdimensionen und eventuell zusätzlich eine hohe Anzahl von Newton-Schritten auftreten.*

Zum anderen hat sich herausgestellt, dass die Wirksamkeit der Vorkonditionierung vom Prädiktor im Newton-Verfahren abhängt. Wir verwenden einen Prädiktor der Form (3.23) *mit der hohen Ordnung* $s - 1$*. In den von uns durchgeführten Tests an den Beispielen aus Abschnitt 5.2.1hat sich gezeigt, dass dieser Prädiktor schon ohne Vorkonditionierung sehr gute Ergebnisse liefert. Diese können mit Hilfe der aufgeführten Vorkonditionierer nur selten verbessert werden, weil der Zusatzaufwand zu groß und der Nutzen zu klein ist. Der Test von Startwerten mit einer geringeren Ordnung führte zu dem Ergebnis, dass diese zwar ohne Vorkonditionierung schlechtere Ergebnisse liefern als der Startwert hoher Ordnung, wenn man hier allerdings die beiden Vorkonditionierer anwendet, können die Ergebnisse meist erheblich verbessert werden. Jedoch*

sind die Ergebnisse mit Vorkonditionierung meist nicht besser als die Ergebnisse mit dem Startwert hoher Ordnung ohne Vorkonditionierung.

Fazit

Die Vorkonditionierung ist abhängig vom Beispiel und vom Prädiktor im Newton-Verfahren. Es ist oft schwierig einen geeigneten Vorkonditionierer zu finden. Der Startwert hoher Ordnung liefert bereits ohne Vorkonditionierung sehr gute Ergebnisse. Nur im Fall hoher Krylovdimensionen und einer hohen Anzahl von Newton-Schritten ist eine Verbesserung der Ergebnisse durch Vorkonditionierung möglich.

5.2.4 Test der FSAL-Methoden

In diesem Abschnitt wollen wir die FSAL-Peer-Methoden an den in Abschnitt 5.2.1 beschriebenen großen steifen Systemen testen. An den Beispielen geringer Dimension wurden sie bereits in [41] getestet und mit den Krylov-Peer-Verfahren verglichen. Wir verwenden dazu die in Abschnitt 2.4 für $s = 3$ und 4 konstruierten Verfahren. Bei den dreistufigen haben wir die Methode FSAL3a ausgewählt und bei den vierstufigen die Methode FSAL4. Diese beiden Verfahren stellen wir den drei- und vierstufigen Krylov-Peer- und Peer-AMF-Verfahren gegenüber. Die Implementierung der FSAL-Methoden mit Krylov und AMF haben wir wie in Abschitt 3.3 bzw. 4.3 beschrieben vorgenommen. Wir verwenden für Krylov und AMF jeweils die gleichen Sätze von Koeffizienten (FSAL3a, FSAL4) und bezeichnen die resultierenden Verfahren mit

$$\text{FSALKry}s, \qquad \text{FSALAMF}s, \qquad s = 3, 4.$$

Die absolute (*atol*) und relative (*rtol*) Fehler-Toleranz haben wir wieder auf $atol = rtol = tol$ gesetzt. Wir verwenden die Toleranzen

$$tol = 10^{-2}, 10^{-2.5}, \ldots, 10^{-8}.$$

Die bei den Test-Rechnungen erzielten Ergebnisse haben wir in den Abbildung 5.8 und 5.9 grafisch dargestellt. Wir vergleichen wieder den Fehler der numerischen Lösung zum Endzeitpunkt des Integrationsintervalls, welcher sich nach Formel (5.4) berechnet, mit der benötigten Rechenzeit. Wir verwenden erneut die doppelt-logarithmische Darstellung.

Diskussion der Ergebnisse

Der Vorteil der FSAL-Verfahren ist die Möglichkeit mit Hilfe von Interpolationspolynomen (Hermite-Interpolation) eine global stetig differenzierbare Lösung berechnen zu können, da man die Interpolationspolynome zweier aufeinanderfolgender Integrationsintervalle stetig differenzierbar verknüpfen kann. Der Vergleich mit den Peer-Verfahren ohne FSAL-Eigenschaft zeigt, dass man mit den FSAL-Peer-Methoden relativ ähnliche Ergebnisse erhält. Bei fast allen Beispielen liegen die Kurven für beide Verfahrenstypen dicht beieinander, sowohl bei den Krylov- als auch bei den AMF-Methoden. Bei Version eins des Brusselator Beispiels gibt es nur zwischen PeerKry3 und FSALKry3a

ein paar Unterschiede, für grobe Toleranzen ist FSALKry3a besser, für mittlere Toleranzen rechnet PeerKry3 etwas schneller und für feine Toleranzen erhält man wieder mit FSALKry3a die genauere Lösung. Bei der zweiten Version des Brusselators gibt es nur bei den dreistufigen Verfahren größere Unterschiede. FSALAMF3a ist für mittlere und feine Toleranzen deutlich genauer als PeerAMF3. Bei den Krylov-Methoden benötigt FSALKry3a für grobe Toleranzen die geringeren Rechenzeiten und für feine Toleranzen liefert PeerKry3 die bessere Genauigkeit. Bei den zwei Combustion Beispielen schneiden für feine Toleranzen die Krylov-FSAL-Methoden deutlich schlechter ab als die Krylov-Peer-Verfahren ohne FSAL-Eigenschaft. Mit PeerKry3 und PeerKry4 erhält man für diese Beispiele im Bereich feiner Toleranzen eine jeweils um eine Zehnerpotenz genauere Lösung als mit FSALKry3a und FSALKry4. Bei den AMF-Methoden liegen nur die Kurven der vierstufigen Verfahren etwas weiter auseinander. Bei beiden Beispielen erhält man für alle Toleranzen mit PeerAMF4 bessere Ergebnisse als mit FSALAMF4. Die Ergebnisse der beiden Radiation Beispiele ergeben, dass die Kurven der FSAL-Verfahren und die der normalen Peer-Verfahren alle dicht beieinander liegen und kaum Unterschiede erkennbar sind.

Fazit

Zusammenfassend kann man feststellen, dass die hier getesteten FSAL-Methoden für die betrachteten Beispiele nicht effizienter als die Peer-Methoden ohne FSAL-Eigenschaft sind. Sie erlauben jedoch mit Hilfe der Hermite-Interpolation die Berechnung einer stetig differenzierbaren Ausgabe.

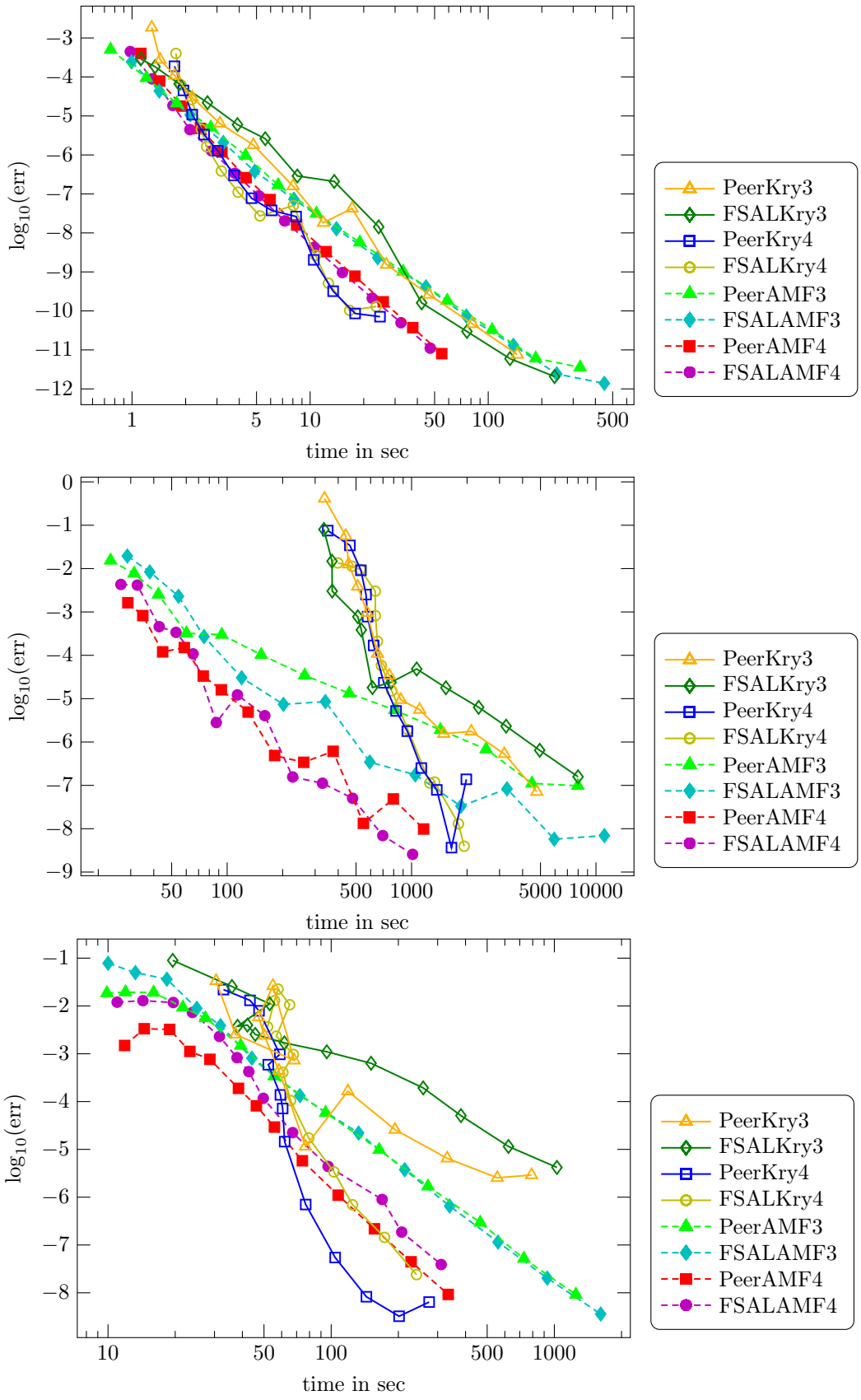


Abbildung 5.8: Ergebnisse für Brusselator V1 (oben), Brusselator V2 (Mitte) und Combustion $M = 40$ (unten)

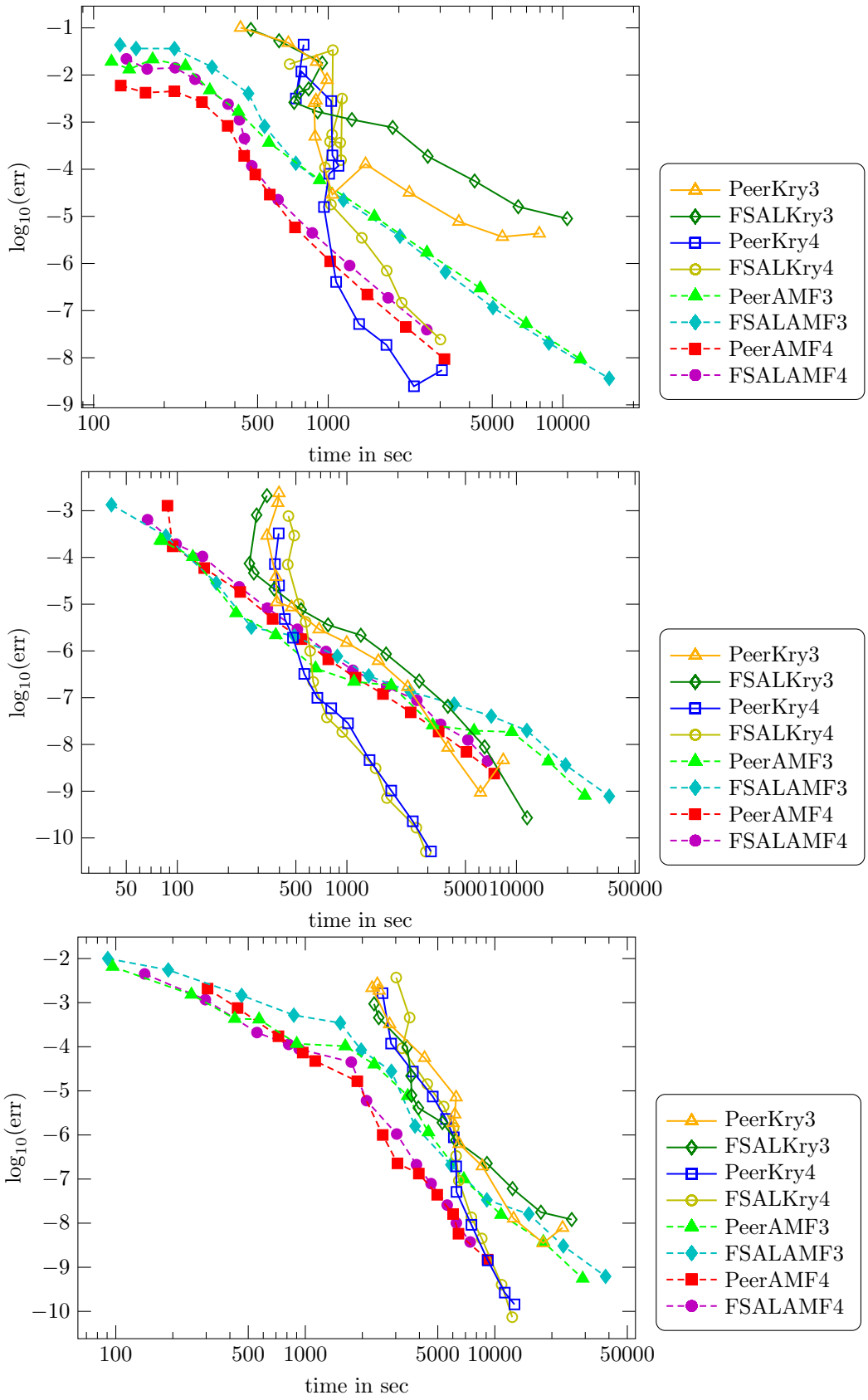


Abbildung 5.9: Ergebnisse für Combustion $M = 80$ (oben), Radiation $M = 200, Z0 = 1$ (Mitte) und Radiation $M = 200$, $Z0 = 10$ (unten)

6 Zusammenfassung und Ausblick

In dieser Arbeit wurden drei-, vier-, und fünfstufige optimal nullstabile superkonvergente implizite Peer-Verfahren konstruiert und getestet. Im Vordergrund stand dabei die Anwendung dieser Methoden auf große steife Differentialgleichungssysteme. Die größten Schwierigkeiten, die in diesem Zusammenhang auftreten, sind die sehr aufwendige Lösung der linearen Gleichungssysteme im Newton-Verfahren und die speicherplatz- und rechenintensive Berechnung der Jacobi-Matrix. Zur Verringerung des Rechenaufwandes wurden zwei Ansätze diskutiert: Zum einen die Kombination der Peer-Verfahren mit dem Krylov-Solver FOM und zum anderen die Verwendung der Approximierenden Matrix-Faktorisierung (AMF).

Die konstruierten Methoden wurden in Matlab implementiert und getestet. Die numerischen Tests und der Vergleich mit bereits existierenden Verfahren zeigten, dass sowohl die Krylov-Peer- als auch die Peer-AMF-Verfahren konkurrenzfähig sind. Unter den verglichenen Krylov-Methoden schnitten die Krylov-Peer-Verfahren am besten ab und im Vergleich mit RadauAMF lieferten die Peer-AMF-Methoden ähnliche Ergebnisse.

Weiterhin haben wir in dieser Arbeit drei- und vierstufige implizite Peer-Verfahren untersucht, welche die sogenannte FSAL-Eigenschaft besitzen. Auch diese Methoden haben wir mit FOM und AMF kombiniert und an großen steifen Systemen getestet. Es zeigte sich, dass diese Verfahren ähnliche Ergebnisse liefern wie die Peer-Methoden ohne FSAL-Eigenschaft, aber den Vorteil haben, dass man mit Hilfe von Interpolationspolynomen (Hermite-Interpolation) eine global stetig differenzierbare Lösung berechnen kann.

In der durchgeführten Verfahrenssuche wurden verschiedene Kriterien optimiert. Maximierung des Winkels der $L(\alpha)$-Stabilität, Minimierung der Fehlerkonstante err gegeben durch (3.16) und Minimierung des Parameters γ. Mit Hilfe des in Abschnitt 3.2 beschriebenen Evolutionsalgorithmus haben wir für die Stufenzahlen $s = 3, 4$ und 5 jeweils eine Menge von Verfahren erzeugt, welche die genannten Eigenschaften möglichst gut erfüllen. Das bedeutet, es können nicht alle Kriterien gleichzeitig erfüllt werden, sondern man muss einen guten Kompromiss finden. Diese gefundenen Methoden wurden an ausgewählten Beispielen getestet und jene Verfahren bestimmt, welche die besten Ergebnisse lieferten. Im Nachhinein hat sich gezeigt, dass die für $s = 4$ und 5 ausgewählten Methoden, welche in Tabelle 3.1 aufgeführten sind, genau die Verfahren unter den getesteten sind, bei denen die Zeilensummennorm der Matrix B für $\sigma = 1$, $\|B\|_\infty$ am kleinsten ist. In einer zukünftigen Verfahrenssuche könnte man diese Eigenschaft speziell berücksichtigen und so eventuell noch bessere Methoden finden.

In zukünftigen Betrachtungen könnten implizite Peer-Verfahren des Typs

$$Y_{m,i} = \sum_{j=1}^{s} b_{ij} Y_{m-1,j} + h_m \sum_{j=1}^{s} a_{ij} F_{m-1,j} + h_m \sum_{j=1}^{i} g_{ij} F_{m,j}, \quad i = 1, 2, \ldots, s, \tag{6.1}$$

welche zusätzlich die Funktionswerte $F_{m-1,j} = f(t_{m-1} + c_j h_{m-1}, Y_{m-1,j})$ des vergangenen Zeitschritts verwenden, untersucht werden. Diese Methoden versprechen eine höhere Konsistenzordnung, sind aber leider nicht mehr automatisch L-stabil, da $M(\infty) \neq 0$ ist. Es ist zu untersuchen, wie sich dies in den Tests auswirkt. Explizite Peer-Verfahren dieses Typs sind bereits sehr gut untersucht. Sie haben zum einen Konsistenzordnung $p = s$ und Konvergenzordnung $p = s + 1$ für variable Schrittweiten [53] und zum anderen Konsistenz- und Konvergenzordnung $p = 2s - 1$ für konstante Schrittweiten [11].

Literaturverzeichnis

[1] Beam, R.M. und R.F. Warming: *An implicit finite-difference algorithm for hyperbolic systems in conservation-law form.* J. Comput. Phys., 22:87–110, 1976.

[2] Beck, S.: *Implementierung eines Krylov-Solvers für große steife Systeme in Matlab.* Diplomarbeit, Universität Halle, 2008.

[3] Beck, S., S. González-Pinto, S. Pérez-Rodríguez und R. Weiner: *A comparison of AMF- and Krylov-methods in Matlab for large stiff ODE systems.* Journal of Computational and Applied Mathematics, 262:292–303, 2014.

[4] Beck, S., R. Weiner, B.A. Schmitt und H. Podhaisky: *Implicit peer methods for large stiff ODE systems.* J. Appl. Math. Comput., 38(1-2):389–406, 2012.

[5] Bornemann, F., D. Laurie, S. Wagon und J. Waldvogel: *Vom Lösen numerischer Probleme.* Springer, 2006.

[6] Brown, P.N., D.G. Byrne und A.C. Hindmarsh: *VODE: A variable-coefficient ODE solver.* SIAM J. Sci. St, 10(5):1038–1051, 1989.

[7] Butcher, J.C.: *On the implementation of implicit Runge-Kutta methods.* BIT, 16:237–240, 1976.

[8] Butcher, J.C.: *Order and effective order.* Applied Numerical Mathematics, 28:179–191, 1998.

[9] Butcher, J.C.: *Numerical Methods for Ordinary Differential Equations.* Wiley, 2008.

[10] Byrne, G.D.: *Pragmatic experiments with Krylov methods in the stiff ODE setting.* Computational Ordinary Differential Equations (Clarendon Press, Oxford), Seiten 323–356, 1992.

[11] Calvo, M., J.I. Montijano, L. Rández und M. Van Daele: *On the derivation of explicit two-step peer methods.* Applied Numerical Mathematics, 61:395–409, 2011.

[12] Dormand, J.R. und P.J. Prince: *A family of embedded Runge-Kutta formulae.* Journal of Computational and Applied Mathematics, 6:19–26, 1980.

[13] D'Yakonov, E.G.: *Difference systems of second order accuracy with a divided operator for parabolic equations without mixed derivatives.* USSR Comput. Math. Phys., 4(5):206–216, 1964.

[14] Erdmann, B., J. Lang und R. Roitzsch: *Kardos User´s Guide.* Technischer Bericht ZR 02-42, Konrad-Zuse-Zentrum Berlin, 2002.

[15] Freund, R.W. und R.H.W. Hoppe: *Stoer/Bulirsch: Numerische Mathematik 1.* Springer, 10. Auflage, 2007.

[16] Gerisch, A., J. Lang, H. Podhaisky und R. Weiner: *High-order linearly implicit two-step peer - finite element methods for time-dependent PDEs.* Applied Numerical Mathematics, 59:624–638, 2009.

[17] Giraud, L., J. Langou und M. Rozložník: *On the round-off error analysis of the Gram-Schmidt algorithm with reorthogonalization.* Technischer Bericht TR/PA/02/33, CERFACS, 2002.

[18] Giraud, L., J. Langou und M. Rozložník: *Robust selective Gram-Schmidt reorthogonalization.* Technischer Bericht TR/PA/02/52, CERFACS, 2002.

[19] González-Pinto, S. und S. Pérez-Rodríguez: *A variable time-step-size code for advection-diffusion-reaction PDEs.* Applied Numerical Mathematics, 62:1447–1462, 2012.

[20] Hairer, E., S.P. Nørsett und G. Wanner: *Solving Ordinary Differential Equations I.* Springer, 2. Auflage, 1993.

[21] Hairer, E. und G. Wanner: *Solving Ordinary Differential Equations II.* Springer, 2. Auflage, 1996.

[22] Hochbruck, M., Ch. Lubich und H. Selhofer: *Exponential integrators for large systems of differential equations.* SIAM J. Sci. Comput., 5:1552–1574, 1998.

[23] Houwen, P.J. van der und B.P. Sommeijer: *Approximate factorization for time-dependent partial differential equations.* J. Comput Appl. Math., 128:447–466, 2001.

[24] Hundsdorfer, W. und J.G. Verwer: *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations.* Springer, 2003.

[25] Jackiewicz, Z.: *General Linear Methods for Ordinary Differential Equations.* Wiley, 2009.

[26] Kaps, P. und P. Rentrop: *Generalized Runge-Kutta methods of order four with stepsize control for stiff ordinary differential equations.* Numer. Math., 38:55–68, 1979.

[27] Keller, H.B.: *Numerical Solution of Bifurcation and Nonlinear Eigenvalue Problems.* Publ. Math. Res. Center, 38:359–384, Academic Press, New York, 1977.

[28] Kulikov, G.Yu. und R. Weiner: *Variable-stepsize interpolating explicit parallel peer methods with inherent global error control.* SIAM Journal on Scientific Computing, 32:1695–1723, 2010.

[29] Lioen, W.M. und J.J.B. de Swart: *Test Set for Initial Value Problem Solvers.* Technischer Bericht Report MAS-R9832, CWI Amsterdam, 1998.

[30] Meister, A.: *Numerik linearer Gleichungssysteme.* Vieweg, 1999.

[31] Mousseau, V.A., D.A. Knoll und W.J. Rider: *Physics based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion.* J. Comput. Phys., 160:743–765, 2000.

[32] Parlett, B.N.: *The Symmetric Eigenvalue Problem.* Englewood Cliffs, N.J., Prentice-Hall, 1980.

[33] Peaceman, D.W. und H.H. Rachford: *The numerical solution of parabolic and elliptic differential equations.* J. Soc. Indust. Appl. Math, 3:28–41, 1955.

[34] Pérez-Rodríguez, S., S. González-Pinto und B.P. Sommeijer: *An iterated Radau method for time-dependent PDEs.* J. Comput. Appl. Math., 231:49–66, 2009.

[35] Podhaisky, H., R. Weiner und B.A. Schmitt: *Rosenbrock-type 'Peer' two-step methods.* Applied Numerical Mathematics, 53:409–420, 2005.

[36] Podhaisky, H., R. Weiner und B.A. Schmitt: *Linearly-implicit two-step methods and their implementation in Nordsieck form.* Applied Numerical Mathematics, 56:374–387, 2006.

[37] Saad, Y.: *Iterative Methods for Sparse linear systems.* PWS Publishing Company, 1996.

[38] Saad, Y. und M.H. Schultz: *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems.* SIAM J. Sci. Stat. Comput., 7:856–869, 1986.

[39] Schmitt, B.A. und R. Weiner: *Matrix-free W-methods using a multiple Arnoldi iteration.* APNUM, 18:307–320, 1995.

[40] Schmitt, B.A. und R. Weiner: *Parallel two-step W-methods with peer variables.* SIAM J. Numer. Anal., 42:265–282, 2004.

[41] Schmitt, B.A., R. Weiner und S. Beck: *Two-step peer methods with continuous output.* BIT Numerical Mathematics, 53:717–739, 2013.

[42] Schmitt, B.A., R. Weiner und K. Erdmann: *Implicit parallel peer methods for stiff initial value problems.* Applied Numerical Mathematics, 53:457–470, 2005.

[43] Schmitt, B.A., R. Weiner und H. Podhaisky: *Multi-implicit peer two-step W-methods for parallel time integration.* BIT Numerical Mathematics, 45:197–217, 2005.

[44] Shampine, L.F. und M.W. Reichelt: *The MATLAB ODE Suite.* SIAM Journal on Scientific Computing, 18:1–22, 1997.

[45] Skeel, R.D.: *Analysis of fixed-stepsize methods.* SIAM Journal on Numerical Analysis, 13(5):664–685, 1976.

[46] Sommeijer, B.P., L.F. Shampine und J.G. Verwer: *RKC: An explicit solver for parabolic PDEs.* J. Comput. Appl. Math., 88:315–326, 1997.

[47] Strehmel, K., R. Weiner und H. Podhaisky: *Numerik gewöhnlicher Differentialgleichungen*. Springer Spektrum, 2012.

[48] Verwer, J.G., E.J. Spee, J.G. Bloom und W. Hundsdorfer: *A second order Rosenbrock method applied to photochemical dispersion problems*. SIAM J. Sci. Comput., 20:1456–1480, 1999.

[49] Weiner, R.: *Vorlesungsskript: Numerik großer steifer Systeme*. WS 2007/08.

[50] Weiner, R. und B.A. Schmitt: *Order Results for Krylov-W-methods*. Computing, 61:69–89, 1998.

[51] Weiner, R., B.A. Schmitt und H. Podhaisky: *ROWMAP - a ROW-code with Krylov techniques for large stiff ODEs*. Applied Numerical Mathematics, 25:303–319, 1997.

[52] Weiner, R., B.A. Schmitt und H. Podhaisky: *Parallel Peer two-step W-methods and their application to MOL-systems*. Applied Numerical Mathematics, 48:425–439, 2004.

[53] Weiner, R., B.A. Schmitt, H. Podhaisky und S. Jebens: *Superconvergent explicit two-step peer methods*. Journal of Computational and Applied Mathematics, 223(2):753–764, 2009.

Lebenslauf

Persönliche Daten

Name	Steffen Beck
Anschrift	Gustav-Adolf-Str. 31a 06686 Lützen
geboren	am 25.09.1982 in Hohenmölsen
Familienstand	ledig
Kinder	einen Jungen (5 Jahre)

Bildungsgang

1989 – 1993	Grundschule Muschwitz
1993 – 2002	Agricolagymnasium Hohenmölsen, Abschluss Abitur
2002 – 2003	Grundwehrdienst
2003 – 2008	Mathematikstudium mit Nebenfach Physik an der Martin–Luther–Universität Halle–Wittenberg, Abschluss Diplom–Mathematiker, Thema der Diplomarbeit: „Implementierung eines Krylov-Solvers für große steife Systeme in Matlab“ Betreuer: Prof. Dr. Rüdiger Weiner
2008 – 2014	Promotionsstudium am Institut für Mathematik der Martin–Luther–Universität Halle–Wittenberg

Berufliche Laufbahn

2008 – 2014	Wissenschaftlicher Mitarbeiter in der Arbeitsgruppe Numerik am Institut für Mathematik an der Martin–Luther–Universität Halle–Wittenberg

Publikationsliste

[1] Beck, S., S. González-Pinto, S. Pérez-Rodríguez und R. Weiner: *A comparison of AMF- and Krylov-methods in Matlab for large stiff ODE systems*. Journal of Computational and Applied Mathematics, 262:292–303, 2014

[2] Beck, S., R. Weiner, B.A. Schmitt und H. Podhaisky: *Implicit peer methods for large stiff ODE systems*. J. Appl. Math. Comput., 38(1-2):389–406, 2012.

[3] Schmitt, B.A., R. Weiner und S. Beck: *Two-step peer methods with continuous output*. BIT Numerical Mathematics, 53:717–739, 2013.

Eidestattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig, ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die aus anderen Werken wörtlich oder inhaltlich entnommenen Daten, Fakten und Konzepte sind unter Angabe der entsprechenden Quelle als solche gekennzeichnet.

Diese Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem anderen Prüfungsverfahren vorgelegt.

Halle (Saale), 26. Juni 2014

Steffen Beck