Numerische Verfahren zur Lösung parabolischer partieller Differentialgleichungen

Masterarbeit

von Jalo Liljo

Heinrich-Heine-Universität Düsseldorf

Lehrstuhl für Angewandte Mathematik

September 2006

Inhaltsverzeichnis

1 Einleitung

Das Ziel dieser Arbeit ist die Lösung parabolischer partieller Differentialgleichungen mit numerischen Verfahren. Hierzu werden verschiedene Verfahren zur Lösung großer Systeme gewöhnlicher Differentialgleichungen, welche bei der Ortsdiskretisierung parabolischer Differentialgleichungen auftreten, betrachtet.

Nach der Diskretisierung erhält man ein System von Anfangswertproblemen der Form

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0. \tag{1.1}$$

Eine spezielle Klasse sind semilineare parabolische Differentialgleichungen

$$y'(t) + Ay(t) = g(t, y(t)), \qquad y(t_0) = y_0, \tag{1.2}$$

mit einer konstanten Matrix A und nichtlinearen Termen die in g enthalten sind. A ist dabei positiv semidefinit oder sektoriell. Eine Matrix heißt sektoriell, wenn alle Eigenwerte in der komplexen Ebene betrachtet in einem Sektor $S = \{z \in \mathbb{C} : |arg(z)| \leq \alpha\}$ um die positive reelle Achse liegen, wobei $0 \leq \alpha < \frac{\pi}{2}$.

Wenn nicht anders angegeben, ist in dieser Arbeit vorausgesetzt, dass A symmetrisch und positiv semidefinit ist und, dass die Jacobi-Matrix $J = \frac{\partial f(t,y)}{\partial y}$ der Differentialgleichung (1.1) symmetrisch und negativ semidefinit ist.

Zunächst werden in Kapitel 2 einige exponentielle Integratoren und die wesentlichen Aspekte vorgestellt. Exponentielle Integratoren sind dadurch charakterisiert, dass Matrixfunktionen verwendet werden. Die Idee die Exponentialfunktion einer Matrix zu verwenden, geht in die 60er Jahre zurück und galt lange als nicht praktikabel. Neue Methoden und Verbesserungen in der Berechnung des Produkts einer Matrixexponentialfunktion mit einem Vektor haben das Interesse in den letzten Jahren wieder verstärkt.

In Kapitel 3 wird ein Verfahren vorgestellt, welches auf einer Familie von Runge-Kutta-Chebyshev-Formeln basiert und 1980 von Sommeijer und Verwer vorgestellt wurde [15]. 1997 wurde dann ein Fortran-Code veröffentlicht, [17], welcher im Rahmen dieser Arbeit in Matlab implementiert wurde. Es werden die wichtigsten Optionen und Aspekte des Matlab-Codes erläutert und in Kapitel 5 numerische Vergleiche mit anderen Verfahren aufgeführt.

Um das Produkt der Matrixexponentialfunktion mit einem Vektor zu berechnen, haben sich Krylov-Verfahren als effizient herausgestellt. Dies wollen wir in Kapitel 2 kurz motivieren und in Kapitel 4 ein vorkonditioniertes Lanczos-Verfahren vorstellen. Durch die Vorkonditionierung soll die Anzahl der Lanczos-Schritte reduziert werden, um die Berechnung zu beschleunigen. Auch hierzu werden Beispiele betrachtet.

In Kapitel 5 werden die Verfahren dann an einigen Beispielen getestet.

Besonderer Dank gilt Prof. Dr. M. Hochbruck und J. Niehoff für die hilfreichen Diskussionen.

2 Exponentielle Integratoren

In diesem Kapitel wird zunächst die Verwendung der Matrixexponentialfunktion in numerischen Verfahren zur Integration von Differentialgleichungssystemen und die Verwendung von Krylov-Verfahren motiviert. Anschließend werden einige exponentielle Integratoren behandelt. Zum einem der in [6] von Hochbruck, Lubich und Selhofer vorgestellte Integrator *exp*4 auf welchen wir dann in Abschnitt 4 wieder eingehen werden, zum anderen eine Klasse expliziter exponentieller Runge-Kutta-Verfahren.

2.1 Motivation

Betrachtet man die Differentialgleichung

$$y'(t) + Ay(t) = 0, \qquad y(0) = y_0,$$

so ist die Lösung durch $y(t) = \exp(-tA)y_0$ gegeben. Für das inhomogene Problem

$$y'(t) + Ay(t) = b, \qquad y(0) = y_0,$$

mit einer konstanten Matrix A und konstantem Vektor b ist die Lösung durch

$$y(t) = y_0 + t\varphi(-tA)(Ay_0 + b),$$

mit der analytischen Funktion

$$\varphi(\lambda) = (\exp(\lambda) - 1)/\lambda$$

gegeben. Daher liegt die Idee nahe, die Matrixexponentialfunktion in numerischen Integratoren zu verwenden.

Die Verwendung von Krylov-Verfahren ist folgendermaßen motiviert, wobei wesentlich ist, dass man $f(A)$ nicht explizit benötigt, sondern nur das Produkt mit einem Vektor v. Die Cauchy-Integralformel besagt, dass für eine in einem Gebiet $\Omega \subset \mathbb{C}$ analytische Funktion

$$f(a) = \frac{1}{2\pi i} \int_\Gamma f(\lambda)(\lambda - a)^{-1} d\lambda$$

gilt, wobei Γ eine einfach geschlossene Kurve in Ω und $a \in \mathbb{C}$ ein Punkt ist, der nicht auf Γ liegt und genau einmal von Γ in positiver Richtung umlaufen wird. Die Cauchy-Integralformel kann auch auf quadratische Matrizen verallgemeinert werden und man kann zeigen, dass

$$f(A) = \frac{1}{2\pi i} \int_\Gamma f(\lambda)(\lambda I - A)^{-1} d\lambda$$

gilt, wobei verlangt wird, dass die Kurve Γ den Wertebereich

$$\mathcal{F}(A) = \{x^H Ax : x \in \mathbb{C}^n, \|x\| = 1\}$$

im Inneren enthält.

Um $f(A)v$ zu approximieren wird ausgenutzt, dass für jedes $\lambda \in \Gamma$ in der Cauchy-Integralformel

$$f(A)v = \frac{1}{2\pi i}\int_\Gamma f(\lambda)(\lambda I - A)^{-1}v d\lambda = \frac{1}{2\pi i}\int_\Gamma f(\lambda)x(\lambda)d\lambda$$

die Lösung eines linearen Gleichungssytems auftritt. Das Gleichungssystem

$$(\lambda I - A)x(\lambda) = v \tag{2.1}$$

können wir näherungsweise mit einem Krylov-Verfahren lösen.

Zunächst konstruiert man eine Basis V_m des m-dimensionalen Krylovraumes. Ist V_m eine Basis von $\mathcal{K}_m(A, v)$, so gilt

$$AV_m = V_m H_m + \beta_m v_{m+1} e_m^T, \quad \text{mit} \quad V_m e_1 = v \quad \text{und} \quad V_m^H V_m = I,$$

wobei H_m eine Matrix in Hessenbergform ist und tridiagonal, falls A symmetrisch ist. Ist H_m tridiagonal, so schreiben wir im Folgenden T_m.

Aus $\mathcal{K}_m(A, v) = \mathcal{K}_m(\lambda I - A, v)$ für alle $\lambda \in \mathbb{C}$ folgt, dass V_m auch Basis der verschobenen Krylovräume ist. Also gilt auch

$$(\lambda I - A)V_m = V_m(\lambda I - H_m) - \beta_m v_{m+1} e_m^T$$

und die Galerkin-Iterierte zu (2.1) ist daher durch

$$x_m(\lambda) = V_m(\lambda I - H_m)^{-1}\|v\|e_1$$

gegeben. Dabei ist die Galerkin-Iterierte $x_m(\lambda) \in \mathcal{K}_m(A, v)$ durch $r_m(\lambda) \perp \mathcal{K}_m(A, v)$ definiert, wobei

$$r_m(\lambda) = v - (\lambda I - A)x_m(\lambda)$$

das Residuum bezeichnet. Sie existiert für jedes m, da Γ wegen $\mathcal{F}(H_m) \subset \mathcal{F}(A)$ den Wertebereich von H_m umschließt.

Setzt man dies in die Integraldarstellung ein, so erhält man:

$$\begin{aligned}
f(A)v &= \frac{1}{2\pi i}\int_\Gamma f(\lambda)(\lambda I - A)^{-1}v d\lambda \\
&\approx \frac{1}{2\pi i}\int_\Gamma f(\lambda)x_m(\lambda)d\lambda \\
&= \frac{1}{2\pi i}\int_\Gamma f(\lambda)V_m(\lambda I - H_m)^{-1}(\|v\|e_1)d\lambda \qquad (2.2)\\
&= V_m\frac{1}{2\pi i}\int_\Gamma f(\lambda)(\lambda I - H_m)^{-1}d\lambda(\|v\|e_1) \\
&= V_m f(H_m)(\|v\|e_1)
\end{aligned}$$

Die Matrix H_m hat kleinere Dimension und $f(H_m)$ kann man zum Beispiel durch Diagonalisierung oder Padé-Approximation berechnen.

Für die Funktion $\exp(\lambda)$, v mit $\|v\| = 1$ und eine Matrix A mit $\lambda(A) \in [-4\rho, 0]$, wobei $\lambda(A)$ die Eigenwerte von A bezeichnet, gilt beispielsweise, [5]:

$$\|e^{\tau A}v - V_m e^{\tau H_m} e_1\| \leq \begin{cases} 10e^{-m^2/(5\rho\tau)} & \sqrt{4\rho\tau} \leq m \leq 2\rho\tau \\ 10(\rho\tau)^{-1} e^{-\rho\tau}(\frac{e\rho\tau}{m})^m & m \geq 2\rho\tau. \end{cases}$$

In [5] findet man weitere Resultate für symmetrische und schief symmetrische Matrizen, sowie für Matrizen, deren Eigenwerte in einem Kreissektor liegen. Gallopolous und Saad zeigten, dass der Fehler der m-ten Iterierten proportional zu $\|\tau A\|^m/m!$ ist, was superlineare Konvergenz für $m \gg \|\tau A\|$ bedeutet. In [5] wird gezeigt, dass dies für negativ definite symmetrische Matrizen bereits für $m \geq \sqrt{\|\tau A\|}$ der Fall ist, wohingegen bei schief symmetrischen Matrizen mit gleichmäßig verteilten Eigenwerten m nah bei $\|\tau A\|$ liegt. Für eine gewisse Klasse von Matrizen, deren Eigenwerte in einem Kreissektor enthalten sind, wird eine schnelle Fehlerreduktion für $m \ll \|\tau A\|$ gezeigt. Weiterhin konvergieren Krylov-Verfahren für $\varphi(\tau A)v$ fast so schnell wie für $\exp(\tau A)v$, da der Fehler der Approximation an $\varphi(\tau A)v$ vom Fehler der Approximation an $\exp(\tau A)v$ abhängt:

$$\begin{aligned} \|\varphi(\tau A)v - V_m\varphi(\tau H_m)e_1\| &= \|\frac{1}{\tau}\int_0^\tau e^{\sigma A}v d\sigma - \frac{1}{\tau}\int_0^\tau V_m e^{\sigma H_m} e_1 d\sigma\| \\ &\leq \frac{1}{\tau}\int_0^\tau \|e^{\sigma A}v - V_m e^{\sigma H_m} e_1\| d\sigma \\ &\leq \max_{\sigma\in[0,\tau]} \|e^{\sigma A}v - V_m e^{\sigma H_m} e_1\|. \end{aligned}$$

2.2 Verfahren

Ein häufig verwendeter Ansatz zur Konstruktion exponentieller Verfahren, auf welchen wir später nochmals eingehen werden, ist die Variation-der-Konstanten-Formel, welche besagt, dass für die Lösung von (1.2)

$$y(t_n + h) = e^{-hA}y(t_n) + \int_0^h e^{-(h-\tau)A} g(t_n + \tau, y(t_n + \tau)) d\tau \qquad (2.3)$$

gilt. Hieraus ergibt sich sofort eine erste einfache Methode, die exponentielle Euler-Methode.

2.2.1 Exponentielle Euler-Methode

Aus (2.3) folgt

$$\begin{aligned} y(t_n+h) &= e^{-hA}y(t_n) + \int_0^h e^{-(h-\tau)A} g(t_n+\tau, y(t_n+\tau))d\tau \\ &= e^{-hA}y(t_n) + h\varphi(-hA)g(t_n, y(t_n)) + d_{n+1}, \end{aligned}$$

wobei

$$\varphi(-hA) = \frac{1}{h}\int_0^h e^{-(h-\tau)A}d\tau. \tag{2.4}$$

Mit $f(t_n+\tau) = g(t_n+\tau, y(t_n+\tau))$ gilt

$$y(t_n+h) = e^{-hA}y(t_n) + \int_0^h e^{-(h-\tau)A}\left(f(t_n) + \int_{t_n}^{t_n+\tau} f'(s)ds\right) \tag{2.5}$$

und daher

$$d_{n+1} = \int_0^h e^{-(h-\tau)A}\left(\int_{t_n}^{t_n+\tau} f'(s)ds\right)d\tau.$$

Woraus

$$\|d_{n+1}\| \le \int_0^h \|e^{-(h-\tau)A}\| \int_{t_n}^{t_n+\tau} \|f'(s)\|dsd\tau \le Ch\int_{t_n}^{t_n+h} \|f'(s)\|ds,$$

mit einer Konstanten C, folgt.

Durch Approximationen an $g(t_n+\tau, y(t_n+\tau))$ kann man unterschiedliche Methoden konstruieren, wie wir später sehen werden. Die triviale Approximation $g(t_n+\tau, y(t_n+\tau)) \approx g(t_n, y(t_n))$ liefert die exponentielle Euler-Methode

$$y_{n+1} = e^{-hA}y_n + h\varphi(-hA)g(t_n, y_n),$$

welche klassische Ordnung 2 hat und für lineare Probleme exakt ist.

Im nächsten Abschnitt betrachten wir das in [6] vorgestellte Verfahren *exp*4, welches zu einer in [5] vorgestellten Klasse von exponentiellen Integratoren gehört.

2.2.2 EXP4

In diesem Abschnitt betrachten wir eine Klasse von numerischen Verfahren zur Lösung von großen, steifen Systemen nichtlinearer Anfangswertprobleme in autonomer Form

$$y'(t) = f(y(t)), \qquad y(t_0) = y_0. \tag{2.6}$$

Die hier angeführten Verfahren verwenden Matrix-Vektor Multiplikationen $\varphi(\tau J)v$, wobei $J = \frac{\partial f}{\partial y}$ die Jacobi-Matrix von f bezeichnet und τ von der Schrittweite abhängt. Dabei ist φ die gleiche Funktion wie in (2.4)

$$\varphi(z) = \frac{e^z - 1}{z}. \tag{2.7}$$

Bei der ursprünglichen Form berechnet man ausgehend von einer Näherung y_0 an $y(t_0)$, $y_1 = y(t_0 + h)$ folgendermaßen:

$$\begin{aligned} k_i &= \varphi(\gamma h J)\left(f(u_i) + hJ\sum_{j=1}^{i-1}\gamma_{ij}k_j\right), \qquad i = 1, ..., s \\ u_i &= y_0 + h\sum_{j=1}^{i-1}\alpha_{ij}k_j \\ y_1 &= y_0 + h\sum_{i=1}^{s} b_i k_i \end{aligned} \tag{2.8}$$

Hierbei sind γ, γ_{ij}, α_{ij} und b_i Koeffizienten, die sich aus Ordnungsbedingungen ergeben, welche man in [6] findet. Für $i \leq j$ gilt $\gamma_{ij} = \alpha_{ij} = 0$. Setzt man $J = 0$ folgt $\varphi(z) \equiv 1$ und man hätte ein explizites Runge-Kutta-Verfahren, und mit der Wahl $\varphi(z) = 1/(1 - z)$ ein Rosenbrock-Wanner-Verfahren.

Bei einem Schritt nach dem numerischen Schema (2.8) müssen s Multiplikationen von $\varphi(\gamma h J)$ mit s unterschiedlichen Vektoren berechnet werden. Wenn diese Produkte mit Krylov-Methoden approximiert werden, müssen in jedem Schritt s Krylov-Räume konstruiert werden, was die Berechnung zu teuer macht. Durch geschickte Wahl der Koeffizienten kann man die Anzahl der Multiplikationen reduzieren. Grundlage hierfür ist, dass man $\varphi(jz)$, $j = 2, 3, ...$ rekursiv aus $\varphi(z)$ berechnen kann.

Es gilt:

$$
\begin{aligned}
\varphi(2z) &= \frac{e^{2z}-1}{2z} = \frac{1}{2}\frac{e^{2z}-2e^z+1}{z} + \frac{e^z-1}{z} = \left(\frac{1}{2}z\left(\frac{e^z-1}{z}\right)^2 + \frac{e^z-1}{z}\right) \\
&= (\frac{1}{2}z\varphi(z)+1)\varphi(z) \\
\varphi(3z) &= \frac{e^{3z}-2e^{2z}+e^z}{3z} + \frac{2e^{2z}-2e^z}{3z} + \frac{e^z-1}{3z} \\
&= \frac{2}{3}e^z\left(\frac{1}{2}\frac{e^{2z}-2e^z+1}{z} + \frac{e^z-1}{z}\right) + \frac{1}{3}\frac{e^z-1}{z} \\
&= \frac{2}{3}\left(z\frac{e^z-1}{z}+1\right)\left(\frac{1}{2}z\frac{e^z-1}{z}+1\right)\frac{e^z-1}{z} + \frac{1}{3}\frac{e^z-1}{z} \\
&= \frac{2}{3}(z\varphi(z)+1)\varphi(2z) + \frac{1}{3}\varphi(z) \\
\varphi(4z) &= ...
\end{aligned}
$$

In [6] wird $\gamma = 1/n$ gewählt und gezeigt, dass man durch geschickte Koeffizientenwahl die Anzahl der Multiplikationen um den Faktor n reduzieren kann. Mit $\gamma = 1/3$ wird hierzu folgender Ansatz verwendet.

Zunächst wählen wir $\alpha_{2,1} = \alpha_{3,1} = \alpha_{3,2} = 0$ woraus $u_1 = u_2 = u_3 = y_0$ folgt. Es gilt

$$
k_1 = \varphi(\frac{1}{3}hJ)f(u_1) \quad \text{und} \quad k_2 = \varphi(\frac{1}{3}hJ)\left(f(u_1) + hJ\gamma_{2,1}\varphi(\frac{1}{3}hJ)f(u_1)\right).
$$

Mit $\gamma_{2,1} = 1/6$ und der Rekursion folgt

$$
k_2 = \varphi(\frac{1}{3}hJ)\left(1 + \frac{1}{2}\frac{hJ}{3}\varphi(\frac{1}{3}hJ)\right)f(u_1) = \varphi(\frac{2}{3}hJ)f(u_1).
$$

Auf diese Weise erhält man mit $s = 7$ folgende Methode, die mit 3 f-Auswertungen in jedem Schritt auskommt:

$$
\begin{aligned}
k_1 &= \varphi(\tfrac{1}{3}hJ)f(y_0)\\
k_2 &= \varphi(\tfrac{2}{3}hJ)f(y_0)\\
k_3 &= \varphi(hJ)f(y_0)\\
w_4 &= -\tfrac{7}{300}k_1 + \tfrac{97}{150}k_2 - \tfrac{37}{300}k_3\\
u_4 &= y_0 + hw_4\\
d_4 &= f(u_4) - f(y_0) - hJw_4\\
k_4 &= \varphi(\tfrac{1}{3}hJ)d_4\\
k_5 &= \varphi(\tfrac{2}{3}hJ)d_4\\
k_6 &= \varphi(hJ)d_4\\
w_7 &= \tfrac{59}{300}k_1 - \tfrac{7}{75}k_2 + \tfrac{269}{300}k_3 + \tfrac{2}{3}(k_4 + k_5 + k_6)\\
u_7 &= y_0 + hw_7\\
d_7 &= f(u_7) - f(y_0) - hJw_7\\
k_7 &= \varphi(\tfrac{1}{3}hJ)d_7\\
y_1 &= y_0 + h(k_3 + k_4 - \tfrac{4}{3}k_5 + k_6 + \tfrac{1}{6}k_7)
\end{aligned}
\tag{2.9}
$$

Hierbei sind w_i und d_i Hilfsvektoren. Die Methode, die im Folgenden mit $exp4$ bezeichnet wird, hat klassische Ordnung 4 für Differentialgleichungen der Form (2.6) und ist exakt für lineare Differentialgleichungen. In [6] wird gezeigt, dass (2.8) auch auf differentiell-algebraische Gleichungen erweitert werden kann, und es wird eine Ordnungsbedingung für differentiell-algebraische Gleichungen angegeben, welche auch für steife Differentialgleichungen von Bedeutung ist. Für differentiell-algebraische Gleichungen hat $exp4$ Ordnung 3. Weiterhin wird angegeben, dass eine für Rosenbrock-Verfahren bekannte Fehlerschranke auch für exponentielle Verfahren gezeigt werden kann. Daher ist $exp4$ auch für steife Differentialgleichungen geeignet.

Sei $\|v\| = 1$. Dann folgt mit

$$\varphi(\tau J)v \approx V_m \varphi(\tau H_m) e_1$$

aus der obigen Rekursionsformel

$$\varphi(2\tau J)v \approx V_m \varphi(2\tau H_m)e_1 = V_m \left(\frac{1}{2}\tau H_m \varphi(\tau H_m) + I_m\right)\varphi(\tau H_m)e_1$$

und

$$\varphi(3\tau J)v \approx V_m \left(\frac{2}{3}(\tau H_m \varphi(\tau H_m) + I_m)\varphi(-2\tau H_m) + \frac{1}{3}\varphi(\tau H_m)\right) e_1.$$

Zur Schrittweitensteuerung werden zwei eingebettete Verfahren verwendet und man wählt eine Näherung des lokalen Fehlers als Minimum der lokalen Feh-

lerschätzungen dieser beiden Verfahren. Dies führt dann mit Hilfe eines Schrittweitenwahlverfahrens von Gustafsson [4] auf eine Schrittweite h_{err}.

Weiterhin wird eine vom Krylov-Verfahren abhängige Schrittweite bestimmt. Hierzu wählt man ein Intervall $[\mu, M]$ für die Anzahl der Krylov-Schritte m und eine darin enthaltene erstrebenswerte Anzahl m_{opt}. Die aktuelle Schrittweite h wird beibehalten, wenn $m \in [\mu, M]$ liegt. Falls $m > M$ wird die neue Schrittweite solange reduziert, bis die gewünschte Genauigkeit mit einem $m \in [\mu, M]$ erreicht wird. Falls $m < \mu$ wird

$$h_{kry} = h \left(\frac{m_{opt}}{m} \right)^{\alpha}$$

gesetzt, wobei $\alpha = 1/3$ vorgeschlagen wird. Man kann sogar, falls m mehr als zwei Schritte hintereinander vergleichsweise klein ist, die Schrittweite noch weiter vergrößern. Es wird

$$h_{kry} = 2^{j-1} h$$

vorgeschlagen, falls $m < 4$ in den letzten j Zeitschritten war. Letztendlich wählt man dann die neue Schrittweite als

$$h_{new} = \min\{h_{err}, h_{kry}\}.$$

2.2.3 Explizite exponentielle Runge-Kutta-Verfahren

Explizite exponentielle Runge-Kutta-Verfahren zur Lösung der Differentialgleichung (1.2) sind in den letzten Jahren viel diskutiert worden. Unter anderem stellten Cox und Matthews [2], Hochbruck und Ostermann [8] und Krogstad [10] einige Verfahren vor. Eine übersichtliche Zusammenfassung der besten Verfahren findet man in [8]. Dort werden die Verfahren in einem abstraktem Banachraum für sektorielle Operatoren und lokal Lipschitz-stetigen Nichtlinearitäten analysiert. Weiterhin werden die Ordnungsbedingungen, sowohl für den nicht steifen als auch den steifen Fall aufgeführt und die Konvergenz für Verfahren bis zur Ordnung 4 gezeigt. In diesem Abschnitt sollen kurz die Konstruktionsidee exponentieller Runge-Kutta-Verfahren, sowie 3 explizite Verfahren und die steifen Ordnungsbedingungen vorgestellt werden.

Die Grundlage für die Konstruktion exponentieller Runge-Kutta-Verfahren ist, dass für die Lösung der Differentialgleichung (1.2) die Variation-der-Konstanten-Formel (2.3) gilt. Um nun eine Näherung y_{n+1} an $y(t_n + h)$ zu erhalten, wird der Term $g(t_n + \tau, y(t_n + \tau))$ durch ein Polynom $\widehat{g}_n$ approximiert. Hierzu wählt man Kollokationsknoten $c_1, \ldots, c_s$.

Hat man Approximationen

$$y_n \approx y(t_n) \qquad \text{und} \qquad Y_{ni} \approx y(t_n + c_i h),$$

definiert man $\widehat{g}_n(c_i h) = g(t_n + c_i h, Y_{ni}) = G_{ni}$, so dass

$$\widehat{g}_n(\tau) = \sum_{j=1}^{s} l_j(\tau) G_{nj},$$

wobei l_j das Lagrange-Interpolationspolynom

$$l_j(\tau) = \prod_{m \neq j} \frac{\tau/h - c_m}{c_j - c_m}$$

bezeichnet.

Dadurch ergibt sich die Näherung zur Zeit t_{n+1} als

$$\begin{aligned} y_{n+1} &= e^{-hA} y_n + \int_0^h e^{-(h-\tau)A} \widehat{g}_n(\tau) d\tau \\ &= e^{-hA} y_n + \int_0^h e^{-(h-\tau)A} \sum_{j=1}^{s} l_j(\tau) G_{nj} \\ &= e^{-hA} y_n + h \sum_{j=1}^{s} b_i(-hA) G_{nj} \end{aligned}$$

mit

$$b_i(-hA) = h^{-1} \int_0^h e^{-(h-\tau)A} l_i(\tau) d\tau$$

Um eine Approximation an Y_{ni} zu erhalten, wird derselbe Ansatz benutzt. Es folgt

$$Y_{ni} = e^{-c_i hA} y_n + h \sum_{j=1}^{s} a_{ij}(-hA) G_{nj},$$

mit

$$a_{ij}(-hA) = h^{-1} \int_0^{c_i h} e^{-(c_i h - \tau)A} l_i(\tau) d\tau.$$

Da die Polynome l_j höchstens vom Grad $s-1$ sind, kann man die Koeffizienten $b_i(-hA)$ und $a_{ij}(-hA)$ als Linearkombination der Funktionen

$$\varphi_k(-tA) = t^{-k} \int_0^t e^{-(t-\tau)A} \frac{\tau^{k-1}}{(k-1)!} d\tau, \qquad 1 \leq k \leq s \tag{2.10}$$

schreiben, welche wir später benötigen. Es gilt $\varphi_0(z) = e^z$, sowie

$$\varphi_{k+1}(z) = \frac{\varphi_k(z) - 1/k!}{z}, \qquad \varphi_k(0) = \frac{1}{k!}.$$

Das numerische Schema der Klasse von exponentiellen Runge-Kutta-Verfahren ist dann durch

$$\begin{aligned} y_{n+1} &= e^{-hA} y_n + h \sum_{i=1}^{s} b_i(-hA) G_{ni} \\ Y_{ni} &= e^{-c_i hA} y_n + h \sum_{j=1}^{s} a_{ij}(-hA) G_{nj} \\ G_{nj} &= g(t_n + c_j h, Y_{nj}) \end{aligned} \tag{2.11}$$

gegeben.

Eine wünschenswerte Eigenschaft numerischer Verfahren ist, dass sich Punkte y^*, die sich im Gleichgewicht befinden, erhalten bleiben. Zwei Punkte befinden sich im Gleichgewicht, wenn an den Punkten $y' = 0$ gilt. Aus der Differentialgleichung folgt dann $Ay = g(t, y)$. Dies lässt sich mit der Forderung $Y_{ni} = y_n = y^*$ für alle i, $n \geq 0$ und (2.11) folgendermaßen formulieren:

$$y^* = e^{-hA} y^* + h \sum_{i=1}^{s} b_i(-hA) A y^*$$

$$y^* = e^{-c_i hA} y^* + h \sum_{i=1}^{s} a_{ij}(-hA) A y^*$$

Diese Forderung ist genau dann erfüllt, wenn

$$\sum_{j=1}^{s} b_j(z) = \frac{e^z - 1}{z} = \varphi(z)$$

und

$$\sum_{j=1}^{s} a_{ij}(z) = \frac{e^{c_i z} - 1}{z} = c_i \varphi(c_i z) \qquad i = 1, \ldots, s$$

gilt. Die erste Gleichung liefert

$$e^{-hA} = I - hA \sum_{i=1}^{s} b_i(-hA)$$

woraus

$$\begin{aligned} y_{n+1} &= (I - hA \sum_{i=1}^{s} b_i(-hA)) y_n + h \sum_{i=1}^{s} b_i(-hA) G_{ni} \\ &= y_n + h \sum_{i=1}^{s} b_i(-hA)(G_{ni} - A y_n) \end{aligned}$$

folgt.

Analog folgt

$$Y_{ni} = y_n + h \sum_{j=1}^{s} a_{ij}(-hA)(G_{nj} - Ay_n).$$

Also kann man das numerische Schema (2.11) als

$$\begin{aligned} y_{n+1} &= y_n + h \sum_{i=1}^{s} b_i(-hA)(G_{ni} - Ay_n) \\ Y_{ni} &= y_n + h \sum_{j=1}^{s} a_{ij}(-hA)(G_{nj} - Ay_n) \\ G_{nj} &= g(t_n + c_j h, Y_{nj}) \end{aligned} \tag{2.12}$$

schreiben.

Ein wesentlicher Aspekt sind die Ordnungsbedingungen für den Fall einer steifen Differentialgleichung. Bevor wir diese angeben, betrachten wir die Fehler $e_n = y_n - y(t_n)$ und $E_{ni} = Y_{ni} - y(t_n + c_i h)$. Hierzu setzen wir der Einfachheit halber $f(t) = g(t, y(t))$ und entwickeln f in eine Taylorreihe. Es gilt

$$f(t_n + \tau) = \sum_{j=1}^{q} \frac{\tau^{j-1}}{(j-1)!} f^{(j-1)}(t_n) + \int_0^\tau \frac{(\tau - \sigma)^{q-1}}{(q-1)!} f^{(q)}(t_n + \sigma) d\sigma. \tag{2.13}$$

Setzt man dies in die Variation-der-Konstanten-Formel ein, folgt

$$\begin{aligned} y(t_n + c_i h) &= e^{-c_i hA} y(t_n) + \sum_{j=1}^{q_i} (c_i h)^j (c_i h)^{-j} \int_0^{c_i h} e^{-(c_i h - \tau)A} \frac{\tau^{j-1}}{(j-1)!} f^{(j-1)}(t_n) \\ &+ \int_0^{c_i h} e^{-(c_i h - \tau)A} \int_0^\tau \frac{(\tau - \sigma)^{q_i - 1}}{(q_i - 1)!} f^{(q_i)}(t_n + \sigma) d\sigma d\tau \\ &= e^{-c_i hA} y(t_n) + \sum_{j=1}^{q_i} (c_i h)^j \varphi_j(-c_i hA) f^{(j-1)}(t_n) \\ &+ \int_0^{c_i h} e^{-(c_i h - \tau)A} \int_0^\tau \frac{(\tau - \sigma)^{q_i - 1}}{(q_i - 1)!} f^{(q_i)}(t_n + \sigma) d\sigma d\tau. \end{aligned} \tag{2.14}$$

Setzt man nun die exakte Lösung in das numerische Schema ein, ergibt sich

$$y(t_n + c_i h) = e^{-c_i hA} y(t_n) + h \sum_{j=1}^{i-1} a_{ij}(-hA) f(t_n + c_j h) + \Delta_{ni} \tag{2.15}$$

und

$$y(t_{n+1}) = e^{-hA} y(t_n) + h \sum_{i=1}^{s} b_i(-hA) f(t_n + c_i h) + \delta_{n+1} \tag{2.16}$$

mit den Defekten Δ_{ni} und δ_{n+1}, deren Darstellung man aus (2.14) und (2.15) bzw. (2.16) erhält.

Die Fehlerrekursion erhält man nun indem man (2.15) bzw. (2.16) von (2.11) subtrahiert.

$$E_{ni} = e^{-c_i hA} e_n + h \sum_{j=1}^{i-1} a_{ij}(-hA) \left(g(t_n + c_j h, U_{nj} - f(t_n + c_j h)) - \Delta_{ni} \right. \quad (2.17)$$

$$e_{n+1} = e^{-hA} e_n + h \sum_{i=1}^{s} b_i(-hA) \left(g(t_n + c_i h, U_{ni} - f(t_n + c_i h)) - \delta_{n+1} \right. \quad (2.18)$$

Die explizite Darstellung der Defekte wollen wir hier nicht angeben, sie enthalten die Funktionen

$$\psi_{j,i}(-hA) = \varphi(-c_i hA) c_i^j - \sum_{k=1}^{i-1} a_{ik}(-hA) \frac{c_k^{j-1}}{(j-1)!} \quad (2.19)$$

und

$$\psi_j(-hA) = \varphi_j(-hA) - \sum_{k=1}^{s} b_k(-hA) \frac{c_k^{j-1}}{(j-1)!}, \quad (2.20)$$

welche für die Ordnungsbedingungen benötigt werden. Weiterhin seien im Folgenden

$$J_n = \frac{\partial g}{\partial y}(t_n, y(t_n))$$

und

$$K_n = \frac{\partial^2 g}{\partial t \partial y}(t_n, y(t_n)).$$

Hiermit können wir jetzt die steifen Ordnungsbedingungen angeben. Für eine detailliertere Herleitung sei auf [8] verwiesen.

Num.	Ordnung	Bedingung
1	1	$\psi_1(-hA) = 0$
2	2	$\psi_2(-hA) = 0$
3	2	$\psi_{1,i}(-hA) = 0$
4	3	$\psi_3(-hA) = 0$
5	3	$\sum_{i=1}^{s} b_i(-hA)J\psi_{2,1}(-hA) = 0$
6	4	$\psi_4(-hA) = 0$
7	4	$\sum_{i=1}^{s} b_i(-hA)J\psi_{3,i}(-hA) = 0$
8	4	$\sum_{i=1}^{s} b_i(-hA)J\sum_{j=2}^{i-1} a_{ij}(-hA)J\psi_{2,j}(-hA) = 0$
9	4	$\sum_{i=1}^{s} b_i(-hA)c_iK\psi_{2,i}(-hA) = 0$

Tabelle 2.1: Steife Ordnungsbedingungen für explizite exponentielle Runge-Kutta-Verfahren

Um Bedingung 6 zu erfüllen, muss

$$\sum_{i=2}^{s} b_i(-hA)c_i^3 = 6\varphi_4(-hA)$$

gelten, aber nach ([8], Theorem 4.7) ist es ausreichend die Bedingung in der abgeschwächten Form $\psi_4(0) = 0$ zu erfüllen, d.h.

$$\sum_{i=2}^{s} b_i(0)c_i^3 = \frac{1}{4}.$$

Im Folgenden werden 3 Verfahren angegeben, die von den Autoren als Verfahren 4-ter Ordnung vorgestellt wurden. Die ersten beiden Verfahren wurden konstruiert bevor obige Ordnungsbedingungen bekannt waren, und wir werden sehen, dass diese nicht alle erforderlichen Bedingungen erfüllen. Daher tritt eine Ordnungsreduktion auf.

Um die Verfahren darzustellen, werden hier Butcher-Tableaus verwendet und das Argument der Koeffizientenfunktionen a_{ij} und b_i weggelassen. Ein Butcher-Tableau eines expliziten Verfahrens hat die Form

$$\begin{array}{c|cccc} c_1 = 0 & 0 & \cdots & \cdots & 0 \\ c_2 & a_{21} & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & \cdots & \cdots & b_s \end{array} .$$

Weiterhin werden in den Tableaus die Abkürzungen

$$\varphi_{i,j} = \varphi_{i,j}(-hA) = \varphi_i(-c_j hA), \qquad 2 \le j \le s.$$

benutzt.

Cox und Matthews schlagen in [2] ein 4-stufiges Verfahren vor, welches durch das Tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2}\varphi_{1,2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2}\varphi_{1,3} & & \\ 1 & \frac{1}{2}\varphi_{1,3}(\varphi_{0,3} - 1) & 0 & \varphi_{1,3} & \\ \hline & \varphi_1 - 3\varphi_2 + 4\varphi_3 & 2\varphi_2 - 4\varphi_3 & 2\varphi_2 - 4\varphi_3 & 4\varphi_3 - \varphi_2 \end{array}$$

gegeben ist. Dieses Verfahren erfüllt die Bedingungen 1-4 und Bedingung 6 in der abgeschwächten Form $\psi_4(0) = 0$. Die Bedingungen 5 und 9 sind nur in einer abgeschwächten Form erfüllt, $\psi_{2,2}(0) + \psi_{2,3}(0) = 0$ und $\psi_{2,4}(0) = 0$. Bedingungen 7 und 8 in einer ganz schwachen Form, wo $A = 0$ gesetzt wird. Dies führt im schlechtesten Fall zu einer Ordnungsreduktion auf Ordnung 2.

Ein weiteres 4-stufiges Verfahren wurde von Krogstad in [10] vorgeschlagen. Es ist durch das Tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2}\varphi_{1,2} & & & \\ \frac{1}{2} & \frac{1}{2}\varphi_{1,3} - \varphi_{2,3} & \varphi_{2,3} & & \\ 1 & \varphi_{1,4} - 2\varphi_{2,4} & 0 & 2\varphi_{2,4} & \\ \hline & \varphi_1 - 3\varphi_2 + 4\varphi_3 & 2\varphi_2 - 4\varphi_3 & 2\varphi_2 - 4\varphi_3 & 4\varphi_3 - \varphi_2 \end{array}$$

gegeben. Dieses Verfahren erfüllt Bedingungen 1-5 und 9, sowie die abgeschwächte Bedingung 6. Die Bedingungen 7 und 8 sind auch hier nur in der ganz schwachen Form erfüllt. Daher führt die Ordnungreduktion im schlechtesten Fall auch nur auf Ordnung 3.

Mit 4 Stufen ist es auch nicht möglich ein Verfahren zu konstruieren, dass Ordnung 4 für allgemeine parabolische Probleme der Form (1.2) hat. Daher betrachten wir ein 5-stufiges Verfahren, welches von Hochbruck und Ostermann in [8] vorgeschlagen wurde. Es ist durch das Tableau

$$\begin{array}{c|ccccc}
0 & & & & & \\
\frac{1}{2} & \frac{1}{2}\varphi_{1,2} & & & & \\
\frac{1}{2} & \frac{1}{2}\varphi_{1,3} - \varphi_{2,3} & \varphi_{2,3} & & & \\
1 & \varphi_{1,4} - 2\varphi_{2,4} & \varphi_{2,4} & \varphi_{2,4} & & \\
\frac{1}{2} & \frac{1}{2}\varphi_{1,5} - 2a_{5,2} - a_{5,4} & a_{5,2} & a_{5,2} & \frac{1}{4}\varphi_{2,5} - a_{5,2} & \\
\hline
 & \varphi_1 - 3\varphi_2 + 4\varphi_3 & 0 & 0 & 4\varphi_3 - \varphi_2 & 4\varphi_2 - 8\varphi_3
\end{array}$$

gegeben. Dabei ist

$$a_{5,2} = \frac{1}{2}\varphi_{2,5} - \varphi_{3,4} + \frac{1}{4}\varphi_{2,4} - \frac{1}{2}\varphi_{3,5}.$$

Dieses Verfahren wurde unter Berücksichtigung der steifen Ordnungsbedingungen konstruiert und hat Ordnung 4.

Bei den numerischen Vergleichen in Kapitel 5 wird das Verfahren von Krogstad verwendet. Die verwendete Version arbeitet mit konstanter Schrittweite und beinhaltet eine geschickt angewandte QR-Zerlegung, um Rechenzeit einzusparen. Die Idee wird hier kurz erläutert. Für Details sei auf [9] verwiesen.

Benötigt man das das Produkt von $\varphi(-hA)$ mit unterschiedlichen Vektoren

$$\varphi(-hA)b_j, \qquad b_j = f(t + c_j h), \qquad 1 \le j \le s,$$

berechnet man

$$QR = [b_1, \ldots, b_s], \quad \text{mit} \quad Q = [q_1, \ldots, q_s] \quad \text{und} \quad R \in \mathbb{R}^{s,s},$$

wobei Q orthogonal ist und R obere Dreicksform hat. Dann verwendet man $\mathcal{K}_m(A, q_j)$ anstelle von $\mathcal{K}_m(A, b_j)$, um $z_j = \varphi(-hA)q_j$ zu approximieren. Dann gilt

$$\varphi(-hA)b_j = \varphi(-hA)[q_1, \ldots, q_j]R_{1:j,j} = [z_1, \ldots, z_j]R_{1:j,j}.$$

3 RKC

In diesem Kapitel betrachten wir ein explizites Runge-Kutta-Chebyshev-Verfahren zur Berechnung einer Lösung von (1.1). 1997 wurde von Sommeijer [17] ein Fortran Programm vorgestellt, welches, genau wie die bisher vorgestellten Verfahren, zur Integration steifer Systeme gewöhnlicher Differentialgleichungen dient, die bei der Ortsdiskretisierung parabolischer partieller Differentialgleichungen auftreten. Hier werden die wesentlichen Aspekte des Verfahrens und des Fortran-Codes vorgestellt und die Optionen des erstellten Matlab Programms erläutert. In Kapitel 5 findet man die numerischen Resultate. Bevor wir das Verfahren betrachten, benötigen wir eine kurze Vorbereitung.

3.1 Stabilitätsbereiche und Stabilitätsfunktionen

Ein allgemeines s-stufiges Runge-Kutta-Verfahren angewandt auf (1.1) ist von der Form

$$\begin{aligned} Y_i &= y_n + h\sum_{j=1}^{s} a_{ij}Y_j' \\ Y_i' &= f(t_n + c_i h, Y_i) \\ y_{n+1} &= y_n + h\sum_{i=1}^{s} b_i Y_i' \end{aligned}$$

mit den Kollokationsknoten c_i, den Gewichten b_i und den Koeffizienten a_{ij}. Für ein explizites Runge-Kutta-Verfahren angewandt auf die Testgleichung

$$y'(t) = \lambda y(t) \tag{3.1}$$

erhält man also

$$\begin{aligned} Y_i &= y_n + h\sum_{j=1}^{i-1} a_{ij}Y_j' \\ Y_i' &= \lambda Y_i \\ y_{n+1} &= y_n + h\sum_{i=1}^{s} b_i Y_i' \end{aligned}$$

und daher

$$y_{n+1} = P(h\lambda)y_n \tag{3.2}$$

mit einem Polynom P vom Grad $\leq s$. P wird als Stabilitätsfunktion bezeichnet.

Der Stabilitätsbereich eines Runge-Kutta-Verfahrens ist definiert als

$$S = \{z \in \mathbb{C} \mid z = h\lambda;\ \text{das Verfahren liefert eine beschränkte Lösung } (y_n)_{n\geq 0},\\ \text{wenn es mit Schrittweite } h \text{ auf } y' = \lambda y,\ y(0) = y_0 \text{ angewandt wird.}\}.$$

Aus (3.2) folgt für ein explizites Runge-Kutta-Verfahren

$$S = \{z \in \mathbb{C} \mid |P(z)| \leq 1\}.$$

Die Stabilitätsbereiche expliziter Runge-Kutta-Verfahren umfassen typischerweise einen Bereich um die negative reelle Achse.

3.2 Vorbemerkungen

Die Konstruktion des Verfahrens beruht auf einer geschickten Wahl der Stabilitätsfunktion. Wie wir in Abschnitt 3.3 sehen werden, werden hierzu Chebyshev-Polynome gewählt.

Im Gegensatz zu exponentiellen Integratoren wird bei dem Verfahren nicht die Jacobi-Matrix $\frac{\partial f(t,y)}{\partial y}$ benötigt, sondern nur eine obere Schranke für den Spektralradius dieser. Das Verfahren hat Ordnung 2 und verwendet eine variable Stufenanzahl s, welche so mit der Schrittweite angepasst wird, dass die erforderliche Stabilitätsbedingung erfüllt ist, worauf wir in Abschnitt 3.3 eingehen werden.

RKC ist geeignet für Probleme deren Jacobi-Matrix fast normal ist und deren Eigenwerte alle nah an der negativen reellen Achse liegen. Dies ist sicherlich der Fall, wenn $\frac{\partial f(t,y)}{\partial y}$ symmetrisch und negativ definit ist, was bei der Diskretisierung elliptischer Operatoren häufig der Fall ist. Die Formeln, die in RKC verwendet werden, sind Erweiterungen einer in [18] vorgeschlagenen Familie expliziter Runge-Kutta-Chebyshev-Formeln. Die Stabilitätsbereiche aller Formeln beinhalten einen schmalen Streifen um die negative reelle Achse. Ein großer Vorteil ist, dass um die Formeln auszuwerten nur wenige Vektoren gespeichert werden müssen, unabhängig von der Anzahl der Stufen. Ein weiterer Vorteil ist, dass die lokalen Fehler für alle Formeln die gleichen sind. Daher kann RKC zuerst die effizienteste Schrittweite wählen, um dann den Spektralradius zu approximieren und die effizienteste Formel zu wählen, welche für die Schrittweite stabil ist.

Beispielsweise können Reaktions-Diffusions-Systeme

$$y' = \nabla \cdot (K \nabla y) + f(y, x, t), \qquad x \in \mathbb{R}^d$$

effizient mit RKC gelöst werden, wenn der Reaktionsterm f nicht zu steif ist. Falls f zu großer Steifheit führt, kann RKC auch in einem Splitting-Verfahren verwendet werden, wo der Reaktionsterm an den Gitterpunkten mit einem

Standard-Integrator für steife Probleme behandelt wird. Hierunter fallen auch Transportprobleme wie

$$y' + \nabla \cdot (ay) = \nabla \cdot (K \nabla y) + f(y, x, t), \qquad x \in \mathbb{R}^d,$$

welche z.B. bei der Untersuchung der Umweltverschmutzung auftreten, [16].

3.3 Die Runge-Kutta-Chebyshev-Formeln, Stabilitätsbedingungen und Konvergenzresultate

Das Ziel ist Formeln zu konstruieren, die auf einem möglichst großen Bereich um die negative reelle Achse stabil sind. Die Länge dieses Bereichs wird im Folgenden mit $\beta(s)$ bezeichnet und ist proportional zu s^2, wie wir später sehen werden.

Hat man eine Näherung y_n an $y(t_n)$ und bezeichnet mit h die Schrittweite, so verwendet man folgenden Ansatz für das Runge-Kutta-Schema:

$$\begin{array}{rcl} Y_0 & = & y_n, \\ Y_1 & = & Y_0 + \widetilde{\mu}_1 h f_0, \\ Y_j & = & (1 - \mu_j - \nu_j) Y_0 + \mu_j Y_{j-1} + \nu_j Y_{j-2} + \widetilde{\mu}_j h f_{j-1} + \widetilde{\gamma}_j h f_0, \\ y_{n+1} & = & Y_s, \end{array} \tag{3.3}$$

mit $j = 2, ..., s$ und $f_j = f(t_n + c_j h, Y_j)$.

Eine Formel für die Knoten c_j erhält man aus der Entwicklung von $Y_j \approx y(t_n + c_j h)$.

$$Y_j \approx y(t_n + c_j h) = y(t_n) + c_j h y'(t_n) + \mathcal{O}(h^2). \tag{3.4}$$

Vernachlässigt man nun die Terme der Größenordnung $\mathcal{O}(h^2)$ und setzt (3.3) und (3.4) gleich, erhält man sofort $c_0 = 0$, $c_1 = \widetilde{\mu}_1$ und

$$y(t_n) + c_j h y'(t_n) = (1 - \mu_j - \nu_j) Y_0 + \mu_j Y_{j-1} + \nu_j Y_{j-2} + \widetilde{\mu}_j h f_{j-1} + \widetilde{\gamma}_j h f_0.$$

Mit $Y_{j-1} \approx y(t_n + c_{j-1} h)$, $Y_{j-2} \approx y(t_n + c_{j-2} h)$ und $f(t_n + c_{j-1} h, Y_{j-1}) = y'(t_n + c_{j-1} h)$ folgt

$$\begin{array}{rcl} y(t_n) + c_j h y'(t_n) & = & (1 - \mu_j - \nu_j) y_n + \mu_j y(t_n + c_{j-1} h) + \nu_j y(t_n + c_{j-2} h) \\ & + & \widetilde{\mu}_j h y'(t_n + c_{j-1} h) + \widetilde{\gamma}_j h y'(t_n) \end{array}$$

und daher

$$\begin{array}{rcl} y(t_n) + c_j h y'(t_n) & = & (1 - \mu_j - \nu_j) y_n + \mu_j y(t_n) + \mu_j c_{j-1} h y'(t_n) + \nu_j y(t_n) \\ & + & \nu_j c_{j-2} h y'(t_n) + \widetilde{\mu}_j h y'(t_n) + \widetilde{\gamma}_j h y'(t_n) \end{array}$$

Mit $y_n \approx y(t_n)$ folgt

$$c_j h y'(t_n) = \mu_j c_{j-1} h y'(t_n) + \nu_j c_{j-2} h y'(t_n) + \widetilde{\mu}_j h y'(t_n) + \widetilde{\gamma}_j h y'(t_n)$$

und die Rekursion

$$c_0 = 0, \quad c_1 = \widetilde{\mu}_1, \quad c_j = \mu_j c_{j-1} + \nu_j c_{j-2} + \widetilde{\mu}_j + \widetilde{\gamma}_j, \quad 2 \leq j \leq s.$$

Setzt man rekursiv in (3.3) ein, so erhält man die Standardform eines Runge-Kutta-Verfahrens

$$Y_j = y_n + h \sum_{l=0}^{j-1} a_{jl} f(t_n + c_l h, Y_l)$$

und es gilt die bei Runge-Kutta-Verfahren übliche Bedingung

$$c_j = \sum_{l=0}^{j-1} a_{jl}.$$

Auf Seite 27 findet man eine Formel für die Koeffizienten a_{jl}.

Ursprünglich wurde eine RKC-Methode der Ordnung 1 und eine der Ordnung 2 konstruiert. Da die Methode der Ordnung 2 effizienter ist, wird nur diese hier beschrieben.

Die einzige Bedingung, um Konsistensordnung 1 zu erhalten, ist $c_s = 1$. Um die Bedingung für Konsistenzordnung 2 zu erhalten, entwickeln wir (3.4) um eine Potenz weiter

$$Y_j = y(t_n) + c_j h y'(t_n) + X_j h^2 y''(t_n) + \mathcal{O}(h^3). \tag{3.5}$$

Setzt man dies in das numerische Schema (3.3) ein erhält man

$$X_0 = X_1 = 0, \quad X_j = \mu_j X_{j-1} + \nu_j X_{j-2} + \widetilde{\mu}_j c_{j-1}, \quad 2 \leq j \leq s.$$

Also hat die RKC-Methode Konsistenzordnung 2 in jeder Stufe, falls $X_j = \frac{1}{2} c_j^2$, für $2 \leq j \leq s$ gilt. Hieraus erhält man

$$c_2^2 = 2\widetilde{\mu}_2 c_1, \quad c_3^2 = \mu_3 c_2^2 + 2\widetilde{\mu}_3 c_2$$

und

$$c_j^2 = \mu_j c_{j-1}^2 + \nu_j c_{j-2}^2 + 2\widetilde{\mu}_j c_{j-1}, \quad 4 \leq j \leq s,$$

was sich leicht durch Induktion überprüfen lässt. In [15] wird gezeigt, dass diese Bedingungen erfüllbar sind. Eine weitere Forderung ist, dass alle Punkte $t_n + c_j h$ im Intervall $[t_n, t_{n+1}]$ liegen und

$$0 = c_0 < c_1 < \ldots < c_s = 1$$

erfüllen.

Um den Stabilitätsbereich und die Formeln für die Koeffizienten des Verfahrens zu erhalten, betrachten wir die Testgleichung $y'(t) = \lambda y(t)$. Dies liefert

$$y_{n+1} = P_s(z) y_n, \quad z = h\lambda,$$

wobei $P_s(z)$ die Stabilitätsfunktion vom Grad s ist. Mit (3.3) ergibt sich die rekursive Darstellung

$$P_0(z) = 1, \quad P_1(z) = 1 + \widetilde{\mu}_1 z,$$

$$P_j(z) = (1 - \mu_j - \nu_j) + \widetilde{\gamma}_j z + (\mu_j + \widetilde{\mu}_j z) P_{j-1}(z) + \nu_j P_{j-2}(z), \quad 2 \le j \le s. \tag{3.6}$$

Nun wählt man die Koeffzienten μ_j, ν_j, $\widetilde{\mu}_j$ und $\widetilde{\gamma}_j$ in (3.6), so dass die Stabilitätsschranke

$$\beta(s) = \max\{-z | z \le 0, |P_s(z)| \le 1\}$$

so groß wie möglich ist.

Die Idee ist, die Stabilitätsfunktionen $P_j(z)$ durch Chebyshev-Polynome darzustellen, welche durch

$$T_s(x) = \cos(s \ \ \arccos(x)), \quad -1 \le x \le 1$$

definiert sind, und dann die Koeffizienten der Rekursion für die gewählten Polynome und denen von der Rekursion (3.6) anzupassen. Beispielsweise erfüllen die Polynome $P_j(z) = T_j(1 + z/s^2)$ die Rekursion

$$\begin{aligned} P_0(z) &= 1, \quad P_1(z) = 1 + \frac{z}{s^2}, \\ P_j(z) &= 2\left(1 + \frac{z}{s^2}\right) P_{j-1}(z) - P_{j-2}(z) \end{aligned}$$

und ein Koeffizientenvergleich mit (3.6) liefert sofort

$$\widetilde{\mu}_1 = \frac{1}{s^2}, \quad \mu_j = 2, \quad \widetilde{\mu}_j = \frac{2}{s^2}, \quad \nu_j = -1, \quad \widetilde{\gamma}_j = 0, \quad 2 \le j \le s.$$

Für die RKC-Formeln wird

$$P_j(z) = a_j + b_j T_j(w_0 + w_1 z), \quad 0 \le j \le s,$$

mit

$$a_j = 1 - b_j T_j(w_0), \quad b_j = \frac{T_j''(w_0)}{(T_j'(w_0))^2}, \quad 2 \le j \le s,$$

$$w_0 = 1 + \frac{\epsilon}{s^2}, \quad w_1 = \frac{T_s'(w_0)}{T_s''(w_0)}, \quad a_0 = 1 - b_0, \quad a_1 = 1 - b_1 w_0, \quad b_0 = b_1 = b_2$$

gewählt, [16]. w_0 wird dabei verwendet, um $P_s(z)$ im Intervall $[-1 + \epsilon, 1 - \epsilon]$ alternieren zu lassen. Für den Parameter ϵ wird $\frac{2}{13}$ vorgeschlagen.

Mit der Rekursionsformel für Chebyshev-Polynome

$$T_j(x) = 2x T_{j-1}(x) - T_{j-2}(x)$$

und der Beziehung

$$T_j(w_0 + w_1 z) = \frac{1}{b_j}\Big(P_j(z) - (1 - b_j T_j(w_0))\Big)$$

folgt

$$\begin{aligned}
P_j(z) &= 1 - b_j T_j(w_0) + b_j T_j(w_0 + w_1 z) \\
&= 1 - b_j T_j(w_0) + b_j\Big(2(w_0 + w_1 z)T_{j-1}(w_0 + w_1 z) - T_{j-2}(w_0 + w_1 z)\Big) \\
&= 1 - b_j\Big(2w_0 T_{j-1}(w_0) - T_{j-2}(w_0)\Big) \\
&+ b_j\Bigg(2(w_0 + w_1 z)\frac{1}{b_{j-1}}\Big(P_{j-1}(z) - (1 - b_{j-1}T_{j-1}(w_0))\Big) \\
&- \frac{1}{b_{j-2}}\Big(P_{j-2}(z) - (1 - b_{j-2}T_{j-2}(w_0))\Big)\Bigg) \\
&= \Big(1 - \frac{2b_j w_0}{b_{j-1}} + \frac{b_j}{b_{j-2}}\Big) - \Big(\frac{2b_j w_1}{b_{j-1}} - 2b_j w_1 T_{j-1}(w_0)\Big)z \\
&+ \Big(\frac{2b_j w_0}{b_{j-1}} + \frac{2b_j w_1}{b_{j-1}}z\Big)P_{j-1}(z) - \frac{b_j}{b_{j-2}}P_{j-2}(z) \\
&+ \Big(-2b_j w_0 T_{j-1}(w_0) + b_j T_{j-2}(w_0) + 2b_j w_0 T_{j-1}(w_0) - b_j T_{j-2}(w_0)\Big).
\end{aligned}$$

Also erfüllen die Polynome die Rekursion

$$\begin{aligned}
P_0(z) &= 1, \quad P_1(z) = 1 + b_1 w_1 z, \\
P_j(z) &= \left(1 - \frac{2b_j w_0}{b_{j-1}} - \frac{-b_j}{b_{j-2}}\right) - (1 - b_{j-1}T_{j-1}(w_0))\frac{2b_j w_1}{b_{j-1}}z \\
&+ \left(\frac{2b_j w_0}{b_{j-1}} + \frac{2b_j w_1}{b_{j-1}}z\right)P_{j-1}(z) + \frac{-b_j}{b_{j-2}}P_{j-2}(z).
\end{aligned}$$

Der Koeffizientenvergleich mit (3.6) liefert:

$$\widetilde{\mu}_1 = b_1 w_1, \quad \mu_j = \frac{2b_j w_0}{b_{j-1}}, \quad \nu_j = \frac{-b_j}{b_{j-2}}, \quad \widetilde{\mu}_j = \frac{2b_j w_1}{b_{j-1}},$$
$$\widetilde{\gamma}_j = -(1 - b_{j-1}T_{j-1}(w_0))\widetilde{\mu}_j$$

für $2 \le j \le s$.

Für diese Wahl der Parameter gilt

$$\beta(s) \approx \frac{(w_0 + 1)T_s''(w_0)}{T_s'(w_0)} \approx \frac{2}{3}(s^2 - 1)(1 - \frac{2}{15}\epsilon), \quad \epsilon \to 0, \tag{3.7}$$

[18].

Für die Stabilitätsbetrachtung wird die Semidiskretisierung einer linearen partiellen Differentialgleichung

$$y'(t) = Ay(t) + g(t), \quad 0 < t \leq T$$

betrachtet und vorausgesetzt, dass A symmetrisch ist und negative Eigenwerte $\lambda_i(M)$ besitzt.

Da A normal ist, gilt $\|A\| = \sigma(A)$, wobei σ den Spektralradius bezeichnet. Wenn $h\lambda$ die Werte des Spektrums von hA durchläuft, dann nimmt das Spektrum des Matrixpolynoms $P(hA)$ die Werte $P(h\lambda)$ an. Da $P(hA)$ auch normal ist, gilt

$$\|P(hA)\| = \sigma(P(hA)) = \max_{h\lambda} |P(h\lambda)|$$

und wegen der Voraussetzung an A

$$-h\sigma(A) \leq h\lambda_i(A) \leq \max(h\lambda_i(A)) \leq 0.$$

Daher gilt $\|P_s(hA)\| \leq 1$, wenn man die Schrittweite und die Stufenanzahl so wählt, dass die Stabilitätsbedingung

$$h\sigma(A) \leq \beta(s) \tag{3.8}$$

erfüllt ist.

Um die Konvergenzresultate angeben zu können, betrachten wir zunächst die Auswirkungen von Störungen wie z.b. Rundungsfehlern. Dazu fügen wir in dem Schema (3.3) eine Störung r_j hinzu

$$\begin{aligned}
\widetilde{Y}_0 &= \widetilde{y}_n, \\
\widetilde{Y}_1 &= \widetilde{Y}_0 + \widetilde{\mu}_1 h\widetilde{f}_0 + r_1, \\
\widetilde{Y}_j &= (1-\mu_j-\nu_j)\widetilde{Y}_0 + \mu_j\widetilde{Y}_{j-1} + \nu_j\widetilde{Y}_{j-2} + \widetilde{\mu}_j h\widetilde{f}_{j-1} + \widetilde{\gamma}_j h\widetilde{f}_0 + r_j, \\
\widetilde{y}_{n+1} &= \widetilde{Y}_s,
\end{aligned}$$

wobei $\widetilde{f}_j = f_j(t+c_jh, \widetilde{Y}_j)$. Für den Fehler $e_n = \widetilde{y}_n - y_n$ wird folgende Abschätzung gezeigt, ([21], Theorem 3.1):

Satz 3.1 Sind h und s so gewählt, dass die Bedingung $h\sigma(A) \leq \beta$ erfüllt ist. Dann gilt

$$\|e_{n+1}\| \leq \|e_n\| + C\sum_{k=1}^{s}(s-k+1)\|r_k\| \leq \|e_n\| + \frac{1}{2}s(s+1)C\max_k\|r_k\|, \tag{3.9}$$

wobei die Konstante C moderate Größe hat und unabhängig von A, h und s ist.

Die Abschätzung (3.9) und die Betrachtung der lokalen Defekte, welche auftreten wenn man die exakte Lösung in das Runge-Kutta-Schema einsetzt, liefern

dann eine Schranke für den globalen Fehler. Die lokalen Defekte leitet man über die Differentialgleichung

$$y_l'(t) = f(t, y_l(t)) + \alpha_l(t), \quad 0 < t \leq T.$$

her, worauf wir hier nicht näher eingehen wollen. Dabei ist $y_l(t)$ eine exakte Lösung auf einem Gitter mit Gitterweite l, $\alpha_l(t)$ der lokale Abschneidefehler und es gilt

Satz 3.2 Unter den Voraussetzungen $y_l \in \mathcal{C}^3[0,T]$ und $h\sigma(A) \leq \beta$ für das ungedämpfte ($\epsilon = 0$ für w_0) Verfahren gilt für den globalen Fehler

$$\|e_n\| \leq C(s^{-3}h \max_{0\leq t\leq T} \|y_l^{(2)}(t)\| + h^2 \max_{0\leq t\leq T} \|y_l^{(3)}(t)\| + \max_{0\leq t\leq T} \|\alpha_l(t)\|),$$

mit $n = 1, 2, \ldots$ und $nh \leq T$ ([21], Theorem 5.2). Dabei ist $C > 0$ eine von A, h und s unabhängige Konstante.

Diese Abschätzung zeigt, dass das Verfahren 'fast' Ordnung 2 hat, wenn $s^{-3} \approx h$ gilt. Weiterhin sieht man, dass der globale und auch lokale Fehler hauptsächlich von der dritten Ableitung der Lösung abhängen und weitgehend unabhängig von s sind.

3.4 Implementation

Der von Sommeijer vorgestellte Fortran-Code wählt genau wie Adams, BDF und Extrapolations-Verfahren die zu verwendende Formel dynamisch. Die meisten Verfahren für Anfangswertprobleme schätzen in jedem Schritt den lokalen Fehler und passen die Schrittweite so an, dass das Problem effizient gelöst wird. Die größte Schwierigkeit die effizienteste Formel auszuwählen, ist es die richtige Schrittweite für diese Formel zu wählen. RKC approximiert zunächst den Spektralradius der Jacobi-Matrix und ermittelt dann die effizienteste Formel, die mit der jeweiligen Schrittweite im Sinne von (3.8) stabil ist. In 3.4.1 wird die Fehlerkontrolle beschrieben und in 3.4.2 die Schrittweitenwahl, sowie die Berechnung des Spektralradius.

3.4.1 Fehlerkontrolle

Für eine glatte Funktion f ist die Taylorentwicklung der Lösung bei $t = t_n$ durch

$$y_{n+1} = y + hy' + \frac{h^2}{2}y'' + \frac{h^3}{6}y''' + \mathcal{O}(h^4)$$

gegeben. Aus der Runge-Kutta-Theorie ist bekannt, dass man die Ableitungen durch elementare Differentiale ausdrücken kann. Dabei gilt

$$y^{(k)}(t_0) = \sum_{\kappa\in\Upsilon,\rho(\kappa)=k} \alpha(\kappa)F(\kappa)(y_0),$$

wobei Υ die Menge aller Bäume bezeichnet, F die elementaren Differentiale und $\alpha(\kappa)$ eine von der Differentialgleichung unabhängige Konstante ist.

Die elementaren Differentiale bis zur Ordnung 3 sind durch

$$y' = f, \quad y'' = f'y' = f'f \quad \text{und} \quad y''' = f''(f,f) + f'f'f$$

gegeben. Setzt man dies nun in die Taylorentwicklung ein, so ergibt sich

$$y_{n+1} = y + hy' + \frac{h^2}{2}y'' + C_{1,s}h^3 f'f'f + C_{2,s}h^3 f''(f,f) + \mathcal{O}(h^4), \qquad s \geq 2,$$

wobei die Konstanten $C_{1,s}$ und $C_{2,s}$ von der Formel und der Anzahl der Stufen s abhängen. In [17] wird angegeben, dass beide Konstanten für wachsendes s nahe bei $1/10$ liegen. Um dies zu überprüfen, verwenden wir die aus der Runge-Kutta-Theorie bekannten Beziehungen. Betrachtet man ein Runge-Kutta-Tableau

$$\begin{array}{c|c} \widetilde{c}_i & \widetilde{a}_{i,j} \\ \hline & \widetilde{b}_i \end{array}$$

so gilt

$$C_{1,s} = \frac{1}{2}\sum_{i,j,k}^{s} \widetilde{b}_i \widetilde{a}_{i,j}\widetilde{a}_{i,k} \ \text{ und } \ C_{2,s} = \sum_{i,j,k}^{s} \widetilde{b}_i \widetilde{a}_{i,j}\widetilde{a}_{j,k}.$$

Die Runge-Kutta-Koeffizienten $\widetilde{a}_{i,j}$ und $\widetilde{b}_i$ kann man aus der Standardform eines Runge-Kutta-Verfahrens ablesen. Um diese zu erhalten, betrachten wir den Übergang von Y_{j-1} nach Y_j in (3.3). Es gilt

$$\begin{aligned} Y_j &= y_n + h\mu_j \sum_{k=0}^{j-2} \widetilde{a}_{j-1,k} f_k + h\nu_j \sum_{k=0}^{j-3} \widetilde{a}_{j-2,k} f_k + h\widetilde{\mu}_j f_{j-1} + h\widetilde{\gamma}_j f_0 \\ &= y_n + h\sum_{k=0}^{j-1} \widetilde{a}_{j,k} f_k. \end{aligned} \tag{3.10}$$

Hieraus liest man die Rekursion

$$\begin{aligned} \widetilde{a}_{j,0} &= \mu_j \widetilde{a}_{j-1,0} + \nu_j \widetilde{a}_{j-2,0} + \widetilde{\gamma}_j \\ \widetilde{a}_{j,k} &= \mu_j \widetilde{a}_{j-1,k} + \nu_j \widetilde{a}_{j-2,k} \\ \widetilde{a}_{j,j-1} &= \widetilde{\mu}_j \end{aligned} \tag{3.11}$$

ab und aus $y_{n+1} = Y_s$ folgt $\widetilde{b}_i = \widetilde{a}_{s,i}$. Berechnet man nun die beiden Konstanten, so ergibt sich

$$C_{1,200} \approx 0.1011410626225052, \qquad C_{1,300} \approx 0.1011452066261325$$

und

$$C_{2,200} \approx 0.1011410626153832, \qquad C_{2,300} \approx 0.1011452066252039.$$

In den Abbildungen 3.1 und 3.2 sind $|C_{1,s} - C_{1,300}|$ und $|C_{2,s} - C_{2,300}|$ für $s = 2, \ldots, 300$ dargestellt.

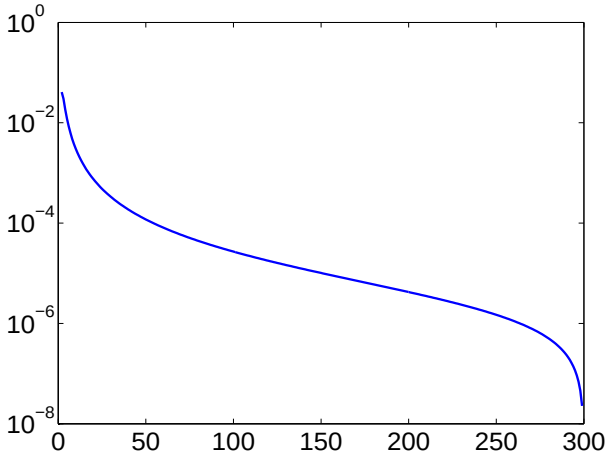

Abb. 3.1: $|C_{1,s} - C_{1,300}|$ als Funktion der Stufenanzahl s.

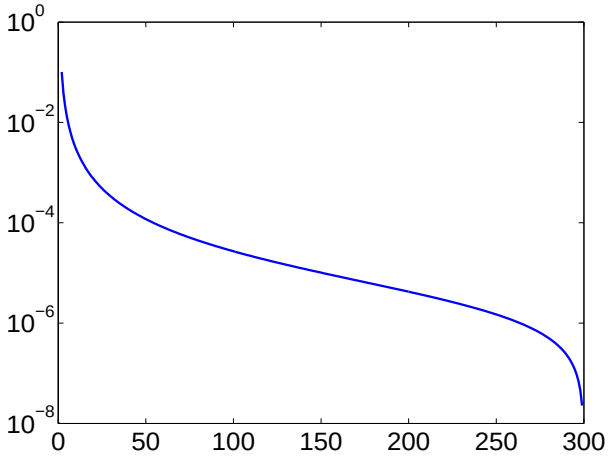

Abb. 3.2: $|C_{2,s} - C_{2,300}|$ als Funktion der Stufenanzahl s.

Bezeichnet man nun mit $Err(t_{n+1})$ die Näherung an den führenden Term der Taylorentwicklung des lokalen Fehlers und ersetzt die von s abhängigen Konstanten durch ihre Grenzwerte, so gilt

$$Err(t_{n+1}) = \frac{h^3}{15} y'''(t_n).$$

Dieser Ausdruck wird durch

$$Est_{n+1} = \frac{1}{15}\Big(12(y_n - y_{n+1}) + 6h\big(f(y_n) + f(y_{n+1})\big)\Big) \tag{3.12}$$

approximiert. Um dies einzusehen, betrachten wir die Gleichung

$$\begin{aligned} h^3 y'''(t_n) &= ay_n + by_{n+1} + chf(y_n) + dhf(y_{n+1}) \\ &= ay(t_n) + by(t_n + \tau) + chy'(t_n) + dhy'(t_n + h) \end{aligned}$$

und stellen ein lineares Gleichungssystem für die Koeffizienten a, b, c und d auf. Mit Taylorentwicklung folgt

$$\begin{aligned} h^3 y'''(t_n) &= ay(t_n) + by(t_n + h) + chy'(t_n) + dhy'(t_n + h) \\ &= ay(t_n) + by(t_n) + hby'(t_n) + \frac{h^2}{2}by''(t_n) + \frac{h^3}{6}by'''(t_n) \\ &+ hcy'(t_n) + hdy'(t_n) + h^2 dy''(t_n) + \frac{h^3}{2}dy'''(t_n) + \mathcal{O}(h^4) \\ &\approx (a+b)y(t_n) + (b+c+d)hy'(t_n) + \Big(\frac{1}{2}b + d\Big)h^2 y''(t_n) \\ &+ \Big(\frac{1}{6}b + \frac{1}{2}d\Big)h^3 y'''(t_n). \end{aligned} \tag{3.13}$$

Um (3.13) zu erfüllen, müssen

$$\frac{1}{6}b + \frac{1}{2}d = 1, \quad \frac{1}{2}b + d = 0, \quad b + c + d = 0 \text{ und } a + b = 0$$

gelten. Hieraus folgt $a = 12$, $b = -12$, $c = 6$ und $d = 6$, was (3.12) bestätigt.

Der Benutzer kann zwei Toleranzen vorgeben. Zum einen die relative Toleranz *rtol* als Skalar, und zum anderen die absolute Toleranz *atol*, welche man als Skalar angeben kann, aber auch als Vektor, dessen Komponenten dann die Genauigkeit in der zugehörigen Komponente der Lösung vorgeben. Diese Toleranzen gehen dann in eine gewichtete Norm ein

$$\|Est_{n+1}\| = \|w^{-1}Est_{n+1}\|_2, \qquad w = \sqrt{m}\ \ diag(Tol_1, ..., Tol_m).$$

Dabei ist

$$Tol_k = atol_k + rtol|y_{n+1,k}|,$$

m die Dimension des Problems und $y_{n+1,k}$ die k-te Komponente von y_{n+1}. Falls $\|Est_{n+1}\| \leq 1$, wird der Schritt akzeptiert und sonst wiederholt. Die Fehlerkontrolle beinhaltet eine Art *relaxation*, wie wir sie in Kapitel 4 auch sehen werden. In [13] wird gezeigt, dass sich der Fehler ungefähr um den Faktor 0.2 reduziert, wenn man die Toleranzen mit 0.1 multipliziert.

3.4.2 Schrittweitenwahl und Berechnung des Spektralradius

Da es teuer ist einen Schritt zu wiederholen, wird versucht die Anzahl der verworfenen Schritte möglichst gering zu halten, indem die optimale Schrittweite vorhergesagt wird. Hierzu wird wie bei *exp*4 ein Verfahren von Gustafsson [4] verwendet.

Hier wird nach einem erfolgreichen Schritt die neue Schrittweite als

$$h_{new} = \min(10, \max(0.1, fac))h$$

gewählt, wobei

$$fac = 0.8\left(\frac{\|Est_n\|^{1/(p+1)}}{\|Est_{n+1}\|^{1/(p+1)}}\frac{h_n}{h_{n-1}}\right)\frac{1}{\|Est_{n+1}\|^{1/(p+1)}}.$$

Nach einem verworfenen Schritt wird

$$fac = 0.8\frac{1}{\|Est_{n+1}\|^{1/(p+1)}}$$

verwendet. Hier wird $p = 2$ gewählt.

In jedem Schritt wählt RKC eine Schrittweite und anschließend eine mit dieser Schrittweite stabile Formel, um den lokalen Fehler zu steuern. Die Stabilitätsbereiche der Formeln beinhalten einen Bereich um die negative reelle Achse. Die Länge $\beta(s)$ dieses Bereichs kann mit (3.7) durch $0.653s^2$ abgeschätzt werden. Um Stabilität für die Schrittweite zu erhalten, reicht es eine obere Schranke des Spektralradius der Jacobi-Matrix zu kennen, sofern die Eigenwerte in dem jeweiligen Segment liegen. Daher wird

$$h\sigma\left(\frac{\partial f(t,y)}{\partial y}\right) \leq 0.653s^2 \tag{3.14}$$

als Auswahlkriterium für die Anzahl der Stufen verwendet.

Der Spektralradius wird mit einer Potenzmethode approximiert, welche üblicherweise das Produkt der Matrix mit einem Vektor verwenden. Da die Jacobi-Matrix J im gesamten Verfahren nicht verwendet wird, wird dies auch hier umgangen. Dazu approximiert man das Produkt Jv durch einen Differenzenquotienten

$$Jv \approx \frac{f(y_n + \varepsilon v)}{\varepsilon} \approx \frac{\partial f}{\partial v}.$$

Literaturhinweise hierzu findet man in [17]. Abbildung 3.3 zeigt den Algorithmus bei dem der Differenzenquotient zusätzlich durch Normen skaliert ist. Da eine obere Schranke für den Spektralradius benötigt wird, wird die Schätzung sicherheitshalber etwas vergrößert. Der Parameter ε wird als $\sqrt{\epsilon}$ gewählt, wobei

ϵ die Maschinengenauigkeit bezeichnet. Dies hat sich als praktikabel herausgestellt, um Rundungsfehler gering zu halten. Da die Berechnung wiederum Speicherplatz und Rechenzeit in Anspruch nimmt, kann der Benutzer eine generell gültige Schranke für den Spektralradius angeben oder eine andere Berechnung vornehmen, was leicht eingebaut werden kann. Für Probleme mit konstanter Jacobi-Matrix wird der Spektralradius nur einmal berechnet, sofern der Benutzer dies angibt.

Eingabe: t_n, y_n, $f_n = f(t_n, y_n)$, $\varepsilon = \sqrt{\epsilon}$ und der Startvektor v, wobei $v = f_n$ beim ersten Aufruf

$v = y_n + (\|y_n\|\varepsilon/\|v\|)v$;
$\sigma = 0$;
for $i = 1$:n
    $f_v = f(t_n, v)$;
    $\sigma_1 = \sigma$;
    $\sigma_2 = \|f_v - f_n\|/\|y_n\|\varepsilon$;
    $\sigma = 1.2 * \sigma_2$;
    if $i \geq 2$ und $|\sigma_2 - \sigma_1| \leq 0.01 * \max(\sigma_2, 1/h_{max})$
        $v = v - y_n$; STOP
    end
    $v = y_n + (\|y_n\|\varepsilon/\|f_v - f_n\|)(f_v - f_n)$;
end

Abb. 3.3: Potenzmethode zur Berechnung des Spektralradius

Ein weiterer Aspekt ist, dass bei expliziten Runge-Kutta-Verfahren Rundungsfehler bei einer großen Anzahl von Stufen eine Rolle spielen können. Falls RKC ein sehr große Stufenanzahl für die aktuelle Schrittweite benötigt, wird die Schrittweite so weit verkleinert, dass das resultierende s akzeptabel und die Stabilitätsbedingung (3.14) erfüllt ist.

4 Vorkonditionierte Lanczos-Approximation

In Kapitel 2 haben wir gesehen, dass in exponentiellen Integratoren das Matrix-Vektor-Produkt

$$w(\tau) = \varphi(-\tau A)v,$$

wobei φ die in (2.7) definierte Funktion ist, benötigt wird. In diesem Kapitel soll ein vorkonditioniertes Lanczos-Verfahren vorgestellt werden, um dieses Produkt zu approximieren, wobei hier $\|v\| = 1$ vorausgesetzt ist. Das Ziel ist, durch die Vorkonditionierung die Anzahl der benötigten Lanczos-Schritte zu verringern. In diesem Kapitel ist A als symmetrisch und positiv semidefinit vorausgesetzt.

4.1 Idee

Meistens finden Lanczos-Verfahren sehr schnell Näherungen an Eigenwerte, die in gewissem Sinne gut voneinander getrennt sind. Die Idee ist das Spektrum der Matrix A so zu transformieren, dass das Lanczos-Verfahren auch die Eigenwerte und zugehörigen Eigenvektoren schnell findet, die nah beieinander liegen. Dazu wendet man das Lanczos-Verfahren auf die Matrix $(I + \gamma A)^{-1}$ mit einem Shift $\gamma > 0$ an. Die Lanczos-Relation für die transformierte Matrix ist dann durch

$$(I + \gamma A)^{-1} V_m = V_m T_m + \beta_m v_{m+1} e_m^T \quad \text{wobei} \quad V_m e_1 = v \quad \text{und} \quad V_m^H V_m = I$$

gegeben. Definiert man die Funktion

$$f_\gamma^\tau(t) = \varphi((1 - t^{-1})\tau/\gamma) \quad \text{für} \quad t \in (0, 1] \quad \text{und} \quad f_\gamma^\tau(0) = 0,$$

so ist $f_\gamma^\tau((I + \gamma A)^{-1}) = \varphi(-\tau A)$ und man erhält

$$w(\tau) \approx w_m(\tau) = V_m f_\gamma^\tau(T_m) e_1 = V_m \varphi(\tau \widetilde{T}_m) e_1 \quad \text{mit} \quad \widetilde{T}_m = \frac{1}{\gamma}(T_m^{-1} - I). \tag{4.1}$$

Analoge Rechnung kann man auch für die Funktion $\exp(\cdot)$ anstelle von $\varphi(\cdot)$ durchführen. Der Einfachheit halber betrachten wir im Folgenden Approximationen an

$$y(\tau) = \exp(-\tau A)v,$$

für welche man in der Literatur viele Konvergenzaussagen findet. In der Praxis stellt man fest, dass Näherungen an $\varphi(-\tau A)v$ fast genauso schnell konvergieren wie an $\exp(-\tau A)v$.

Um den Vorteil der Transformation des Spektrums einzusehen, betrachten wir ein Maß, welches die Trennung der Eigenwerte beschreibt. Für den zum kleinsten Eigenwert gehörenden invarianten Unterraum ist dieses durch

$$\frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1} \tag{4.2}$$

gegeben, wobei λ_n den größten Eigenwert bezeichnet. Dieses Maß tritt auch in Aussagen über die Konvergenzgeschwindigkeit von Lanczos-Verfahren auf. Für die transformierte Matrix gilt

$$\frac{(1+\gamma\lambda_1)^{-1}-(1+\gamma\lambda_n)^{-1}}{(1+\gamma\lambda_1)^{-1}-(1+\gamma\lambda_2)^{-1}} \leq \frac{(1+\gamma\lambda_1)^{-1}}{(1+\gamma\lambda_1)^{-1}-(1+\gamma\lambda_2)^{-1}} = \frac{1+\gamma\lambda_2}{\gamma(\lambda_2-\lambda_1)}.$$

Beispielsweise beim eindimensionalen Poisson-Problem mit Dirichlet-Randbedingungen ist (4.2) unbeschränkt, wohingegen der Wert bei transformiertem Spektrum durch $(1+\gamma 4\pi^2)/(3\gamma\pi^2)$ beschränkt ist.

Daher liegt die Vermutung nahe, dass das Lanczos-Verfahren durch diese Transformation schneller konvergiert. Natürlich kommt weiterer Rechenaufwand hinzu, da in jedem Schritt ein lineares Gleichungssystem gelöst werden muss. Weiterhin stellt sich die Frage wie der Shift γ zu wählen ist und wie man den Fehler der Approximation abschätzen kann.

4.2 Fehlerabschätzungen und die Wahl des Shifts

Um eine obere Schranke für den Fehler der Approximation (4.1) zu erhalten, schreiben wir die Lanczos-Relation zunächst als

$$V_m f_\gamma^\tau(T_m)e_1 = p((I+\gamma A)^{-1})v = (I+\gamma A)^{-(m-1)}q(A)v$$

mit $p, q \in \Pi_{m-1}$, wobei Π_{m-1} den Raum aller Polynome bis zum Grad $m-1$ bezeichnet. Dies folgt sofort aus ([11], Lemma 3.1 und 3.2) und der Definiton von $f_\gamma^\tau(t)$. Also kann man die Approximation auch als Konstruktion einer rationalen Funktion aus dem Raum

$$\mathcal{R}_i^j = \{p(t)(1+\gamma t)^{-i} | p \in \Pi_j\}$$

auffassen.

Die Aussage von ([11], Lemma 3.1) ist, dass für jede beliebige Matrix A und V_m, H_m resultierend aus dem Lanczos-Verfahren und jedes Polynom $p \in \Pi_{m-1}$

$$p(A)v = V_m p(H_m)e_1 \tag{4.3}$$

gilt, was im Folgenden noch benötigt wird.

Grundlegend für den Beweis des Konvergenzresultates ist

Lemma 3.1 A sei eine beliebige Matrix, V_m, H_m die aus dem Lanczos-Verfahren resultierenden Matrizen und f eine beliebige Funktion, so dass $f(A)$ und $f(H_m)$ definiert sind. Weiterhin sei p ein Polynom vom Grad $\leq m-1$, welches f approximiert und $r_m(z) = f(z) - p(z)$. Dann gilt mit $\|v\| = \beta$

$$f(A)v - \beta V_m f(H_m)e_1 = \beta(r_m(A)v - V_m r_m(H_m)e_1),$$

([11], Theorem 4.1).

Definiert man nun

$$E_i^j(\gamma) = \inf_{r \in \mathcal{R}_i^j} \sup_{t \geq 0} |r(t) - \exp(-t)|,$$

erhält man folgendes Resultat ([19], Lemma 3.1):

Lemma 3.2

Sei μ so gewählt, dass $A - \mu I$ positiv semidefinit ist. Dann gilt

$$||V_m f_\gamma^\tau(T_m)e_1 - \exp(-\tau A)v|| \leq 2\exp(-\tau\mu)E_{m-1}^{m-1}(\widetilde{\gamma}) \quad \text{mit} \quad \widetilde{\gamma} = \frac{\gamma}{\tau(1+\gamma\mu)}.$$

Beweis: Sei $\widetilde{A} = (I + \gamma A)^{-1}$. Da T_m die Matrix aus dem Lanczos-Verfahren mit $\widetilde{A}$ ist, folgt mit $\exp(-\tau A) = f_\gamma^\tau(\widetilde{A})$ und Lemma 3.1

$$\begin{aligned}
& \|V_m f_\gamma^\tau(T_m)e_1 - \exp(-\tau A)v\| \\
= \; & \|V_m f_\gamma^\tau(T_m)e_1 - f_\gamma^\tau(\widetilde{A})v\| \\
= \; & \|V_m r_m(T_m)e_1 - r_m(\widetilde{A})v\| \\
\leq \; & \|r_m(T_m)\| + \|r_m(\widetilde{A})\| \\
= \; & \max_{\lambda \in \sigma(T_m)} |r_m(\lambda)| + \max_{\lambda \in \sigma(\widetilde{A})} |r_m(\lambda)| \\
\leq \; & 2 \sup_{\lambda \in \mathcal{F}(\widetilde{A})} |r_m(\lambda)| \\
\leq \; & 2 \inf_{p \in \Pi_{m-1}} \sup_{t \in (0,(1+\gamma\mu)^{-1}]} |f_\gamma^\tau(t) - p(t)| \\
= \; & 2 \inf_{p \in \Pi_{m-1}} \sup_{t \in (0,1]} |f_\gamma^\tau(\frac{t}{(1+\gamma\mu)}) - p(t)| \\
= \; & 2 \inf_{p \in \Pi_{m-1}} \sup_{t \in (0,1]} |\exp((1 - \frac{1}{t}\frac{\tau(1+\gamma\mu)}{\gamma} - \mu\tau) - p(t)| \\
= \; & 2 \inf_{p \in \Pi_{m-1}} \sup_{t \in (0,1]} |\exp(-\mu\tau)\exp((1 - \frac{1}{t}\frac{\tau(1+\gamma\mu)}{\gamma}) - p(t)| \\
= \; & 2\exp(-\mu\tau)E_{m-1}^{m-1}(\widetilde{\gamma}).
\end{aligned}$$

□

Wichtig ist, dass diese Abschätzung unabhängig von der Norm von A ist. Da alle Eigenwerte von A größer sind als μ, spielt nur der kleinste Eigenwert von A eine Rolle.

Lemma 3.2 zeigt, dass der Fehler klein wird, wenn man $E_{m-1}^{m-1}(\widetilde{\gamma})$ minimiert. Hieraus erhält man einen Wert für $\widetilde{\gamma}$ bzw. γ. Im Folgenden nehmen wir ohne Einschränkung $\tau = 1$ und $\mu = 0$ an woraus $\widetilde{\gamma} = \gamma$ folgt.

j	$E_j^j(\gamma_{\text{opt}})$	γ_{opt}	j	$E_j^j(\gamma_{\text{opt}})$	γ_{opt}
1	$6.7 \cdot 10^{-2}$	$1.73 \cdot 10^{0}$	11	$4.0 \cdot 10^{-6}$	$9.90 \cdot 10^{-2}$
2	$2.0 \cdot 10^{-2}$	$4.93 \cdot 10^{-1}$	12	$1.6 \cdot 10^{-6}$	$1.19 \cdot 10^{-1}$
3	$7.3 \cdot 10^{-3}$	$2.64 \cdot 10^{-1}$	13	$6.1 \cdot 10^{-7}$	$1.00 \cdot 10^{-1}$
4	$3.1 \cdot 10^{-3}$	$1.75 \cdot 10^{-1}$	14	$2.5 \cdot 10^{-7}$	$8.64 \cdot 10^{-2}$
5	$1.4 \cdot 10^{-3}$	$1.30 \cdot 10^{-1}$	15	$1.0 \cdot 10^{-7}$	$7.54 \cdot 10^{-2}$
6	$4.0 \cdot 10^{-4}$	$1.91 \cdot 10^{-1}$	16	$4.0 \cdot 10^{-8}$	$8.67 \cdot 10^{-2}$
7	$1.6 \cdot 10^{-4}$	$1.44 \cdot 10^{-1}$	17	$1.6 \cdot 10^{-8}$	$7.63 \cdot 10^{-2}$
8	$6.5 \cdot 10^{-5}$	$1.90 \cdot 10^{-1}$	18	$6.6 \cdot 10^{-9}$	$6.78 \cdot 10^{-2}$
9	$2.4 \cdot 10^{-5}$	$1.47 \cdot 10^{-1}$	19	$2.7 \cdot 10^{-9}$	$7.62 \cdot 10^{-2}$
10	$9.7 \cdot 10^{-6}$	$1.19 \cdot 10^{-1}$	20	$1.1 \cdot 10^{-9}$	$6.82 \cdot 10^{-2}$

Tabelle 4.1: Numerische Approximation an den optimalen Wert für γ und den zugehörigen Wert $E_j^j(\gamma_{\text{opt}})$, [19].

Näherungen an $E_{m-1}^{m-1}(\gamma)$ können z.b. durch die Methode von Remez gewonnen werden, indem man eine optimale polynomiale Approximation an $f_\gamma^1(t)$ in $[0,1]$ berechnet. Dies führt auf die in Tabelle 4.1 angegebenen Werte für γ bei der angegebenen gewünschten Genauigkeit. Wenn man beispielsweise an einer Genauigkeit von 10^{-5} interessiert ist, wählt man $\gamma = 0.119\tau$. In [19] werden auch andere Wahlen für den Shift γ zitiert, auf welche wir hier nicht eingehen wollen. Sie basieren auf Approximationen an e^{-z} durch rationale Funktionen mit reellen Polen, [1, 12].

Um einen Fehlerschätzer herzuleiten, schreiben wir zunächst die Lanczos-Relation für $(I + \gamma A)^{-1}$ um. Aus

$$(I + \gamma A)^{-1} V_m = V_m T_m + \beta_m v_{m+1} e_m^T$$

folgt

$$V_m T_m^{-1} = (I+\gamma A)V_m + \beta_m \widetilde{v}_{m+1} z_m^T, \qquad \widetilde{v}_{m+1} = (I+\gamma A)v_{m+1}, \qquad z_m = T_m^{-1} e_m.$$

Woraus

$$V_m(T_m^{-1} - I) = \gamma A V_m + \beta_m \widetilde{v}_{m+1} z_m^T$$

und schließlich

$$V_m \widetilde{T}_m = AV_m + \frac{\beta_m}{\gamma} \widetilde{v}_{m+1} z_m^T, \qquad \widetilde{T}_m = \frac{1}{\gamma}(T_m^{-1} - I) \tag{4.4}$$

folgt.

Aus $y_m(t) = V_m \exp(-t\widetilde{T}_m) e_1$ und (4.4) folgt dann

$$y_m'(t) = -V_m \widetilde{T}_m \exp(-t\widetilde{T}_m) e_1 = -A y_m(t) + g_m(t), \qquad y_m(0) = v,$$

mit

$$g_m(t) = -\frac{\beta_m}{\gamma} \widetilde{v}_{m+1} z_m^T \exp(-t\widetilde{T}_m) e_1.$$

Für die exakte Lösung gilt

$$y'(t) = -Ay(t), \qquad y(0) = v$$

und für den Fehler $e_m(t) = y_m(t) - y(t)$ gilt daher

$$e'_m(t) = y'_m(t) - y'(t) = -Ae_m(t) + g_m(t), \qquad e_m(0) = 0.$$

Mit der Variation-der-Konstanten-Formel folgt

$$\begin{aligned} e_m(\tau) &= \int_0^\tau \exp(-(\tau - t)A) g_m(t) dt \\ &= -\frac{\beta_m}{\gamma} \int_0^\tau \exp(-(\tau - t)A) \widetilde{v}_{m+1} z_m^T \exp(-t\widetilde{T}_m) e_1 dt. \end{aligned}$$

Dies kann mit

$$X_m = \int_0^\tau e_m^T (I + \gamma \widetilde{T}_m) \exp(-t\widetilde{T}_m) e_1 \exp(-(\tau - t)A)(I + \gamma A) dt$$

als

$$e_m(\tau) = -\frac{\beta_m}{\gamma} X_m v_{m+1}$$

geschrieben werden. Setzt man

$$\exp(-(\tau - t)A) = I - (\tau - t)A + \frac{1}{2}(\tau - t)^2 A^2 + \ldots$$

ein, so folgt

$$e_m(\tau) = -\frac{\beta_m}{\gamma} \int_0^\tau (I - (\tau - t)A + \frac{1}{2}(\tau - t)^2 A^2 + \ldots) \widetilde{v}_{m+1} z_m^T \exp(-t\widetilde{T}_m) e_1 dt$$

und daher

$$e_m(\tau) = -\frac{\beta_m}{\gamma} \sum_{j=0}^{\infty} z_m^T \phi_{j+1}(-\tau \widetilde{T}_m) e_1 (-A)^j (I + \gamma A) v_{m+1}$$

mit

$$\phi_j(-\tau A) = \frac{1}{(j-1)!} \int_0^\tau (\tau - j)^{j-1} \exp(-tA) dt.$$

Testbeispiele zeigen, dass die Norm des ersten Summanden

$$\|e_m(t)\| \approx \frac{\beta_m}{\gamma} |z_m^T \phi_1(-\tau \widetilde{T}_m) e_1| \ \ \|(I + \gamma A) v_{m+1}\| \tag{4.5}$$

bereits eine gute Schranke für den Fehler ist.

Eine weitere Möglichkeit besteht darin den Fehler durch zwei bereits berechnete Näherungen abzuschätzen. Hierzu nimmt man an, dass für den relativen Fehler $\widetilde{\delta}_m$ im m-ten Schritt

$$\frac{\|y_m(\tau) - y(\tau)\|}{\|y(\tau)\|} = \widetilde{\delta}_m \approx \delta_m = \frac{\|y_m(\tau) - y_{m-1}(\tau)\|}{\|y_m(\tau)\|}$$

gilt. Aus

$$\delta_m \|y(\tau)\| \approx \widetilde{\delta}_m \|y(\tau)\| = \|y_m(\tau) - y(\tau)\| \geq \|y(\tau)\| - \|y_m(\tau)\|$$

folgt dann

$$\|y(\tau)\| \lesssim \frac{1}{1-\delta_m} \|y_m(\tau)\|$$

und daher gilt für den absoluten Fehler

$$\|e_m(\tau)\| = \|y_m(\tau) - y(\tau)\| = \widetilde{\delta}_m \|y(\tau)\| \approx \delta_m \|y(\tau)\| \lesssim \frac{\delta_m}{1-\delta_m} \|y_m(\tau)\|. \quad (4.6)$$

Da (4.6) in den ersten Schritten etwas zu optimistisch sein könnte, kann man das Minimum von (4.6) und der Schranke $\|e_m(\tau)\| \leq 1 + \|y_m(\tau)\|$ wählen.

Um die beiden Fehlerschätzer zu vergleichen, berechnen wir $\varphi(j\tau A)v$ für $j = 1, 2, 3$ und wählen A als die aus der Wärmeleitungsgleichung

$$\frac{\partial u}{\partial t} = \alpha\big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\big), \qquad u(x, y, 0) = u_0(x, y), \quad (4.7)$$

mit $0 \leq x, y \leq 1$ und Neumann-Randbedingungen resultierende Matrix, wenn man die Gleichung mit finiten Differenzen und 50 Gitterpunkten in jede Richtung diskretisiert. Hier ist $\alpha = 1$ gewählt.

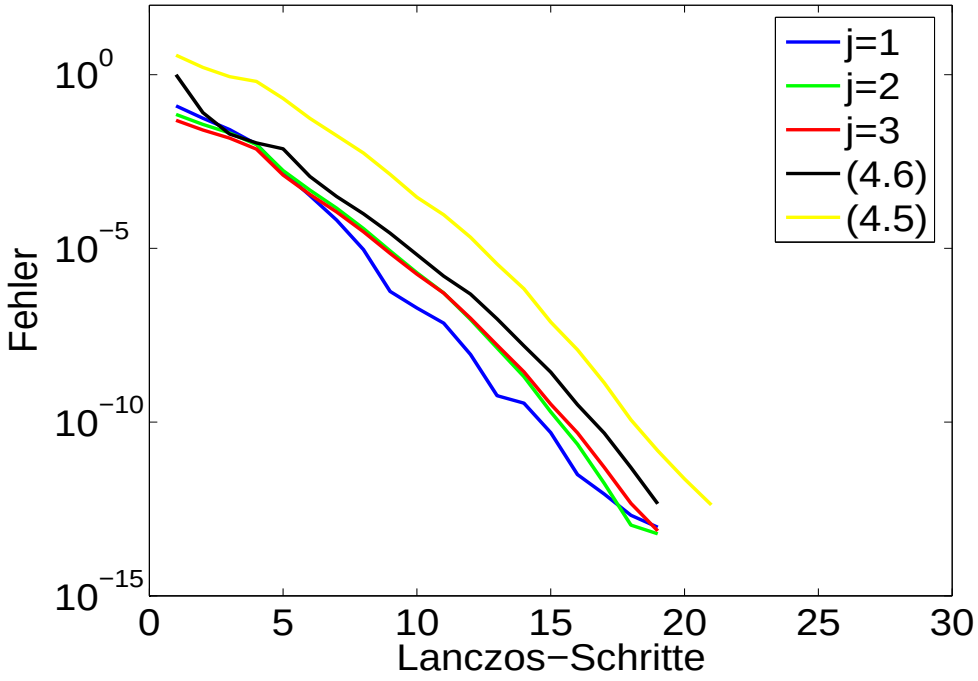

Abb. 4.1: Fehler der vorkonditionierten Lanczos-Approximation für $\tau = 1/10$.

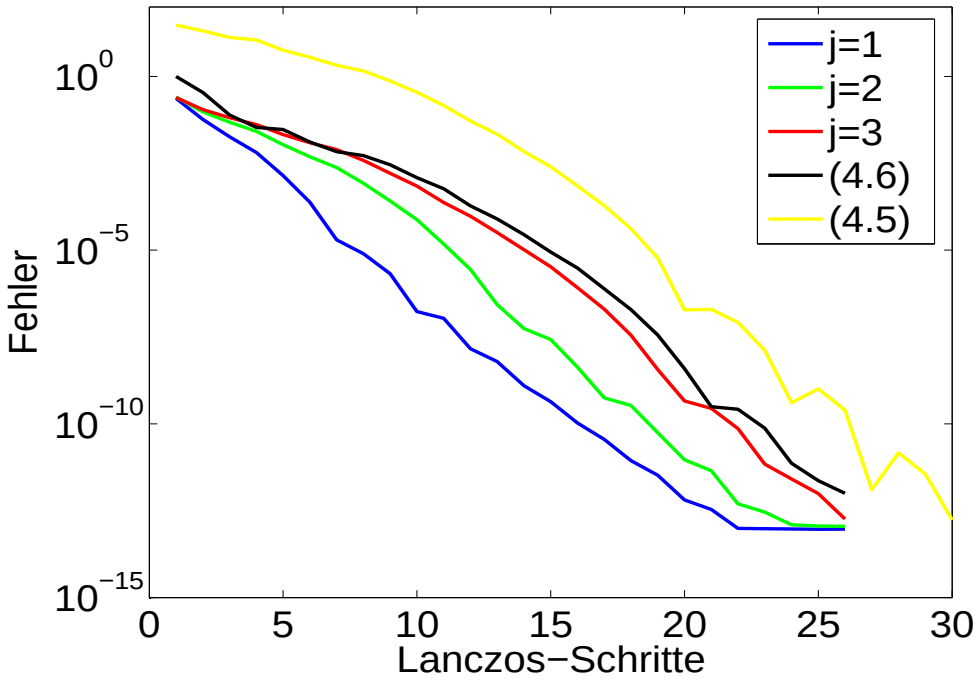

Abb. 4.2: Fehler der vorkonditionierten Lanczos-Approximation für $\tau = 1/100$.

Die beiden Abbildungen zeigen für 2 unterschiedliche Werte von τ den Fehler der Approximation an $\varphi(j\tau A)v$, $j = 1, 2, 3$, wobei der Shift als $\gamma_{opt}\tau$ gewählt

wurde. Sowie die Fehlerschätzer (4.5) und (4.6). Zum einen ist zu sehen, dass in beiden Fällen der Fehler bei $\varphi(3\tau A)v$ am größten ist, daher wird bei beiden Fehlerschätzern der Faktor 3 mit einbezogen. Zum anderen sieht man, dass beide Fehlerschätzer gute obere Schranken für den gesamten Fehler sind, wobei (4.6) eine genauere Näherung liefert.

4.3 Lösen des Gleichungssystems

Durch die Vorkonditionierung ist im j-ten Schritt des Lanczos-Verfahrens ein lineares Gleichungssystem von der Form

$$(I + \gamma A)c_j = v_j$$

zu lösen. Wir betrachten hier zwei Möglichkeiten.

Zum einem lösen wir das Gleichungssystem direkt, wobei eine LU-Zerlegung verwendet wird. Da die Matrizen L und U bei der LU-Zerlegung einer dünnbesetzten Matrix oft prozentual mehr von Null verschiedene Elemente enthalten als die Ausgangsmatrix, ist es sinnvoll die Matrix geeignet zu permutieren bevor man die LU-Zerlegung berechnet. Wird in dem jeweiligen Verfahren mit konstanter Schrittweite gearbeitet und ist die Jacobi-Matrix konstant, so muss man im gesamten Verfahren nur einmal

$$LU = P^T(I + \gamma A)P$$

berechnen. Da der Shift γ von der Schrittweite abhängt, muss bei jedem Aufruf des Lanczos-Verfahrens eine neue Zerlegung berechnet werden sobald die Schrittweite sich ändert.

Zum anderen kann man das Gleichungssystem auch mit einem iterativen Verfahren lösen. Hier betrachten wir das vorkonditionierte cg-Verfahren und verwenden die unvollständige LU-Zerlegung zur Vorkonditionierung. Natürlich muss man dem cg-Verfahren eine Toleranz vorgeben. In [19] findet man eine Strategie wie man ohne Verlust der geforderten Genauigkeit diese Toleranz anheben kann, um schneller an eine Lösung zu gelangen. Dies wird im folgenden kurz erläutert.

Wir sind an einer Lösung $w_m(\tau)$ mit $\|w(\tau) - w_m(\tau)\| \leq \epsilon$ interessiert. Um das Gleichungssystem zu lösen, geben wir dem Löser im j-ten Schritt des Lanczos-Verfahrens eine Toleranz η_j vor. Wenn c_j die Näherung an die Lösung des Gleichungsystems ist, gilt

$$\|r_j\| \leq \eta_j \qquad \text{mit} \qquad r_j = v_j - (I + \gamma A)c_j.$$

Die Idee ist, dass es in späteren Schritten des Lanczos-Verfahren reicht eine geringere Genauigkeit η_j zu fordern, um trotzdem eine gewünschte Genauigkeit

von ϵ zu erreichen. Dies liegt daran, dass die ersten Schritte des Lanczos-Verfahrens die entscheidenden sind.

Um dies einzusehen, betrachten wir die Lanczos-Relation unter Einbeziehung des Fehlers, der durch Lösen des Gleichungssystem entsteht. Definiert man die Matrix F_m durch $F_m e_j = r_j$, d.h. die j-te Spalte der Matrix F_m enthält das Residuum des linearen Gleichungssystem, welches im j-ten Schritt des Lanczos-Verfahren gelöst wird, erhält man die Relation

$$AV_m = V_m \widetilde{T}_m - \frac{\beta_m}{\gamma} \widetilde{v}_{m+1} z_m^T - \frac{1}{\gamma} F_m T_m^{-1}.$$

Diese ist ähnlich zu (4.4), wobei zu beachten ist, dass sowohl V_m als auch T_m nicht die gleichen wie in der fehlerfreien Relation sind. Es ist nicht einmal garantiert, dass die Matrix V_m orthogonal ist. Vernachlässigt man Rundungsfehler und führt die gleiche Rechnung wie im vorigen Abschnitt durch, erhält man, unter Verwendung der gleichen Notation, aus dieser Relation

$$\|e_m(\tau)\| \leq \frac{\beta_m}{\gamma} \|X_m v_{m+1}\| + \frac{1}{\gamma} \| \int_0^\tau \exp(-(\tau - t)A) F_m T_m^{-1} \exp(-t\widetilde{T}_m) e_1 dt \|.$$

Unter der Voraussetzung $\eta_j = \epsilon$ folgt mit der Cauchy-Schwarz-Ungleichung die grobe Abschätzung

$$\frac{1}{\gamma} \| \int_0^\tau \exp(-(\tau - t)A) F_m T_m^{-1} \exp(-t\widetilde{T}_m) e_1 dt \| \leq \|T_m^{-1} e_1\| \frac{\sqrt{m}\tau}{\gamma} \epsilon. \qquad (4.8)$$

Dies folgt zum einen aus

$$\|F_m\| = \sup_{x \neq 0} \frac{\|F_m x\|}{\|x\|} = \max_{\|x\|=1} \|F_m x\| = \max_{\|x\|=1} \|F_m e_1 x_1 + \ldots + F_m e_m x_m\| \leq \sqrt{m}\epsilon,$$

wobei eingeht, dass aus $\eta_j = \epsilon$, $\|F_m e_j\| = \|r_j\| \leq \epsilon$ folgt. Zum anderen daraus, dass für eine symmetrische Matrix A eine orthogonale Matrix Q mit $A = Q\Lambda Q^T$, wobei Λ die Eigenwerte als Diagonalelemente enthält, existiert und daher

$$\| \exp(A)\| = \|Q \exp(\Lambda) Q^T\| = \| \exp(\Lambda)\| = \max_{\lambda \in \sigma(A)} |\exp(\lambda)|$$

gilt. Dies motiviert die $\exp(\cdot)$ Terme in (4.8) grob durch 1 abzuschätzen.

Daher gilt

$$\|e_m(\tau)\| \leq \frac{\beta_m}{\gamma} \|X_m v_{m+1}\| + \|(I + \gamma\widetilde{T}_m) e_1\| \frac{\sqrt{m}\tau}{\gamma} \epsilon.$$

Der Term $z_m^T \phi_{j+1}(-\tau\widetilde{T}_m) e_1$ aus (4.5) nimmt, aufgefasst als Funktion der Iterationsschritte m, mit wachsendem m ab. Woraus folgt, dass die Norm $\|X_m v_{m+1}\|$

viel kleiner als ϵ wird bevor sie stagniert. Dies hat zur Folge, dass die Genauigkeit vom zweiten Term abhängt. Für weitere Details sei auf [19] verwiesen.

Approximiert man Matrix-Vektor-Produkte mit Krylov-Methoden ist es wichtig, in den ersten Iterationsschritten die gewünschte Genauigkeit zu fordern, aber in späteren Schritten genügt eine geringere Genauigkeit, [3, 14, 20].

In [19] wird gezeigt, dass diese Strategie auch auf den hier vorliegenden Fall angewandt werden kann, und es wird die folgende *relaxation strategie* vorgeschlagen:

$$\eta_j = \frac{\epsilon}{\|e_{j-1}(\tau)\| + \epsilon} \tag{4.9}$$

4.4 Beispiele

Als Beispiel betrachten wir wieder die Wärmeleitungsgleichung (4.7) mit $\alpha = 0.5$ und 50 Gitterpunkten in jede Richtung. Hierzu berechnen wir zum einen $\varphi(j\tau A)v$ für $j = 1, 2, 3$, um den gesamten Verlauf in einem Lanczos-Schritt beobachten zu können. Des Weiteren werden unterschiedliche Werte für τ gewählt. Zum anderen lösen wir die Gleichung mit dem Integrator *exp*4, um die Auswirkung der *relaxation* auf den gesamten Fehler und die Anzahl der Iterationen zu vergleichen.

Die Abbildungen $4.3 - 4.6$ zeigen die Ergebnisse. Hierbei sind die Fehler der drei zu berechnenden Vektoren in unterschiedlichen Farben dargestellt, $\varphi(\tau A)v$ in blau, $\varphi(2\tau A)v$ in grün und $\varphi(3\tau A)v$ in rot. Die durchgezogenen Linien gehen aus der Berechnung ohne die *relaxation* hervor und die gestrichelten aus der mit *relaxation*. Für beide ist zusätzlich der Fehlerschätzer mit (o) gekennzeichnet und die beim vorkonditioniertem cg-Verfahren verwendete Toleranz mit (+), wobei einmal $\eta_j = \varepsilon$ und einmal η_j wie in (4.9) verwendet wurde.

Löst man die Gleichung mit *exp*4 und fordert eine Genauigkeit von 10^{-6} ergeben sich folgende Werte:

ohne relaxation		mit relaxation	
Fehler	Iterationen	Fehler	Iterationen
$7.2 \cdot 10^{-8}$	3475	$6.8 \cdot 10^{-8}$	2132

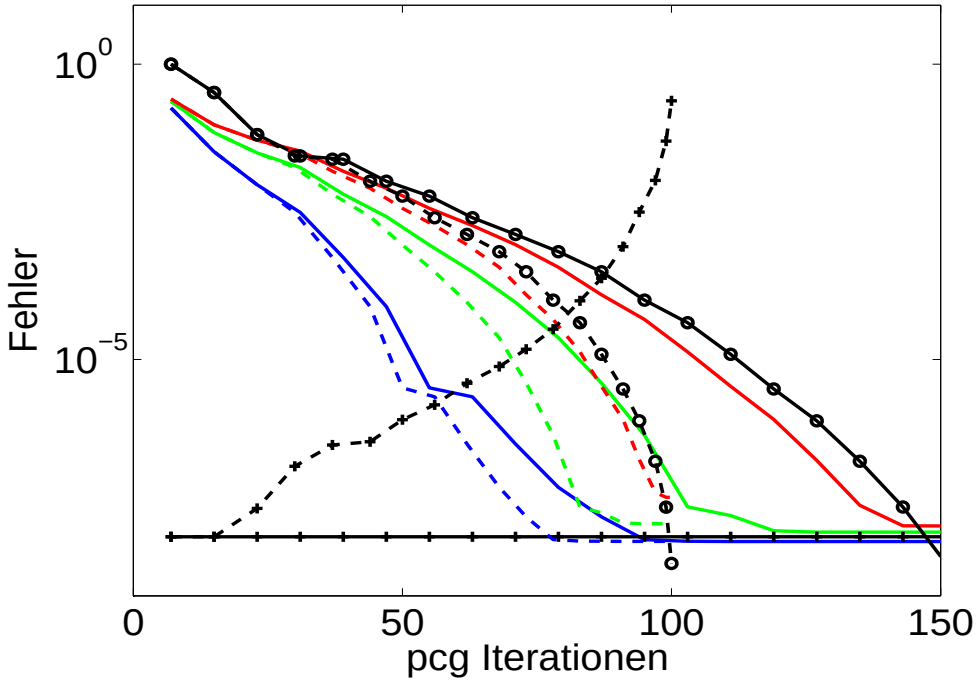

Abb. 4.3: Auswirkungen der *relaxation strategie* für $\tau = 1/100$.

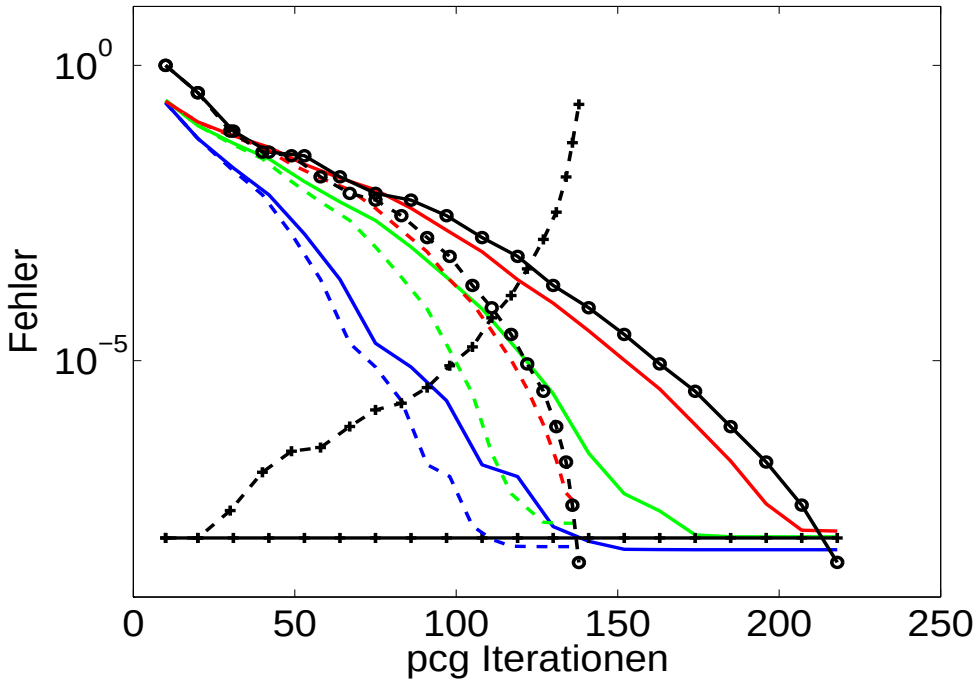

Abb. 4.4: Auswirkungen der *relaxation strategie* für $\tau = 1/50$.

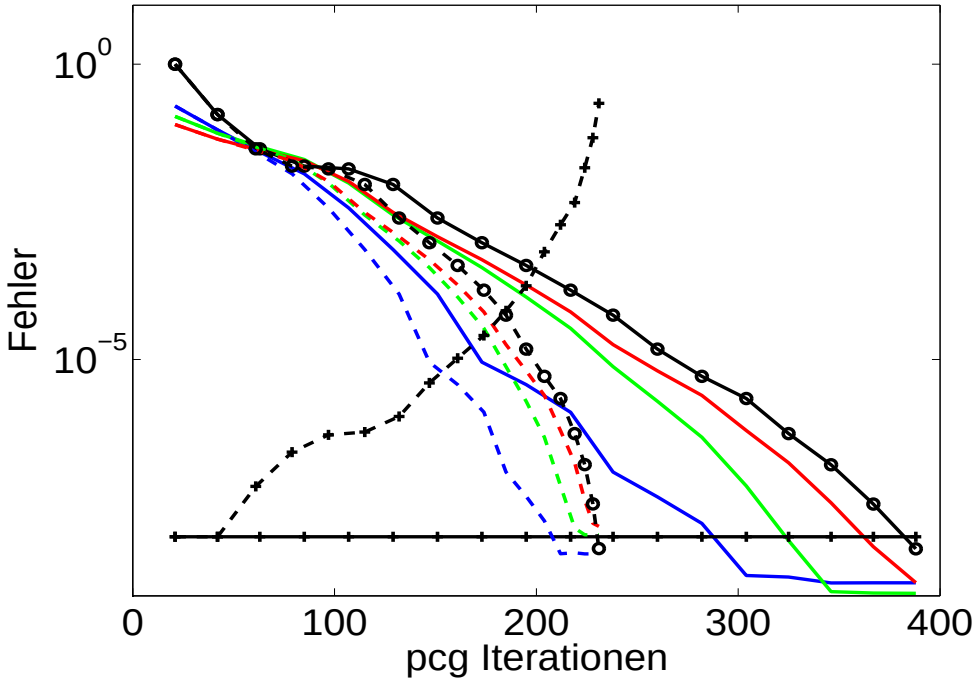

Abb. 4.5: Auswirkungen der *relaxation strategie* für $\tau = 1/10$.

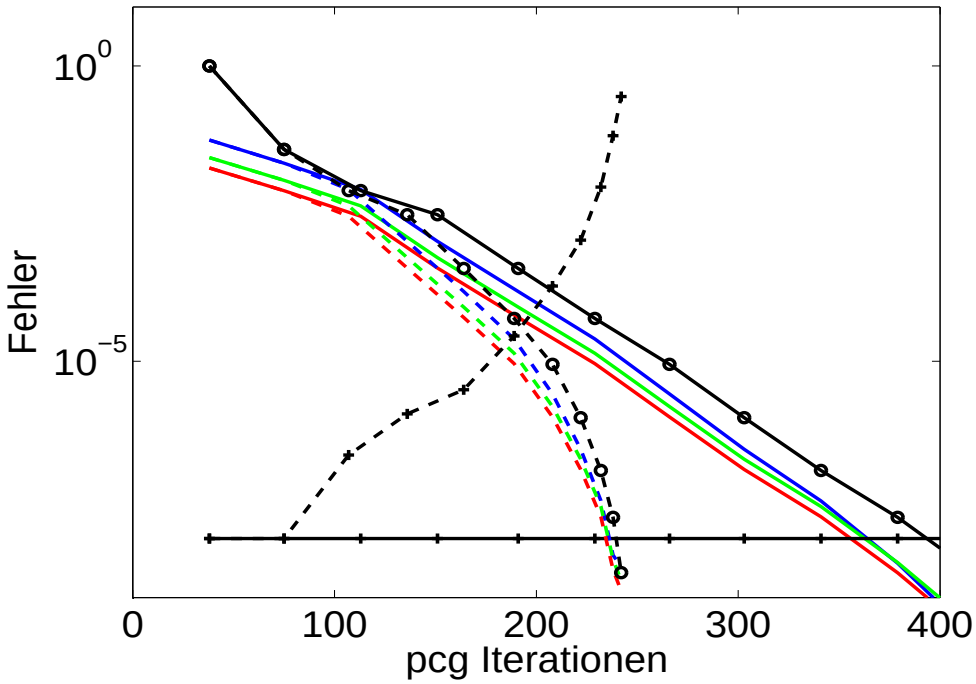

Abb. 4.6: Auswirkungen der *relaxation strategie* für $\tau = 1/2$.

Die Abbildungen zeigen, dass die gewünschte Genauigkeit bei beiden Strategien erreicht wird und bei der *relaxation strategie* durch das Anheben der

Toleranz Iterationen eingespart werden. Weiterhin beobachtet man eine ansteigende Anzahl der Iterationen für wachsendes τ. Dies liegt daran, dass die Norm der Matrix τA für größere τ größer ist.

Bei den numerischen Vergleichen in Kapitel 5 wird das lineare Gleichungssystem mittels der LU-Zerlegung gelöst, da sich dies als effizienter herausgestellt hat. Das Lösen des Gleichungssystems mit einem iterativen Löser kann, unter Verwendung eines effizienteren Verfahrens als des cg-Verfahrens und der *relaxation strategie*, auch von Vorteil sein.

Eine weitere gute Eigenschaft des vorkonditionierten Lanczos-Prozess ist, dass eine Verfeinerung der Ortsdiskretisierung keinen großen Einfluß auf die Anzahl der benötigten Lanczos-Schritte hat. In [19] wird dies am Beispiel der zweidimensionalen Wärmeleitungsgleichung gezeigt. Hier wird die Differentialgleichung

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{1}{1+u^2}, \qquad u(x,0) = u_0(x),$$

mit $0 \leq x \leq 1$ für $0 \leq t \leq 1$ und homogenen Dirichlet-Randbedingungen betrachtet. In Tabelle 4.2 sind die Anzahl der Gitterpunkte und die der Lanczos-Schritte aufgeführt. Die geforderte Genauigkeit von 10^{-6} wurde immer erreicht.

N	Lanczos Schritte
100	1498
200	1578
400	1673
800	1625
1600	1697
3200	1764
6400	1832
12800	1916
25600	2008
51200	2085
102400	2201
204800	3532

Tabelle 4.2: Anzahl der Gitterpunkte und der Lanczos-Schritte.

5 Numerische Vergleiche

In diesem Abschnitt werden die vorgestellten Verfahren miteinander verglichen. Alle Beispiele wurden in Matlab mit einem 1,86 GHz Prozessor und 1GB Arbeitsspeicher berechnet. Als erstes Beispiel betrachten wir die Differentialgleichung

$$\frac{\partial u}{\partial t}(x,t) = \frac{\partial^2 u}{\partial x^2}(x,t) + \frac{1}{1+u(x,t)^2}, \qquad u(x,0) = u_0(x), \tag{5.1}$$

mit $0 \leq x \leq 1$ für $0 \leq t \leq 1$ und homogenen Dirichlet-Randbedingungen. Um den Fehler berechnen zu können, wurde eine Referenzlösung mit dem Verfahren *radau*5 und einer absoluten und relativen Toleranz von 10^{-12} berechnet.

Wir lösen die Gleichung mit den exponentiellen Integratoren *exp*4 und dem Verfahren von Krogstad (*krog*), dem Matlab Löser *ode*15*s* und *rkc*. Bei den beiden exponentiellen Integratoren wurde die Gleichung mit und ohne Vorkonditionierung gelöst. In den Tabellen 5.1-5.6 sind die geforderten Toleranzen, der Fehler gemessen in der Maximumsnorm, die Rechenzeit, die Anzahl der Zeitschritte von $t = 0$ bis $t = 1$ und bei *exp*4 bzw. *krog*, die Anzahl der Krylov-Schritte aufgelistet, sowie die durchschnittliche Anzahl der Krylov-Schritte pro Zeitschritt, wenn man Gleichung (5.1) mit 400 Gitterpunkten bezüglich des Ortes diskretisiert. Das beim vorkonditioniertem Lanczos-Prozess auftretende tridiagonale Gleichungssystem wurde mit Hilfe der LU-Zerlegung direkt gelöst, was auch bei den weiteren Beispielen der Fall ist. Zur Veranschaulichung ist in Abbildung 5.1 der Fehler, gemessen in der Maximumsnorm, als Funktion der Rechenzeit aufgetragen. In Abbildung 5.2 sieht man eine Vergrößerung von Abbildung 5.1.

Wie in Kapitel 2 erläutert, verwendet *exp*4 eine Schrittweitensteuerung. Da das Produkt $\varphi(-\tau A)v$ von der Schrittweite abhängt, muss bei der vorkonditionierten Variante in jedem Zeitschritt eine neue LU-Zerlegung berechnet werden. Die hier verwendete Implementierung des Verfahrens von Krogstad arbeitet mit einer konstanten Schrittweite und daher muss nur eine einzige LU-Zerlegung berechnet werden. Die Schrittweite bei dem Verfahren von Krogstad wurde hier so gewählt, dass die geforderte Genauigkeit erreicht wurde. Wie man in den Tabellen 5.3 und 5.4 sieht, reicht für geringe Genauigkeiten ein einziger Zeitschritt aus.

In Tabelle 5.1 sieht man, dass *exp*4 die geforderte Genauigkeit teilweise nicht erreicht. Dies liegt daran, dass hier die maximale Anzahl der Krylov-Schritte pro Zeitschritt für alle geforderten Genauigkeiten fest auf 100 gesetzt wurde. Durch andere Wahlen der maximalen Krylov-Schritte lässt sich die Genauigkeit steuern. In einigen Fällen ist es von Vorteil die Anzahl zu beschränken und in anderen diese hochzusetzen, denn wenn in einem Durchlauf des Krylov-Verfahrens die maximale Anzahl erreicht wurde, wird die Schrittweite auto-

matisch verkleinert und der Zeitschritt, mit geringerem Aufwand, wiederholt.

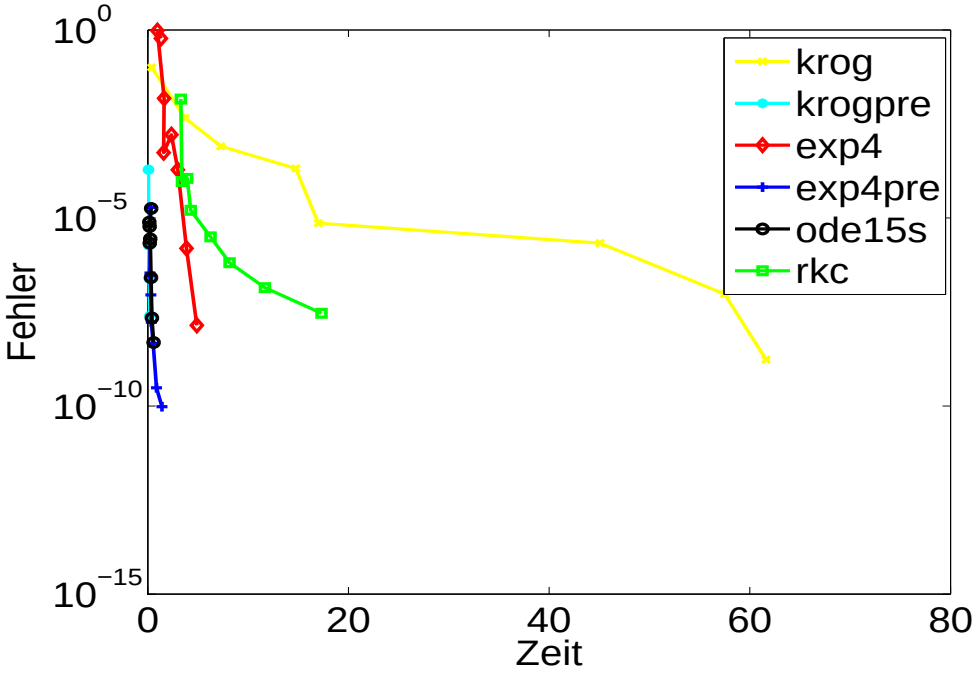

Abb. 5.1: Fehler als Funktion der Rechenzeit für Gleichung (5.1).

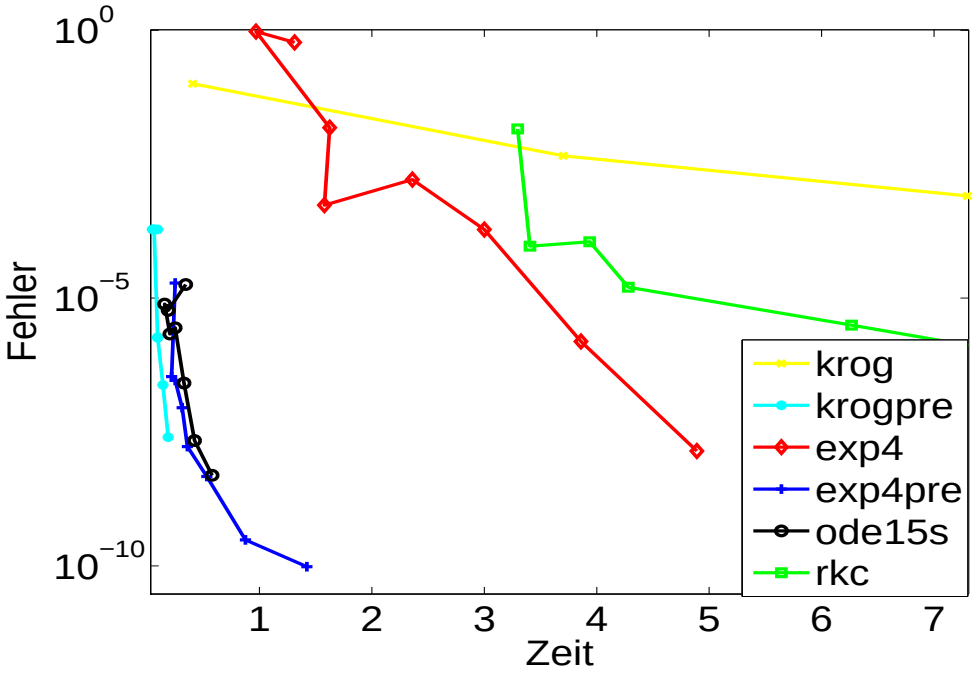

Abb. 5.2: Vergrößerung von Abb. 5.1.

In den Tabellen 5.1 und 5.2, sowie 5.3 und 5.4 sieht man, dass sich durch die Vorkonditionierung die Anzahl der Krylov-Schritte stark reduziert. Dadurch spart man, trotz Berechnung der LU-Zerlegung, Rechenzeit ein.

Beim Verfahren von Krogstad wurden, wie man in Tabelle 5.3 sieht, sehr viele Krylov-Schritte pro Zeitschritt benötigt, was zu einer langen Rechenzeit führt. In einem Zeitschritt werden 4 Krylov-Räume konstruiert, d.h. wenn 200 Krylov-Schritte gemacht werden, hat jeder durschnittlich die Dimension 50, was bei einem Problem der Dimension 400 nicht zufriedenstellend ist. Daher ist es hier nicht sinnvoll das Produkt der φ-Funktionen mit einem Vektor durch Krylov-Methoden zu approximieren.

Abbildung 5.1 und 5.2 veranschaulichen, dass bei diesem Beispiel nur die beiden vorkonditionierten exponentiellen Integratoren mit dem Matlab Löser $ode15s$ konkurieren können. rkc kann den Vorteil ohne die Jacobi-Matrix zu arbeiten bei diesem eindimensionalen Beispiel nicht ausnutzen.

Toleranz	Fehler	Zeit in s	Zeitschritte	Krylov-Schritte	$\frac{\text{Krylov-Schritte}}{\text{Zeitschritte}}$
10^{-1}	$6.00 \cdot 10^{-1}$	1.31	49	481	9.8
10^{-2}	$9.53 \cdot 10^{-1}$	0.97	47	563	12.0
10^{-3}	$1.52 \cdot 10^{-2}$	1.63	78	1150	14.7
10^{-4}	$5.44 \cdot 10^{-4}$	1.58	79	1038	13.1
10^{-5}	$1.60 \cdot 10^{-3}$	2.36	81	2302	28.4
10^{-6}	$1.91 \cdot 10^{-4}$	3.00	83	3353	40.4
10^{-7}	$1.55 \cdot 10^{-6}$	3.86	82	4495	54.8
10^{-8}	$1.40 \cdot 10^{-8}$	4.89	108	5945	55.0

Tabelle 5.1: Ergebnisse der Lösung von Gleichung (5.1) mit $exp4$

Toleranz	Fehler	Zeit in s	Zeitschritte	Krylov-Schritte	$\frac{\text{Krylov-Schritte}}{\text{Zeitschritte}}$
10^{-1}	$1.91 \cdot 10^{-5}$	0.25	5	18	3.6
10^{-2}	$3.42 \cdot 10^{-7}$	0.22	6	37	6.2
10^{-3}	$2.91 \cdot 10^{-7}$	0.25	7	58	8.3
10^{-4}	$9.00 \cdot 10^{-8}$	0.31	9	88	9.8
10^{-5}	$1.70 \cdot 10^{-8}$	0.36	11	147	13.4
10^{-6}	$4.66 \cdot 10^{-9}$	0.53	14	252	18.0
10^{-7}	$3.05 \cdot 10^{-10}$	0.88	20	488	24.4
10^{-8}	$9.61 \cdot 10^{-11}$	1.42	30	885	29.5

Tabelle 5.2: Ergebnisse der Lösung von Gleichung (5.1) mit $exp4$ und Vorkonditionierung

Toleranz	Fehler	Zeit in s	Zeitschritte	Krylov-Schritte	$\frac{\text{Krylov-Schritte}}{\text{Zeitschritte}}$
10^{-1}	$1.01 \cdot 10^{-1}$	0.41	1	21	21
10^{-2}	$4.55 \cdot 10^{-3}$	3.70	1	160	160
10^{-3}	$8.11 \cdot 10^{-4}$	7.30	1	195	195
10^{-4}	$2.07 \cdot 10^{-4}$	14.75	1	384	384
10^{-5}	$7.25 \cdot 10^{-6}$	17.00	2	403	201.5
10^{-6}	$2.13 \cdot 10^{-6}$	45.06	5	1186	237.2
10^{-7}	$9.57 \cdot 10^{-8}$	57.48	7	1656	236.6
10^{-8}	$1.70 \cdot 10^{-9}$	61.63	10	1975	197.5

Tabelle 5.3: Ergebnisse der Lösung von Gleichung (5.1) mit *krog* und vorgegebener Schrittweite

Toleranz	Fehler	Zeit in s	Zeitschritte	Krylov-Schritte	$\frac{\text{Krylov-Schritte}}{\text{Zeitschritte}}$
10^{-1}	$1.92 \cdot 10^{-4}$	0.09	1	6	6
10^{-2}	$1.92 \cdot 10^{-4}$	0.06	1	7	7
10^{-3}	$1.92 \cdot 10^{-4}$	0.05	1	9	9
10^{-4}	$1.92 \cdot 10^{-4}$	0.06	1	9	9
10^{-5}	$1.82 \cdot 10^{-6}$	0.09	2	19	9.5
10^{-6}	$1.86 \cdot 10^{-6}$	0.09	2	24	12
10^{-7}	$2.39 \cdot 10^{-7}$	0.14	3	38	12.7
10^{-8}	$2.53 \cdot 10^{-8}$	0.19	5	67	13.4

Tabelle 5.4: Ergebnisse der Lösung von Gleichung (5.1) mit *krog*, vorgegebener Schrittweite und Vorkonditionierung

Toleranz	Fehler	Zeit in s	Zeitschritte
10^{-1}	$1.79 \cdot 10^{-5}$	0.34	16
10^{-2}	$5.90 \cdot 10^{-6}$	0.19	20
10^{-3}	$7.74 \cdot 10^{-6}$	0.16	28
10^{-4}	$2.13 \cdot 10^{-6}$	0.20	43
10^{-5}	$2.78 \cdot 10^{-6}$	0.25	66
10^{-6}	$2.59 \cdot 10^{-7}$	0.33	101
10^{-7}	$2.19 \cdot 10^{-8}$	0.42	145
10^{-8}	$4.90 \cdot 10^{-9}$	0.58	204

Tabelle 5.5: Ergebnisse der Lösung von Gleichung (5.1) mit *ode*15*s*

Toleranz	Fehler	Zeit in s	Zeitschritte
10^{-1}	$1.45 \cdot 10^{-2}$	3.30	8
10^{-2}	$9.36 \cdot 10^{-5}$	3.40	13
10^{-3}	$1.13 \cdot 10^{-4}$	3.94	21
10^{-4}	$1.60 \cdot 10^{-5}$	4.28	29
10^{-5}	$3.14 \cdot 10^{-6}$	6.27	52
10^{-6}	$6.47 \cdot 10^{-7}$	8.16	101
10^{-7}	$1.40 \cdot 10^{-7}$	11.67	211
10^{-8}	$2.93 \cdot 10^{-8}$	17.31	450

Tabelle 5.6: Ergebnisse der Lösung von Gleichung (5.1) mit *rkc*

Als nächstes Beispiel betrachten wir das semilineare parabolische Problem

$$\frac{\partial u}{\partial t}(x,y,z,t) = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)(x,y,z,t) + g(x,y,z,t), \tag{5.2}$$

für $0 \leq x, y, z \leq 1$, $0 \leq t \leq 1$ und homogenen Dirichlet-Randbedingungen. Dabei ist g so gewählt, dass die exakte Lösung durch

$$u(x,y,z,t) = x(1-x)y(1-y)z(1-z)e^t$$

gegeben ist. Die Lösung u ist ein Polynom 2. Grades und es tritt kein Fehler in der Ortsdiskretisierung auf, da der Diskretisierungsfehler von der dritten Ableitung der Lösung abhängt. Als Anfangswert wurde $u(x,y,z,0)$ gewählt.

Wir lösen die Gleichung mit dem exponentiellen Integrator von Krogstad, sowie dem Matlab Löser *ode*15*s* und *rkc*. Dazu ist der Laplace-Operator in (5.2) mit 20 Punkten in jede Richtung durch einen zentralen Differenzenquotienten zweiter Ordnung diskretisiert, woraus sich ein System von 8000 gewöhnlichen Differentialgleichungen ergibt. Die Jacobi-Matrix ist dann durch $\triangle x^{-2}(1,\ldots,1,\ldots,1,\ldots,-6,\ldots,1,\ldots,1,\ldots,1)$ gegeben, wobei $''\ldots''$ für Nullen steht. Mit dem Satz von Gershgorin folgt, dass alle Eigenwerte in dem Intervall $[\frac{-4}{\triangle x^2+\triangle y^2+\triangle z^2}, 0]$ liegen, wobei hier $\triangle x = \triangle y = \triangle z = \frac{1}{20+1}$. Hier wird *exp*4 nicht betrachtet, da das Verfahren für autonome Probleme ist und die Funktion g von t abhängt.

In Abbildung 5.3 ist der Fehler, gemessen in der Maximumsnorm, als Funktion der Rechenzeit aufgetragen. In den Tabellen 5.7-5.10 findet man die geforderten Toleranzen, den Fehler, die Rechenzeit in Sekunden, die Anzahl der Zeitschritte und bei *krog* die Anzahl der Krylov-Schritte, sowie die durchschnittliche Anzahl der Krylov-Schritte pro Zeitschritt.

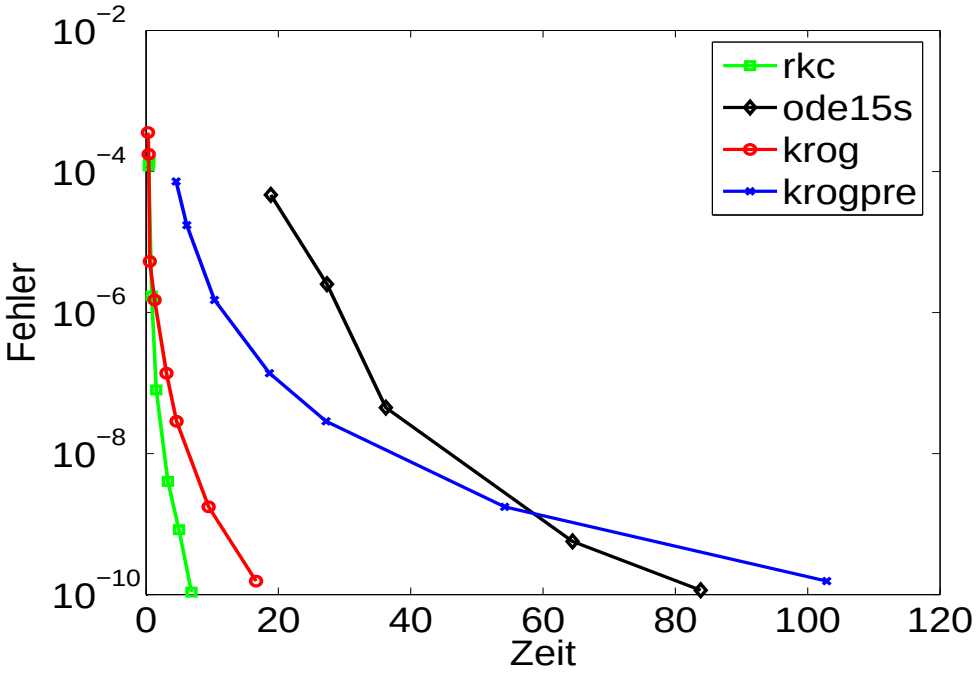

Abb. 5.3: Fehler als Funktion der Rechenzeit für Gleichung (5.2)

Sowohl das Verfahren von Krogstad, als auch *rkc* benötigen weniger Rechenzeit als der Matlab Löser *ode*15*s*. Hier kann *rkc* deutlich den Vorteil nutzen, dass die Jacobi-Matrix im Verfahren nicht explizit verwendet wird. Bei dem Verfahren von Krogstad wurde die Schrittweite wieder so gewählt, dass die geforderte Toleranz erreicht wurde und man sieht wieder, dass nur wenige Zeitschritte benötigt werden. Die vorkonditionierte Variante benötigt hier mehr Rechenzeit als die Variante mit einfachem Lanczos-Verfahren. Dies liegt daran, dass die Anzahl der Krylov-Schritte nicht so stark reduziert wird wie in Beispiel (5.1). Es wird zwar nur eine einzige *LU*-Zerlegung berechnet, was bei diesem Beispiel ungefähr 3.5 Sekunden dauert, aber es muss in jedem Krylov-Schritt ein Gleichungssystem gelöst werden, was hier ungefähr 0.1 Sekunden dauert.

Toleranz	Fehler	Zeit in s	Zeitschritte	Krylov-Schritte	$\frac{\text{Krylov-Schritte}}{\text{Zeitschritte}}$
10^{-1}	$3.57 \cdot 10^{-4}$	0.28	1	4	4
10^{-3}	$1.75 \cdot 10^{-4}$	0.41	2	22	11
10^{-5}	$5.32 \cdot 10^{-6}$	0.51	3	48	16
10^{-6}	$1.51 \cdot 10^{-6}$	1.30	5	125	25
10^{-7}	$1.38 \cdot 10^{-7}$	3.09	10	269	26.9
10^{-8}	$2.86 \cdot 10^{-8}$	4.61	15	372	24.8
10^{-9}	$1.76 \cdot 10^{-9}$	9.45	30	742	24.7
10^{-10}	$1.55 \cdot 10^{-10}$	16.63	55	1319	24.0

Tabelle 5.7: Ergebnisse der Lösung von Gleichung (5.2) mit *krog* und vorgegebener Schrittweite

Toleranz	Fehler	Zeit in s	Zeitschritte	Krylov-Schritte	$\frac{\text{Krylov-Schritte}}{\text{Zeitschritte}}$
10^{-1}	$7.21 \cdot 10^{-5}$	4.53	1	4	4
10^{-5}	$1.73 \cdot 10^{-5}$	6.16	2	14	7
10^{-6}	$1.51 \cdot 10^{-6}$	10.34	5	39	9.8
10^{-7}	$1.38 \cdot 10^{-7}$	18.66	10	94	9.4
10^{-8}	$2.86 \cdot 10^{-8}$	27.20	15	151	10.1
10^{-9}	$1.76 \cdot 10^{-9}$	54.20	30	334	11.1
10^{-10}	$1.55 \cdot 10^{-10}$	102.88	55	670	12.2

Tabelle 5.8: Ergebnisse der Lösung von Gleichung (5.2) mit *krog*, vorgegebener Schrittweite und Vorkonditionierung

Toleranz	Fehler	Zeit in s	Zeitschritte
10^{-1}	$4.66 \cdot 10^{-5}$	18.84	10
10^{-2}	$2.53 \cdot 10^{-6}$	27.34	10
10^{-4}	$4.48 \cdot 10^{-8}$	36.23	11
10^{-7}	$5.68 \cdot 10^{-10}$	64.45	24
10^{-9}	$1.15 \cdot 10^{-10}$	83.83	37

Tabelle 5.9: Ergebnisse der Lösung von Gleichung (5.2) mit $ode15s$

Toleranz	Fehler	Zeit in s	Zeitschritte
10^{-3}	$1.37 \cdot 10^{-4}$	0.38	3
10^{-3}	$1.20 \cdot 10^{-4}$	0.41	3
10^{-5}	$1.71 \cdot 10^{-6}$	0.84	11
10^{-7}	$8.00 \cdot 10^{-8}$	1.53	34
10^{-9}	$4.03 \cdot 10^{-9}$	3.28	142
10^{-10}	$8.32 \cdot 10^{-10}$	4.97	299
10^{-11}	$1.08 \cdot 10^{-10}$	6.86	590

Tabelle 5.10: Ergebnisse der Lösung von Gleichung (5.2) mit *rkc*

Als letztes Beispiel betrachten wir Gleichung (5.1) in 3 Ortsvariablen

$$\frac{\partial u}{\partial t}(x,y,z,t) = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)(x,y,z,t) + \frac{1}{1+u(x,y,z,t)^2}, \qquad (5.3)$$

mit $0 \leq x, y, z \leq 1$ für $0 \leq t \leq 1$ und homogenen Dirichlet-Randbedingungen. Um den Fehler berechnen zu können, wurde wieder eine Referenzlösung mit dem Verfahren $radau5$ berechnet.

Die Gleichung wurde mit den gleichen Verfahren wie das eindimensionale Beispiel (5.1) gelöst und der Laplace-Operator wie im vorigen Beispiel diskretisiert. Die Ergebnisse findet man in den Tabellen 5.11-5.16 und eine graphische Darstellung in den Abbildungen 5.4 und 5.5.

Wie auch bei Beispiel (5.2) benötigt der Matlab Löser *ode*15*s* die meiste Rechenzeit, um die Gleichung zu lösen. Dies ist darauf zurückzuführen, dass wie auch im vorigen Beispiel Gleichungssysteme mit der 8000×8000 Jacobi-Matrix zu lösen sind. Da bei den vorkonditionierten exponentiellen Integratoren, wie in Abschnitt 4 beschrieben, eine Permutation in der *LU*-Zerlegung verwendet wurde, wurde auch hier eine Variante von *ode*15*s* getestet, die mit einer Umsortierung arbeitet, aber dadurch hat sich die Rechenzeit nicht verbessert. Dennoch ist zu vermuten, dass sich dies auch für *ode*15*s* in größeren Dimensionen auszahlen würde. In Tabelle 5.15 sieht man, dass das Verfahren bereits bei geringen geforderten Genauigkeiten hohe Genauigkeiten erzielt. Es sind auch nur diese angegeben, da höhere geforderte Genauigkeiten teilweise geringere oder unwesentlich bessere Genauigkeiten liefern.

*exp*4*pre* erreicht genau wie *ode*15*s* bereits bei geringen geforderten Genauigkeiten gute Approximationen. Daher sind hier auch nur diese angegeben. Vergleicht man in den Tabellen 5.11 und 5.12 die Anzahl der Krylov-Schritte und die der Zeitschritte, stellt man fest, dass hier nichts eingespart wird. Da in jedem Zeitschritt die *LU*-Zerlegung berechnet werden muss und in jedem Krylov-Schritt ein Gleichungssystem gelöst wird, benötigt die vorkonditionierte Variante viel mehr Rechenzeit.

In den Tabellen 5.13 und 5.14 sieht man, dass bei *krog* durch die Vorkonditionierung viele Krylov-Schritte eingespart werden können, aber trotzdem ist die Variante ohne Vorkonditionierung schneller, da die Berechnung der *LU*-Zerlegung und das Lösen des Gleichungssystems den Rechenaufwand dominieren. Auch *krogpre* erzielt höhere Genauigkeiten als gefordert. Hier wurde wieder die Schrittweite dann verkleinert, wenn die geforderte Genauigkeit mit einer größeren Schrittweite nicht erreicht wurde.

Für geringe geforderte Genauigkeiten liefert auch *rkc* schnell gute Approximationen an die Lösung, wohingegen die Rechenzeit, um höhere zu erreichen, schnell ansteigt.

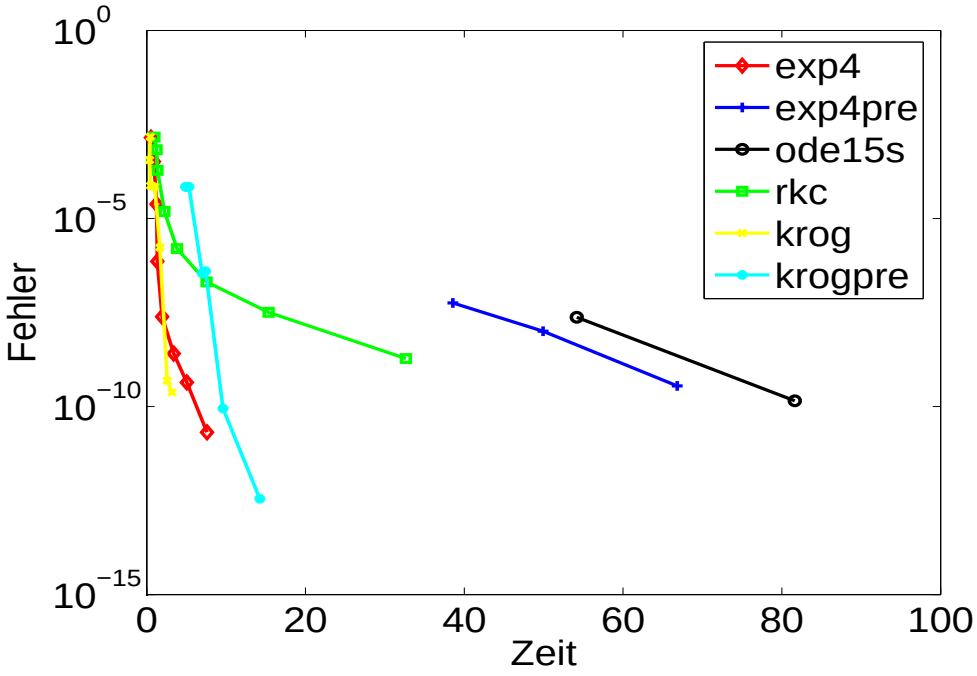

Abb. 5.4: Fehler als Funktion der Rechenzeit für Gleichung (5.3).

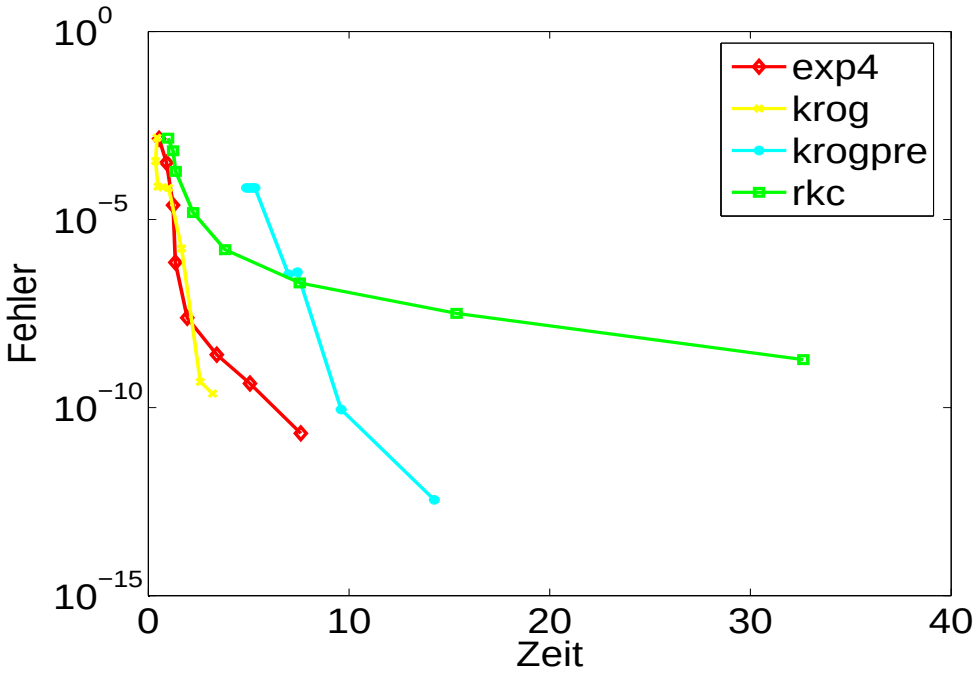

Abb. 5.5: Vergrößerung von Abb. 5.4.

Toleranz	Fehler	Zeit in s	Zeitschritte	Krylov-Schritte	$\frac{\text{Krylov-Schritte}}{\text{Zeitschritte}}$
10^{-1}	$1.40 \cdot 10^{-3}$	0.53	8	43	5.4
10^{-2}	$3.26 \cdot 10^{-4}$	0.91	10	79	7.9
10^{-3}	$2.41 \cdot 10^{-5}$	1.23	11	97	8.8
10^{-4}	$7.26 \cdot 10^{-7}$	1.34	12	130	10.8
10^{-5}	$2.44 \cdot 10^{-8}$	1.94	14	217	15.5
10^{-6}	$2.56 \cdot 10^{-9}$	3.41	18	365	20.3
10^{-7}	$4.36 \cdot 10^{-10}$	5.06	27	556	20.6
10^{-8}	$2.06 \cdot 10^{-11}$	7.59	44	829	18.8

Tabelle 5.11: Ergebnisse der Lösung von Gleichung (5.3) mit *exp*4

Toleranz	Fehler	Zeit in s	Zeitschritte	Krylov-Schritte	$\frac{\text{Krylov-Schritte}}{\text{Zeitschritte}}$
10^{-1}	$5.69 \cdot 10^{-8}$	38.56	9	46	5.1
10^{-2}	$1.00 \cdot 10^{-8}$	49.92	11	81	7.4
10^{-4}	$3.55 \cdot 10^{-10}$	66.83	13	170	13.1

Tabelle 5.12: Ergebnisse der Lösung von Gleichung (5.3) mit *exp*4 und Vorkonditionierung

Toleranz	Fehler	Zeit in s	Zeitschritte	Krylov-Schritte	$\frac{\text{Krylov-Schritte}}{\text{Zeitschritte}}$
10^{-1}	$1.49 \cdot 10^{-3}$	0.44	1	28	28
10^{-2}	$3.60 \cdot 10^{-4}$	0.36	1	40	40
10^{-3}	$7.47 \cdot 10^{-5}$	0.48	1	54	54
10^{-4}	$7.16 \cdot 10^{-5}$	0.72	1	70	70
10^{-5}	$6.94 \cdot 10^{-5}$	1.02	1	98	98
10^{-6}	$1.70 \cdot 10^{-6}$	1.66	2	145	72.5
10^{-7}	$4.85 \cdot 10^{-10}$	2.59	3	223	74.3
10^{-8}	$2.37 \cdot 10^{-10}$	3.20	3	267	89

Tabelle 5.13: Ergebnisse der Lösung von Gleichung (5.3) mit *krog* und vorgegebener Schrittweite

Toleranz	Fehler	Zeit in s	Zeitschritte	Krylov-Schritte	$\frac{\text{Krylov-Schritte}}{\text{Zeitschritte}}$
10^{-1}	$6.93 \cdot 10^{-5}$	4.89	1	7	7
10^{-2}	$6.92 \cdot 10^{-5}$	5.02	1	8	8
10^{-3}	$6.92 \cdot 10^{-5}$	5.22	1	10	10
10^{-4}	$6.93 \cdot 10^{-5}$	5.31	1	11	11
10^{-5}	$3.49 \cdot 10^{-7}$	6.98	2	22	11
10^{-6}	$3.99 \cdot 10^{-7}$	7.42	2	26	13
10^{-7}	$8.90 \cdot 10^{-11}$	9.61	3	42	14
10^{-8}	$3.51 \cdot 10^{-13}$	14.25	5	77	15.4

Tabelle 5.14: Ergebnisse der Lösung von Gleichung (5.3) mit *krog*, vorgegebener Schrittweite und Vorkonditionierung

Toleranz	Fehler	Zeit in s	Zeitschritte
10^{-1}	$2.36 \cdot 10^{-8}$	54.17	21
10^{-2}	$1.42 \cdot 10^{-10}$	81.63	35

Tabelle 5.15: Ergebnisse der Lösung von Gleichung (5.3) mit $ode15s$

Toleranz	Fehler	Zeit in s	Zeitschritte
10^{-1}	$1.50 \cdot 10^{-3}$	0.98	13
10^{-2}	$6.76 \cdot 10^{-4}$	1.23	19
10^{-3}	$1.91 \cdot 10^{-4}$	1.36	25
10^{-4}	$1.53 \cdot 10^{-5}$	2.23	45
10^{-5}	$1.58 \cdot 10^{-6}$	3.83	90
10^{-6}	$2.06 \cdot 10^{-7}$	7.55	189
10^{-7}	$3.21 \cdot 10^{-8}$	15.36	407
10^{-8}	$1.89 \cdot 10^{-9}$	32.64	887

Tabelle 5.16: Ergebnisse der Lösung von Gleichung (5.3) mit rkc

Literatur

[1] J.-E. Andersson. Approximation of e^{-x} by rational functions with concentrated negative poles. J. Approx. Theory, 32(2), 85-95, 1981

[2] S. Cox und P. Matthews. Exponential time differencing for stiff systems, Journal of Computational Physics, 176, 430-455, 2002

[3] G. H. Golub, Z. Zhang und H. Zha. Large sparse symmetric eigenvalue problems with homogeneous linear constraints: the Lanczos process with inner-outer iterations. Linear Algebra Apll., 309(1-3), 289-306, 2000

[4] K. Gustafsson, M. Lundh und G. Söderlind, A PI stepsize control for the numerical solution of ordinary differential equations. BIT 28, 270-287, 1988

[5] M. Hochbruck und Ch. Lubich. On Krylov subspace approximations to the matrix exponential operator. SIAM J. Num. Anal., 34, 1911-1925, 1997

[6] M. Hochbruck, Ch. Lubich und H. Selhofer. Exponential integrators for large systems of differential equations. SIAM J. Num. Anal., 19, 1552-1574, 1998

[7] M. Hochbruck, Skripte zu Numerik Vorlesungen. Heinrich-Heine-Universität Düsseldorf

[8] M.Hochbruck, A. Ostermann. Explicit exponentiel Runge-Kutta methods for semilinear parabolic problems, SINUM 2005

[9] J. Niehoff, Dissertation. Heinrich-Heine-Universität Düsseldorf, 2006

[10] S. Krogstad, Generalized integrating factor methods for stiff PDEs, J. Comput. Phys., 203(1), 72-88, 2005

[11] Y. Saad, Analysis of some Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal., 29(1), 209-228, 1992

[12] E.B. Saff, A. Schönhage und R. S. Varga. Geometric convergence to e^{-z} by rational functions with real poles. Numer. Math., 25(3), 307-322, 1975/76

[13] L.F. Shampine, Numerical Solution of the Ordinary Differential Equations, Chapman and Hall, New York, 1994

[14] V. Simoncini und D. B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. SIAM J. Sci. Comput., 25(2), 454-477, 2003

[15] B.P. Sommeijer und J.G. Verwer, A perfomance evaluation of a class of Runge-Kutta-Chebyshev methods for solving semi-discrete parabolic differential equations, Report NW91/80, Mathematisch Centrum, Amsterdam, 1980

[16] B.P. Sommeijer, RKC, a nearly-stiff ODE solver. Available from netlib@ornl.gov, send rkc.f from ode, 1991

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

[18] P.J. van der Houwen und B.P. Sommeijer, On the internal stability of explicit m-stage Runge-Kutta methods for large values of m, ZAMM 60, 479-485, 1980

[19] J. van den Eshof und M. Hochbruck: Preconditioning Lanczos approximations to the matrix exponentiell, SIAM J. Sci. Comput., 27, 1438-1457, 2006

[20] J. van den Eshof und G. L. G. Sleijpen. Inexact Krylov subspace methods for linear systems, SIAM J. Matrix Anal. and Appl., 26(1), 125-153, 2004

[21] J.G. Verwer, W.H. Hundsdorfer und B.P. Sommeijer, Convergence properties of the Runge-Kutta-Chebyshev method, Numer. Math. 57, 157-178, 1990

[22] H.A. Watts, Step size control in ordinary differential equation solvers, Trans. Soc. for Computer Simulation 1, 15-25, 1984

Ich versichere hiermit, dass ich die vorgelegte Hausarbeit selbstständig verfasst und nur die angegebenen Quellen verwendet habe.

Jalo Liljo