Numerische Mathematik II

Numerik für gewöhnliche Differentialgleichungen, Numerik von Eigenwertproblemen und Singulärwertzerlegungen sowie Krylowraum-Verfahren

Leon Samolik

Mitschrift SoSe 2019
Ergänzungen SoSe 2024

gelesen von

Prof. Dr. Thomas Wick
Institut für Angewandte Mathematik
Leibniz Universität Hannover

7. September 2024

Danksagung

Dank geht an die Studierenden der SoSe 2019 und SoSe 2024 für zahlreiche Rückfragen in den Vorlesungen, die somit direkt und indirekt Eingang in dieses Skriptum erhalten haben. Außerdem geht spezieller Dank an die jeweiligen Übungsleiter Jan Philipp Thiele, Robin Görmer, Leon Kolditz und Tobias Knoke, die durch Rückfragen und Diskussionen ebenfalls zu den Inhalten dieses Skriptums beigetragen haben.

Inhaltsverzeichnis

1. Numerik gewöhnlicher Differentialgleichungen (GDGL)

1.1. Beispiele, Methoden, usw.

1.1.1. Definition einer Differentialgleichung

Für Differentialgleichungen gibt es oft keine geschlossenen Formeln, sondern näherungsweise Bestimmungen. Dabei wird ein kontinuierliches Ausgangsproblem durch eine Folge von Approximationen bestimmt. Aber was sind Differentialgleichungen?

Definition 1.1.1 Differentialgleichung

Eine Differentialgleichung enthält eine unbekannte, d. h. eine zu bestimmende Funktion, welche in der zugrundeliegenden Gleichung die Funktion selbst mit ihren Ableitungen verknüpft.

Definition 1.1.2 Gewöhnliche Differentialgleichung

In einer gewöhnlichen Differentialgleichung hängt die zu suchende Funktion lediglich von einer unabhängigen Variablen ab. Oft ist dies die "Zeit".

In mathematischer Notation beschäftigen wir uns also mit:

$$F\big(t, u(t), u'(t), \ldots\big) = 0 \qquad \text{(implizite Form}^{1}\text{)}$$

$$\alpha\, u(t) + \beta\, u'(t) + \cdots = r(t) \qquad \text{(explizite Form)}$$

Dabei ist t die unabhängige Variable, u die unbekannte gesuchte Funktion und u' die erste Ableitung. Wie in der Notation angedeutet, sind auch höhere Ableitungen von u möglich. Beachtenswert ist hierbei der Unterschied zwischen impliziter und expliziter Form. Salopp ausgedrückt ist eine Differentialgleichung explizit, sofern diese nach der höchsten Ableitung umstellbar ist.

[1] Implizite Formen sind oft sehr anspruchsvoll. Erfreulicherweise hat man es oft mit expliziten Darstellungen zu tun.

Das ist keineswegs immer der Fall, wie am Beispiel

$$u'(t) + e^{u'(t)} = u(t)$$

klar wird. In der Regel wird das Funktionsargument weggelassen, wenn dieses bekannt ist oder aus dem Kontext erschlossen werden kann. Das Beispiel könnte also auch als

$$u' + e^{u'} = u \quad \Leftrightarrow \quad F(t, u, u') := u' + e^{u'} - u = 0$$

formuliert werden. Weil aber eine explizite Differentialgleichung auch in impliziter Form geschrieben werden kann, wird für explizit schreibbare Differentialgleichungen die implizite Form als $f(t, u, u', \ldots)$ bezeichnet, statt als $F(t, u, u', \ldots)$.

Neben den gewöhnlichen Differentialgleichungen gibt es noch die partiellen Differentialgleichungen, welche eine funktionale Beziehung der (impliziten) Form

$$F\big(t, x, \partial_t u(t, x), \partial_x u(t, x)\big) = 0$$

besitzen. Dabei sind nun zwei unabhängige Variablen und die partiellen Ableitungen von u angegeben. Anders ausgedrückt:

Definition 1.1.3 Partielle Differentialgleichung

In einer partiellen Differentialgleichung hängt die zu suchende Funktion von mindestens zwei unabhängigen Variablen ab. Z. B. von der "Zeit" und einer Ortsvariablen.

Partielle Differentialgleichungen sind Objekt der Numerik III und werden hier deshalb nicht weiter behandelt. Womit wir uns allerdings beschäftigen werden, ist die Lösung zum Modellproblem.

Definition 1.1.4 Modellproblem

Die explizite Form einer GDGL 1. Ordnung lautet $u'(t) = f\big(t, u(t)\big)$. Die Ordnung gibt hierbei die Ableitungsstufe in der Gleichung an. Das Modellproblem lautet dann: Sei $I = (t_0, T)$ mit einem Endzeitpunkt $T > t_0$. Finde $u(t) : I \to \mathbb{R}$, sodass die Differentialgleichung $u'(t) = f\big(t, u(t)\big)$ gelöst und die Anfangsbedingung $u(t_0) = u_0$ erfüllt ist.

Explizite wie auch implizite Differentialgleichungen lassen sich in AWAs (Anfangswertprobleme) und RWPs (Randwertprobleme) unterteilen. AWAs sind Differentialgleichungen mit einem gegebenen Anfangswert wohingegen RWPs solche mit gegebenen Randwerten sind. Zur vollständigen Beschreibung einer DGL gehören demnach zwei Dinge: Die Gleichung, z. B. $u'(t) = f(t, u)$,

und Anfangs- bzw. Randwerte. Die Eindeutigkeit einer Lösung wird u. a. vom Anfangswert festgelegt. Für Differentialgleichungen höherer Ordnung benötigt man entsprechend mehr Anfangswerte – genau so viele, wie Ordnungsstufen.

1.1.2. Beispiele von DGLen

Aus der Analysis II sind einige analytische Verfahren zum Lösen gewöhnlicher Differentialgleichungen bekannt. Zwei Standardmethoden sind hierbei die "Trennung der Variablen" und "Variation der Konstanten". Beide Methoden werden hier bloß schematisch anhand ihrer Heuristik vorgestellt, sind also ursprünglich sehr viel abstraker und sogleich präziser. Besonders die Variation der Konstanten trifft eine wesentlich allgemeinere Aussage für lineare Differentialgleichungssysteme, einschließlich höherer Ordnungen. Das heuristische Lösen ist mit den allgemeinen Aussagen allerdings nicht unmittelbar ersichtlich, was gerade das ist, was für uns relevant wird.

1.1.5 Trennung der Variablen

Gegeben sei eine separable Differentialgleichung $u'(t) = f\big(t, u(t)\big) = g(t) \cdot h\big(u(t)\big)$ mit Anfangswert $u(t_0) = u_0$. Ferner sei u eine Lösung der Differentialgleichung mit $h\big(u(t)\big) \neq 0$. Dann gilt

$$\int_{t_0}^{t} \frac{u'(s)}{h\big(u(s)\big)} \,\mathrm{d}s = \int_{t_0}^{t} g(s) \,\mathrm{d}s.$$

Durch Substitution $z := u(s)$ ergibt sich somit

$$\int_{u_0}^{u(t)} \frac{1}{h(z)} \,\mathrm{d}z = \int_{t_0}^{t} g(s) \,\mathrm{d}s.$$

Ohne Anfangswert gilt die allgemeine Integralgleichung

$$\int \frac{1}{h(z)} \,\mathrm{d}z = \int g(s) \,\mathrm{d}s,$$

in welcher die Integrationskonstante durch einen Anfangswert gelöst werden kann.

Nennenswert ist hier die Voraussetzung einer Lösung u. Entsprechend liefert das heuristische Lösen über die Integralgleichungen nicht unbedingt immer eine Lösung der Differentialgleichung, weshalb zum Ende ein Test durch Einsetzen ausgeführt werden sollte.

1.1.6 Variation der Konstanten

Gegeben sei eine lineare Differentialgleichung $u(t) + a(t)\,u(t) = r(t)$ mit $a, r \in C(\mathbb{R})$. Sei v eine Lösung der dazugehörigen homogenen Differentialgleichung $v(t) + a(t)\,v(t) = 0$. Für $c \in \mathbb{R}$ ist $c\,v$ ebenfalls eine Lösung der homogenen Differentialgleichung. Zur Lösung der inhomogenen Differentialgleichung wird v als Ansatzfunktion benutzt, deren Konstante c als Funktion variiert wird, sprich

$$u(t) := c(t)\,v(t) \quad \Rightarrow \quad u'(t) = c(t)\,v'(t) + c'(t)\,v(t).$$

In die inhomogene Differentialgleichung eingesetzt ergibt das

$$\begin{aligned} c(t)\,v'(t) + c'(t)\,v(t) + a(t)\,c(t)\,v(t) &= c(t)\overbrace{\big[v'(t) + a(t)\,v(t)\big]}^{\text{Lösung der homogenen DGL}} + c'(t)\,v(t) \\ &= c'(t)\,v(t) = r(t). \end{aligned}$$

Daraus erhält man

$$c(t) = \int_{t_0}^{t} \frac{r(s)}{v(s)}\,\mathrm{d}s + \gamma, \quad \gamma \in \mathbb{R},$$

womit $u = c\,v$ per Konstruktion eine Lösung der inhomogenen Differentialgleichung ist.

Beispiele

(a) $u'(t) = 0 \Rightarrow u(t) = \text{const}$. Konkreter ist $u(t) \equiv u_0$ mit Anfangswert u_0.

(b) $u'(t) = g(t)$ mit $g(t)$ integrierbar in $t \Rightarrow u(t) = \int g(t)\,\mathrm{d}t + c$. Hier entsteht eine Integrationskonstante, welche mit einem Anfangswert gelöst werden kann.

(c) Modellproblem: $u'(t) = u(t),\ u(t_0) = u_0$. Mit Trennung der Variablen (1.1.5) erhält man

$$\int \frac{1}{u}\,\mathrm{d}u = \int 1\,\mathrm{d}t \quad \Leftrightarrow \quad \ln|u| + c = t \quad \Rightarrow \quad u(t) = \underbrace{\exp(-c)}_{=:\tilde{c}}\,\exp(t)$$

Mittels Anfangswert folgt für $\tilde{c}$:

$$u(t_0) = \tilde{c}\,\exp(t_0) \quad \Rightarrow \quad \tilde{c} = u_0\,\exp(-t_0)$$

Zusammen erhält man die spezielle Lösung $u(t) = u_0\,\exp(t - t_0)$.

(d) Eine DGL 2. Ordnung: $u''(t) = u(t) \Rightarrow u_1(t) = \exp(t),\ u_2(t) = \exp(-t)$. Dementsprechend sind auch alle Linearkombinationen $u(t) = \alpha\,u_1(t) + \beta\,u_2(t),\ \alpha, \beta \in \mathbb{R}$ Lösungen der DGL.

Solche expliziten Lösungen sind lediglich für einfache DGL möglich, ansonsten bräuchte man gar keine Numerik. In der Praxis hat man oft numerische Methoden zur Hand, die speziell für 1. Ordnungs-DGL entwickelt worden sind. Demnach werden DGL der Form $u''(t) = u(t)$ häufig mit Hilfe von 1. Ableitungen formuliert. Dadurch erhält man ein System von DGLen. Für das vierte Beispiel führe man eine Zusatzvariable $v(t)$ ein. Dann erhält man

$$u'(t) = v(t)$$
$$v'(t) = u(t).$$

Diese ineinander eingesetzt ergeben $u''(t) = u(t)$. Nachteilig ist dabei, dass zwei Lösungen $u(t)$ und $v(t)$ bestimmt werden müssen. Der Vorteil ist, dass lediglich 1. Ableitungen vorhanden sind.

Systeme von DGLen entstehen aber auch, wenn unterschiedliche Prozesse beobachtet werden, die sich gegenseitig beeinflussen (gekoppelte DGL). Ein Beispiel ist das Räuber-Beute-Modell[2]:

$$u'(t) = \quad a\,u(t) - b\,u(t)v(t), \quad a, b \geq 0$$
$$v'(t) = -c\,v(t) + d\,u(t)v(t), \quad c, d \geq 0$$

Hierbei bezeichnet $u(t)$ die Anzahl der Beutetiere und $v(t)$ die Anzahl der Räuber.

(i) $b = 0$ bzw. $v(t) \equiv 0 \Rightarrow u'(t) = a\,u(t) \Rightarrow u(t) = \tilde{c}\,\exp(t)$, also exponentielles Wachstum der Beute.

(ii) $u(t) \equiv 0 \Rightarrow v'(t) = -c\,v(t)$, also exponentielles Aussterben der Jäger (keine Nahrung vorhanden).

(iii) Falls $u(t) \not\equiv 0$ und $v(t) \not\equiv 0$, dann gibt es ein stetiges Wechselspiel zwischen Wachstum und Reduzieren von Jägern bzw. Beute.

Für DGLen höherer Ordnung

$$u^{(m)}(t) = f\big(t, u(t), u'(t), \ldots, u^{(m-1)}(t)\big),$$

also DGLen mit Ordnung m, definiert man sich Hilfsfunktionen

$$\begin{aligned} u_1(t) &= u(t) \\ &\vdots \\ u_m(t) &= u^{(m-1)}(t), \end{aligned}$$

[2]Dieses Beispiel ist aus der mathematischen Modellierung und hat zunächst nichts mit Numerik zu tun.

damit man aus der ursprünglichen DGL das DGL-System

$$\begin{aligned} u_1'(t) &= u_2(t) \\ &\vdots \\ u_{m-1}'(t) &= u_m(t) \\ u_m'(t) &= f(t, u_1, u_2, \ldots, u_m) \end{aligned}$$

entwickeln kann. Daraus erhält man die kompakte Schreibweise

$$u(t) = \begin{pmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{pmatrix} \in \mathbb{R}^m, \quad f(t,x) = \begin{pmatrix} f_1(t,x) \\ \vdots \\ f_m(t,x) \end{pmatrix}.$$

Damit lässt sich das ursprüngliche Problem als $u'(t) = f\big(t, u(t)\big)$ schreiben. Mit Hilfe dieses Beispiels kann man nun auf kompakte Weise eine AWA formulieren. So ist eine AWA eine Aufgabe

$$u'(t) = f(t,u), \quad u(t_0) = u_0$$

mit $t \geq t_0 = 0$. AWAs mit lediglich einer Lösungskomponente können bereits sehr unterschiedliches Lösungsverhalten aufweisen, wie folgende Beispiele zeigen sollen:

Beispiele

(a) $u'(t) = t^{-1}\, u(t)$ (1. Ordnung, linear) besitzt die Lösung $u(t) = t$.

(b) $u'(t) = t\,\big(u(t)\big)^{-1}$ (1. Ordnung, nichtlinear) wird für $u(t) \to 0$ singulär, hat aber dennoch eine globale Lösung auf ganz $\mathbb{R}$, nämlich $u(t) = \sqrt{1+t^2}$.

(c) $u'(t) = \big(u(t)\big)^2$ (1. Ordnung, nichtlinear) hat bei $t = 1$ eine Singularität. Darum besitzt diese DGL nur eine lokale Lösung $u(t) = (1-t)^{-1}$. Diese ist bei der Singularität nicht definiert.

(d) Fallender Apfel: Betrachte $-u''(t) = f(t,u)$. Diese DGL ist 2. Ordnung, also werden zwei Anfangswerte zum Lösen benötigt. Für einen fallenden Apfel kann man $-u''(t) = g$ mit der Gravitationskonstanten g nutzen, um die zeitabhängige Position unter Vernachlässigung der Reibung darzustellen. Hier kann man als Anfangswerte die initiale Position $u(t_0)$ und Geschwindigkeit $u'(t_0)$ verwenden.

(e) Randwertproblem: Nicht alle Probleme besitzen Anfangswerte wie das vorangegangene Beispiel. Mit derselben DGL, nur diesmal mit einer Ortsabhängigkeit, kann eine Wäscheleine dargestellt werden: $-u''(x) = f(x,u)$. Hier gibt es sowas wie eine initiale Geschwindigkeit nicht, dafür aber zwei Randwerte, die das Verhalten einer Lösung am Rand beschreiben. Diese sind ebenfalls ausreichend, um eine spezielle Lösung zu finden, solange es so viele

Randwerte wie Ordnungsstufen gibt. Dieses Beispiel ist ein Fall einer Kettenlinie (engl. "catenary"), dessen allgemeine Lösung Cosinus-hyperbolicus-Funktionen sind.

(f) Allgemeine lineare DGL 1. Ordnung: $u'(t) + a(t)\,u(t) = r(t)$. Die Gesamtlösung hiervon ist $u = u_S + u_H$ mit der Lösung der homogenen DGL u_H (d. h. Annahme $r \equiv 0$), welche durch T. d. V. (1.1.5) bestimmt werden kann, und der speziellen Lösung der inhomogenen DGL u_S, die aus V. d. K. (1.1.6) erschlossen werden kann. Um das einmal konkreter zu sehen, sei $u'(t) - \frac{u(t)}{t} = 3t$ für $t \neq 0$. Die dazugehörige homogene DGL lautet dann $u' - \frac{u}{t} = 0$. Per T. d. V. erhält man hieraus

$$\int \frac{1}{u}\,\mathrm{d}u = \int \frac{1}{t}\,\mathrm{d}t \quad \Rightarrow \quad \ln|u| = \ln|t| \quad \Rightarrow \quad u = ct,\ c \in \mathbb{R}.$$

Das liefert $u_H = t$. Per V. d. K. wird der konstante Teil, hier c, als Function $c(t)$ betrachtet. Dann ist

$$u_S(t) := c(t)\,u_H(t) = t\,c(t) \quad \Rightarrow \quad u_S'(t) = c'(t)\,u_H(t) + c(t)\,u_H'(t)$$

Das in die ursprüngliche DGL eingesetzt, sprich $u \to u_S$, resultiert in

$$\begin{aligned} & c'(t)\,u_H(t) + c(t)\,u_H'(t) - \frac{c(t)\,u_H(t)}{t} = 3t \quad \Leftrightarrow \quad t\,c'(t) + c(t) - \frac{t\,c(t)}{t} = 3t \\ \Leftrightarrow \quad & t\,c'(t) = 3t \qquad\qquad\qquad\qquad\quad \Leftrightarrow \quad c'(t) = 3 \end{aligned}$$

Damit erhalten wir $c(t) = 3t$ und daher $u(t) = u_H(t) + u_S(t) = ct + 3t^2$. Diese Lösung kann durch eine Einsetzungsprobe in die DGL verifiziert werden.

1.1.3. Klassifizierungen von DGLen

Klassifizierungen von DGLen

(1) Linear oder nichtlinear

(2) Erste Ordnung (niedrige Ordnung) oder höhere Ordnung

(3) 1-komponentige DGL oder DGL-Systeme

(4) Autonome DGL: $f(t, u) = f(u)$, sprich die rechte Seite hängt nicht explizit von t ab

(5) Separierte DGL: $f(t, u) = a(t)\,g(u)$

(6) Homogene DGL: Für $u'(t) = u(t) + b$ ist die DGL homogen, wenn $b \equiv 0$, nicht-homogen wenn $b \not\equiv 0$ (nur für lineare DGLen interessant)

Beispiele

(a) $u'(t) = a\,u(t),\ a > 0$ ist linear, 1. Ordnung, 1-komponentig, autonom und homogen.

(b) $u'(t) = t^{-1}\,u(t)$ ist linear, 1. Ordnung, 1-komponentig, nicht-autonom, separiert und homogen.

(c) $a\,u''(t) + b\,u'(t) + c\,u(t) = 0,\ a, b, c > 0$ ist linear, 2. Ordnung, 1-komponentig und homogen.

(d) $u'(t) = b\left(u(t)\right)^2 + c,\ b, c > 0$ ist nicht-linear, 1. Ordnung, 1-komponentig, autonom und nicht-homogen.

(e) Die DGL des Räuber-Beute-Modells ist nicht-linear, 1. Ordnung, ein DGL-System, autonom und homogen.

1.2. Einblick in die mathematische Modellierung von GDGL

1.2.1. Räuber-Beute

Hintergrund: Der Biologe Umberto D'Ancona hat sich in den 1920er Jahren die Fischentwicklungen im Mittelmeer von 1914–1923 angeschaut (Braun 1983) (siehe auch (Quarteroni und P.Gervasio 2020)). Die Räuber waren dabei Haie und Rochen. Diese sind nicht sehr schmackhaft und daher nicht zum Verzehr geeignet. Auf der anderen Seite waren die sogenannten Beutefische. Bei diesen gibt es zwei Probleme. Zum Ersten werden sie von dem Menschen gefischt, zum Weiteren von den Räubern gefressen. D'Ancona beobachtete, dass in 1918 der prozentuale Fang der Räuberfische signifikant angestiegen ist. Seine erste Vermutung war, dass es aufgrund des ersten Weltkriegs nur wenig Fischfang gab. Allerdings waren auch mehr Beutefische verfügbar, sodass sich der prozentuale Anteil des Räuber-Fischfangs nicht hätte so stark verändern dürfen. D'Anconas Berechnungen konnten lediglich zeigen, dass absolut-gesehen mehr Räuberfische existieren mussten, gaben aber keine Erklärung für den starken prozentualen Anstieg. D. h. weshalb weniger Fischfang für die Räuberfische vorteilhafter sein sollte, war für D'Ancona vorerst ein Widerspruch. Daraufhin kontaktierte er seinen italienischen Kollegen Vito Volterra, um einen mathematischen Ansatz für sein Problem zu finden.

Volterra separierte zuerst die Fische in Anzahl der Beutefische $x(t)$ und Anzahl der Räuberfische $y(t)$ zum Zeitpunkt t. Hier gibt es eine unabhängige Variable, nämlich die Zeit t. Somit handelt es sich hierum um ein GDGL. Er stellte dann zwei Annahmen. Einmal, dass es keine Räuber gibt und dann noch, dass die Dichte der Beutefische nicht zu groß ist, d. h. die Beutefische nehmen sich nicht gegenseitig das Futter weg. Dann nutze er das einfachste Wachstumsmodell (Thomas Robert Malthus, 1798). Dieses beruht auf der Idee, dass die Änderung proportional zum Ist-Zustand ist:

$$\dot{x} = ax,\ a > 0.$$

Daraus ergibt sich die erste GDGL $\frac{\mathrm{d}x}{\mathrm{d}t} = ax$ (exponentielles Wachstum). Nun kommen die Räuber ins Spiel. Volterra nahm an, dass sich pro Zeiteinheit t Beute und Räuber mit der Rate $b > 0$ treffen. Daraus erhält man die Modifikation der ersten GDGL

$$\frac{\mathrm{d}x}{\mathrm{d}t} = ax - bxy.$$

So steht schon mal die erste Gleichung des Volterra-Systems.

Für die Räuberfische traf er analog die erste Annehme, es gäbe zunächst keine Beutefische, d. h. die Räuberfische werden dezimiert. Daraus erhält man die GDGL

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -cy,$$

wobei $c > 0$ die Sterblichkeitsrate beschreibt. Hier kommen nun die Beutefische zurück ins Spiel. Es gibt $x(t)$ Beutefische, d. h. pro Kontakt zwischen Räuber und Beute steigt die Popoulation $y(t)$ wieder an. Die vorherige GDGL kann also zu

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -cy + dxy$$

mit $d > 0$ modifiziert werden. Das ist gerade die zweite Gleichung des Volterra-Systems.

Daraus erhält man nun insgesamt die Aufgabe: Gegeben seien $a, b, c, d > 0$. Berechne $x(t)$ und $y(t)$ für $t \geq t_0$ mittels des nicht-linearen, gekoppelten Systems

$$\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t} &= \quad ax - bxy \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= -cy + dxy\end{aligned}$$

und $x(t_0) = x_0$, $y(t_0) = y_0$.

Nun fehlt noch der Einfluss des Fischfangs. Dazu nahm Volterra an, man würde Räuber und Beute gleichermaßen mit Rate $\varepsilon > 0$ fangen. Daraus erhält man die mathematische Modellierung $-\varepsilon\, x(t)$ und $-\varepsilon\, y(t)$. Damit erhält man schlussendlich

$$\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t} &= \quad (a-\varepsilon)x - bxy \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= -(c+\varepsilon)y + dxy\end{aligned}$$

Nun sieht man einen Zusammenhang zwischen der Wachstumsrate a und der Fangrate ε. Wenn $a > \varepsilon$, dann kann sich die Beutepopulation prinzipiell selbst regenerieren. Falls $\varepsilon > a$ (Überfischung), dann kann sich die Beutepopulation nicht selbst regenerieren. Volterra schloss weiter:

- Beutefische werden mehr und Räuberfische werden weniger, sofern $\varepsilon < a$.
- Beutefische werden weniger und Räuberfische werden mehr, wenn $\varepsilon \ll a$.

In Abhängigkeit der Parameter b und d ist die entscheidende Beobachtung, dass für $\varepsilon_1 < a$ und $\varepsilon_2 \ll a$ auch

$$-(c+\varepsilon_1) < (c+\varepsilon_2)$$

gilt. D. h. bei moderatem Fischfang nimmt die Räuberfischpopulation durchschnittlich stärker ab, als bei stark reduziertem Fischfang. Diese Schlussfolgerung nennt sich Volterras Prinzip. Mit diesem Modell ging Volterra zu D'Ancona, um seine mathematischen Berechnungen zu diskutieren. Dies löste in der Tat den vermeintlichen Widerspruch D'Anconas.

1.2.2. Populationsmodelle

Hier soll nun eine kleine Herleitung des Malthusischen Gesetzes erbracht werden. Sei zunächst $p(t)$ die Anzahl einer Spezies. Welche Spezies ist für die Theorie irrelevant. Seien g die Geburten- und m die Sterblichkeitsrate. Man definiere $a := g - m$. Dann ist die relative Änderung $\mathrm{d}p$ der Spezies pro Zeiteinheit $\mathrm{d}t$ wie folgt gegeben:

$$\frac{\mathrm{d}p}{p} = a\,\mathrm{d}t \quad \Leftrightarrow \quad \frac{\mathrm{d}p}{\mathrm{d}t} = a\,p(t) \qquad \text{(Malthus-Gesetz)}$$

Prinzipiell wird angenommen, Bevölkerungsvariationen verhielten sich kontinuierlich. Eigentlich liegen aber nur Stichproben vor. Eine Aufgabe würde dann so lauten: Gegeben sei $a > 0$. Berechne $p(t)$ für $t \geq t_0$, sodass das Malthusische Gesetz von oben mit $p(t_0) = p_0$ gilt. Anwendung ist z. B. die Entwicklung der Erdbevölkerung. Dazu soll ein Beispiel Beispiel mit Literaturwerten betrachtet werden: Seien $p(t)$ die Anzahl der Menschen zum Zeitpunkt t, $t_0 = 1965$ das Startjahr, $p_0 = 3,34 \cdot 10^9$ die Anzahl der Menschen im Startjahr. Durch experimentelle Beobachtungen bestimmt man von 1960–1970 die Entwicklung der Erdbevölkerung und erhält $a = 0,02$. Das entspricht einem 2% Wachstum. Somit erhält man

$$p(t) = 3,34 \cdot 10^9 \cdot \exp\big(0,02 \cdot (t - 1965)\big).$$

Für die Jahre $t < 1965$ stimmt diese Formel erstaunlich gut. Aber gilt diese Formel auch für zukünftige Entwicklungen?

t	$p(t)$	Statistik	Fläche[3]/Mensch
2000	$6,72 \cdot 10^9$		
2019	$9,83 \cdot 10^9$	$7.67 \cdot 10^9$	$1,5 \cdot 10^5\,\mathrm{m}^2$
2100	$4,97 \cdot 10^{10}$		$7,45\,\mathrm{m}^2$
2515	$2 \cdot 10^{14}$		$0,826\,\mathrm{m}^2$
2625	$3,6 \cdot 10^{15}$		
2660	$4,97 \cdot 10^{15}$		

[3]Gemeint ist hier die geographische Landfläche.

Schlussfolgerung: Hoffentlich ist das Malthusische Gesetz in dieser Form falsch. Es ist somit eine Modifikation nötig, die logistische DGL von Verhulst (1837). Dabei wird ein Term eingeführt, der berücksichtigt, dass Bevölkerungen dezimiert werden, wenn zu viele Individuen existieren (Nahrungsmangel, Krieg, soziale Verhaltensweisen). Die Idee ist die DGL $\frac{\mathrm{d}p}{\mathrm{d}t} = ap - bp^2$ mit $a, b > 0$ und $b \ll a$. Der Subtrahend $-bp^2$ stellt hierbei den Wettbewerb zwischen zwei Mitgliedern einer Spezies dar, ähnlich wie bei der Räuber/Beute-Interaktion. Dies ist also eine nicht-lineare, separable DGL. Die Separabilität erlaubt damit die Angabe einer expliziten Lösung. Nach (Braun 1983) ist

$$p(t) = \frac{ap_0}{bp_0 + (a - bp_0)\,\exp\big(-a(t - t_0)\big)}$$

eine Lösung der DGL. Interessant ist hierbei der Grenzwert für $t \to \infty$, denn im Divisor konvergiert $\exp\big(-a(t - t_0)\big)$ gegen 0 und somit ist

$$\lim_{t\to\infty} p(t) = \lim_{t\to\infty} \frac{ap_0}{bp_0 + (a - bp_0)\,\exp\big(-a(t - t_0)\big)} = \frac{ap_0}{bp_0} = \frac{a}{b}.$$

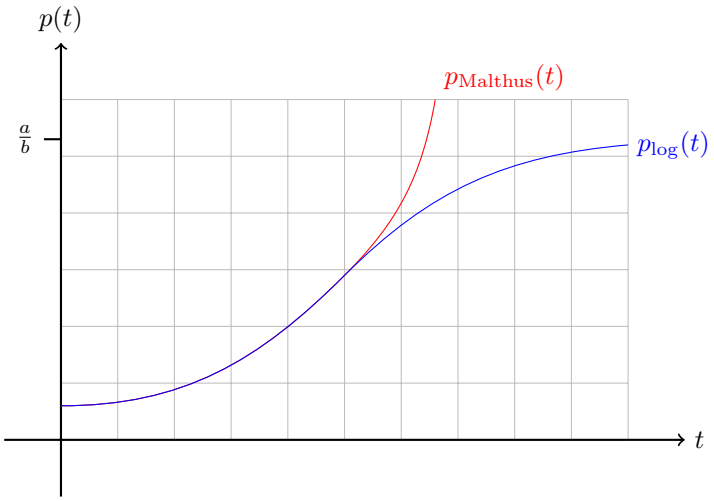


Abbildung 1.1.: Vergleich von exponentiellem und logarithmischem Wachstum.

Damit lässt sich die Anwendung der Modellierung der Erdbevölkerung fortsetzen. Mit $\frac{1}{p} \cdot \frac{\mathrm{d}p}{\mathrm{d}t}$ wird das Wachstum zum Zeitpunkt t beschrieben. Für $t = t_0$ von zuvor ist die Wachstumsrate von $0,02$ bekannt. Zudem ist $p(t_0) = p_0 = 3,34 \cdot 10^9$ für $t_0 = 1965$. Nach Brown ist nun $a = 0,029$ und b ergibt sich als Lösung der DGL $\frac{\mathrm{d}p}{\mathrm{d}t} = ap - bp^2$. Aus der DGL erhält man

$$\frac{1}{p} \cdot \frac{\mathrm{d}p}{\mathrm{d}t} = a - bp \quad \overset{p=p_0}{\Rightarrow} \quad 0,02 = a - bp_0 = 0,029 - b \cdot 3,34 \cdot 10^9$$

$$\Rightarrow b = 2,695 \cdot 10^{-12}.$$

Also sind a und b bestimmt. Insbesondere gilt $b \ll a$. Die Erdbevölkerung konvergiert nach obiger Grenzwertbestimmung gegen $10,76 \cdot 10^9$.

1.3. Lösungsmethoden für DGL

Bemerkung: Die Lösungsmethoden für AWAs und RWPs unterscheiden sich signifikant. Hier an der LUH beschränken wir uns in der Numerik II auf AWAs. Im Folgenden werden übersichtsmäßig vier Lösungsmethoden betrachtet. Drei werden wir im Laufe des Semesters ausführlicher diskutieren. Exemplarisch werden AWAs 1. Ordnung diskutiert.

1.3.1. Sukzessive Approximation / Picard-Lindelöf'sches Iterationsverfahren

Die Kernidee dieser Methode ist eine Fixpunktgleichung. Später werden wir auf den Satz 1.4.9 von Picard-Lindelöf stoßen, in dessen Beweis dieses Iterationsverfahren genutzt wird. Sei hier $I := [t_0, t_0 + T]$, wobei $T > 0$ den Endzeitpunkt bezeichne. Jede Lösung $u(t)$ von $u'(t) = f(t, u)$ mit $u(t_0) = u_0$ genügt der Fixpunktgleichung

$$u(t) = u_0 + \int_{t_0}^{t} f\big(s, u(s)\big)\,\mathrm{d}s, \quad t \in I.$$

Daraus lässt sich eine Fixpunktiteration konstruieren. Der Startwert wird als $u^{(0)} \equiv u_0$ festgelegt.

Algorithmus 1.

1. *Startwert:* $u^{(0)} \equiv u_0$
2. *Für* $k = 1, 2, 3, \ldots$
3. $$u(t) = u^{(k)}(t) = u_0 + \int_{t_0}^{t} f\left(s, u^{(k-1)}(s)\right)\,\mathrm{d}s$$

Die Konvergenz der Iteration kann mit Hilfe des Banachschen Fixpunktsatzes gezeigt werden.

BEISPIEL Betrachte $u'(t) = 2t\,u(t) =: f\big(t, u(t)\big)$ mit Anfangswert $u(t_0) = u(0) = 1$. Dann ist der Startwert der Picard-Iteration $u^{(0)} \equiv 1$ und für die ersten Iterationsschritte berechnet man:

$$\begin{aligned}
u^{(1)}(t) &= u_0 + \int_0^t f\big(s, u^{(0)}(s)\big)\,\mathrm{d}s = 1 + \int_0^t 2s\,\mathrm{d}s \\
&= 1 + t^2 \\
u^{(2)}(t) &= u_0 + \int_0^t f\big(s, u^{(1)}(s)\big)\,\mathrm{d}s = 1 + \int_0^t 2s\,(1 + s^2)\,\mathrm{d}s \\
&1 + t^2 + \frac{1}{2}t^4 \\
u^{(3)}(t) &= u_0 + \int_0^t f\big(s, u^{(2)}(s)\big)\,\mathrm{d}s = 1 + \int_0^t 2s\,\big(1 + s^2 + \tfrac{1}{2}s^4\big)\,\mathrm{d}s \\
&= 1 + t^2 + \frac{1}{2}t^4 + \frac{1}{6}t^6
\end{aligned}$$

Hier kann man bereits ein Muster erkennen und so auf die geschlossene Formel

$$u^{(n)}(t) = \sum_{k=0}^{n} \frac{t^{2k}}{k!} \quad \overset{n\to\infty}{\longrightarrow} \quad \exp(t^2)$$

schließen. Obwohl in diesem Fall die Picard-Iteration eine analytische Darstellung liefert, so ist das im Allgemeinen nicht immer der Fall.

1.3.2. Methode der Taylorentwicklung

Idee für dieses Verfahren ist es, $u(t_0 + T)$ per Taylor zu entwickeln. Eine Taylorreihe einer Funktion f um den Punkt x_0 ist durch

$$T_f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}\,(x - x_0)^k$$

gegeben. Voraussetzung ist also, dass f eine analytische Funktion ist. Hier setzt man jetzt $x_0 := t_0$ und $x := t_0 + T$. Dadurch folgt

$$u(t_0 + T) = \sum_{k=0}^{\infty} \frac{u^{(k)}(t_0)}{k!}\,T^k = u^{(0)}(t_0) + \sum_{k=1}^{\infty} \frac{u^{(k)}(t_0)}{k!}\,T^k$$

und durch $u' = f(t,u)$ ergibt sich ferner

$$= u_0 + T \sum_{k=1}^{\infty} \frac{f^{(k-1)}(t_0,u_0)}{k!}\, T^{k-1}.$$

Hier nutzt man nun die Kettenregel

$$f'\big(t,u(t)\big) = f'_t\big(t,u(t)\big)\cdot\frac{\mathrm{d}t}{\mathrm{d}t} + f'_u\big(t,u(t)\big)\cdot\underbrace{\frac{\mathrm{d}u}{\mathrm{d}t}}_{=u'(t)} = f'_t\big(t,u(t)\big) + f'_u\big(t,u(t)\big)\, f\big(t,u(t)\big),$$

um $f^{(k)}\big(t,u(t)\big)$ zu bestimmen. Damit können explizite Lösungen berechnet werden. Allerdings wird die Berechnung der Ableitung $f^{(k)}(t,u)$ im Allgemeinen sehr aufwändig, weshalb das Verfahren in der Praxis kaum genutzt wird. Zusätzlich ist die Taylorentwicklung nur eine lokale Approximation. Für $T \gg t_0$ erhält man daher in der Regel große Fehlerkonstanten und Ungenauigkeiten.

1.3.3. Methode der finiten Differenzen

Explizites Euler-Verfahren Dieses ist ein sehr klassisches Verfahren, das häufig genutzt wird. Die Idee ist die Unterteilung des Zeitintervalls I in diskrete Punkte (vgl. Interpolationsaufgaben oder numerische Quadratur). Ziel ist die Näherungslösung y_h, für die hoffentlich $y_h \approx u(t_h)$ gilt.

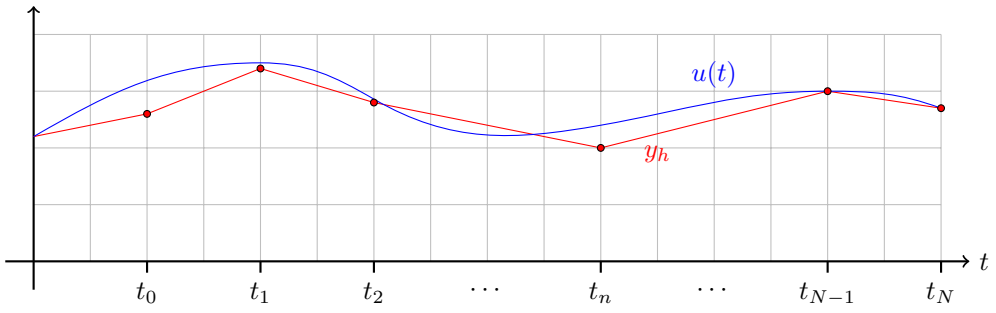


Abbildung 1.2.: Annäherung einer Lösung $u(t)$ durch einen Polygonzug y_h.

Man bildet also einen Polygonzug y_n mit Schrittweise $h_n = t_n - t_{n-1}$, der sich der Lösung $u(t)$ annähern soll. Dazu dient das Eulersche Polygonzugverfahren. Hier bildet man die approximative Ableitung $u'(t_n)$ mit dem Differenzquotienten:

$$u'(t_n) \approx y'_n \approx \frac{y_n - y_{n-1}}{h_n}.$$

Man arbeitet dann also mit

$$\frac{y_n - y_{n-1}}{h_n} = f(t_{n-1}, y_{n-1}).$$

Auflösen nach y_n liefert einem daraufhin

$$y_n = y_{n-1} + h_n\, f(t_{n-1}, y_{n-1}).$$

Man erhält damit den Algorithmus (Polygonzugverfahren)

Algorithmus 2.

1. *Startwert:* $y_0 := u_0$
2. *Für* $n = 1, 2, 3, \ldots, N$
3. $\quad y_n = y_{n-1} + h_n\, f(t_{n-1}, y_{n-1})$

Beispiel Sei $u'(t) = a\, u(t)$ gegeben mit $a\, u(t) =: f(t, u)$. Seien $a = 2$, $b_0 = 1$, $y_0 = u(t_0) = 3$ und $h_n = 4$. Dann ist der erste Iterationsschritt des Eulerschen Polygonzugverfahrens

$$y_1 = y_0 + h_0\, a\, y_0 = 3 + 4 \cdot 2 \cdot 3 = 27$$

Implizites Euler-Verfahren Die oben beschriebene Variante ist die explizite des Euler-Verfahrens. Nun wird hier die entsprechende implizite Form vorgestellt. Wie zuvor auch, will man

$$u'(t_n) \approx \frac{y_n - y_{n-1}}{h_n}.$$

Nun aber nehme man die rechte Seite zu t_n, anstatt zu t_{n-1}, also betrachte man $f(t_n, y_n)$. Daraus erhält man dann den Algorithmus zum impliziten Eulerverfahren

Algorithmus 3.

1. $y_0 := u_0$
2. *Für* $n = 1, \ldots, N$
3. $\quad y_n = y_{n-1} + h_n\, f(t_n, y_n)$

Hier sollte noch erwähnt sein, dass y_n auf der rechten Seite als Funktionsargument auftaucht und im Allgemeinen kann $f(t_n, y_n)$ auch nicht nach y_n aufgelöst werden! Häufig wird y_n darüber berechnet, dass die Gleichung als Nullstellenproblem umgestellt und dann über Fixpunktiteration oder Newton-Verfahren gelöst wird.

Trapezregel Neben den Euler-Verfahren gibt es auch die Trapezregel. Diese ist eine Kombination aus implizitem und explizitem Euler-Verfahren. Aus der Symmetrie erhält man mit diesem

Verfahren eine höhere Konvergenzordnung bzgl. der Schrittweite h, was später bewiesen wird. Man konstruiert sie wie folgt:

$$\frac{y_n - y_{n-1}}{h_n} = \frac{1}{2}\big(\underbrace{f(t_{n-1}, y_{n-1})}_{=f_{n-1}} + \underbrace{f(t_n, y_n)}_{=f_n}\big) \quad \Rightarrow \quad y_n = y_{n-1} + \frac{h_n}{2}(f_{n-1} + f_n)$$

Damit erhält man den Algorithmus

Algorithmus 4.

1. $y_0 := u_0$
2. ***Für*** $n = 1, \ldots, N$
3. $\quad y_n = y_{n-1} + \dfrac{h_n}{2}\big(f(t_{n-1}, y_{n-1}) + f(t_n, y_n)\big)$

1.3.4. Galerkin-Methoden

Die Galerkin-Methode unterscheidet sich deutlich von den zuvor genannten. Die Hintergrundidee ist die variationelle Formulierung der DGL über eine Integraldarstellung und geeignete Funktionenräume. Letztendlich lässt sich das Galerkin-Verfahren oftmals auch wieder wie Finite Differenzen realisieren. Bei der Galerkin-Methode wird die DGL mit einer sogenannten Testfunktion aus einem Funktionenraum V multipliziert und anschließend über $I := [\,0,\, t_0 + T\,]$ integriert. In den Finiten Differenzen Methoden aus (1.3.3) wurde die Ableitung per Differenenzenquotienten approximiert, das Galerkin-Verfahren hingegen approximiert den kontinuierlichen Funktionenraum durch Folgen von diskreten Funktionenräumen. Letztendlich führt das Ganze zu einem Gleichungssystem, das mit bekannten Methoden der Numerik I gelöst werden kann. Die Galerkin-Methode ist sehr umfangreich, weshalb wir dem ein eigenes Kapitel (S. 84) widmen. Hier geben wir deshalb nur eine kurze Herleitung der Methode. Gegeben sei $u'(t) = f(t, u)$. Dann ist: Finde $u \in V$, sodass

$$\int_I u'(t)\,\varphi(t)\,\mathrm{d}t = \int_I f(t,u)\,\varphi(t)\,\mathrm{d}t, \text{ für alle zulässigen Testfunktionen } \varphi \in V;$$

die Darstellung der Galerkin-Methode. Diese Darstellung gilt für stetige und stückweise stetig differenzierbare Funktionen. Die stückweise Differenzierbarkeit bis auf endlich viele Ausnahmestellen impliziert, dass das linke Integral in eine Summe von Einzelintegralen zerfällt. Die sogenannte (stetige) Galerkin-Methode bestimmt eine Näherungslösung u_h in einem endlich-dimensionalen Teilraum $V_h \subset V$. Man legt also zunächst $u_h(t_0) = u_0$ fest und berechnet dann $u_h \in V_h$ so, dass

$$\int_I u_h'(t)\,\varphi_h(t)\,\mathrm{d}t = \int_I f\big(t, u_h(t)\big)\,\varphi_h(t)\,\mathrm{d}t \quad \forall \varphi_h \in W_h$$

gilt, wobei W_h der diskrete Funktionenraum der Testfunktion bedeutet.[4] Grundlegend wird ab hier das Intervall I in Teilintervalle aufgeteilt, worauf jeweils Polynome als Ansatzfunktionen und Testfunktionen dienen, sprich die Lösung soll eine stückweise Polynom-Interpolation sein. Ein Beispiel für den Raum der Ansatzfunktion V_h und den Raum der Testfunktion W_h wären

$$V_h := \left\{ v_h : I \to \mathbb{R} \;\middle|\; v_h \in C(I),\ v_h|_{(t_{n-1},t_n)} \in P_1,\ n = 1, \dots, N \right\}$$
$$W_h := \left\{ \varphi_h : I \to \mathbb{R} \;\middle|\; \varphi_h|_{(t_{n+1},t_n)} \in P_0,\ n = 1, \dots, N \right\}$$

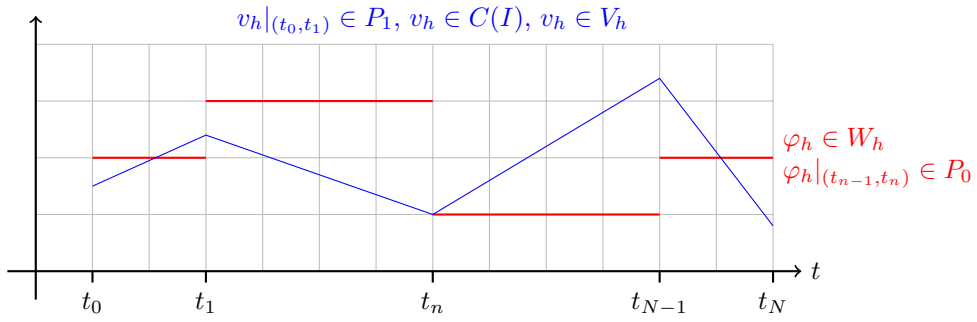


Abbildung 1.3.: Visualisierung von beispielhaften V_h und W_h.

Testfunktionen sind offensichtlich nur stückweise stetig, d. h. man kann das Problem auf jedem Teilintervall $(t_{n-1},\, t_n]$ gesondert betrachten, was eine erhebliche Vereinfachung bedeutet. Wählt man hier in etwa $\varphi_h \in W_h$ derart, dass $\varphi_h(t) = 1$ auf $(t_{n-1},\, t_n]$ und $\varphi_h(t) = 0$ sonst gilt, so erhält man das sequentielle Verfahren

$$u_h(t_n) - u_h(t_{n-1}) = \int_{t_{n-1}}^{t_n} u_h'(t)\,\mathrm{d}t = \int_{t_{n-1}}^{t_n} f(t,u)\,\mathrm{d}t \quad \forall n = 1, \dots, N.$$

Löst man das rechte Integral über eine Quadraturformel, bspw. der Trapezregel, so erhält man

$$u_h(t_n) = u_h(t_{n-1}) + \int_{t_{n-1}}^{t_n} f(t,u_h)\,\mathrm{d}t \approx u_h(t_{n-1}) + \frac{1}{2}\Big(f\big(t_{n-1}, u_h(t_{n-1})\big) + f\big(t_n, u_h(t_n)\big)\Big).$$

Das ist gerade die (zur Quadraturformel gleichnamige) Trapezregel aus dem Differenzenverfahren. Die Herleitung über Finite Differenzen war sehr viel schneller, weshalb diese eher in der Praxis auftauchen. Die Galerkin-Methode hat eine größere mathematische Allgemeinheit und liefert eine reichhaltigere numerische Analyse, was Teil der Numerik III sein wird.

Bemerkung: Im Allgemeinen können durch die Wahl von V_h und W_h systematisch numerisch-

[4] Dieser diskrete Funktionenraum der Testfunktion muss nicht notwendigerweise mit V_h übereinstimmen.

stabile und höher-konvergente Verfahren zur Lösung von DGL konstruiert werden.

1.4. Theorie für Anfangswertaufgaben (AWAs)

1.4.1. Existenzsätze

Betrachte $u'(t) = f\big(t, u(t)\big)$ mit Vektorfunktionen

$$u(t) = \begin{pmatrix} u_1(t) \\ \vdots \\ u_d(t) \end{pmatrix},\ f(t,u) = \begin{pmatrix} f_1(t,u) \\ \vdots \\ f_d(t,u) \end{pmatrix}.$$

Man wähle einen Anfangspunkt $(t_0, u_0) \in \mathbb{R}^1 \times \mathbb{R}^d$. Hierzu suche man eine Lösung $u(t)$ im Intervall $I := [\, t_0,\, t_0 + T\,]$ mit $u(t_0) = u_0$. Die Funktion $f(t,u)$ sei auf $D = I \times \Omega \subset \mathbb{R}^1 \times \mathbb{R}^d$ definiert und dort stetig. Sei weiterhin

$$\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i$$

das euklidische Skalarprodukt und $\|x\| := \sqrt{\langle x, x \rangle}$ mit $x, y \in \mathbb{R}^d$ die davon induzierte Norm.

Für Ableitungen werden die folgenden Notationen benutzt:

$$u'(t) = \frac{\mathrm{d}u}{\mathrm{d}t}, \quad f'_t(t,u) = \frac{\partial f(t,u)}{\partial t}, \quad \partial_i f(t,u) = \frac{f}{\partial u_i}$$

Definition 1.4.1 Anfangswertaufgabe (AWA)

Sei $(t_0, u_0) \in D$. Berechne eine stetig differenzierbare Funktion $u : I \to \mathbb{R}^d$ mit den Eigenschaften:

(1) $\mathrm{Graph}(u) := \big\{ \big(t, u(t)\big) \;\big|\; t \in I \big\} \subset D$

(2) $u'(t) = f\big(t, u(t)\big),\ t \in I$

(3) $u(t_0) = u_0$

Nach dem bekannten Hauptsatz der Integral- und Differentialrechnung ist eine stetige Funktion $u : I \to \mathbb{R}^d$ genau dann eine Lösung der AWA, wenn $\mathrm{Graph}(u) \subset D$ gilt und die Integralgleichung

$$u(t) = u_0 + \int_{t_0}^{t} f\big(s, u(s)\big)\,\mathrm{d}s,\ t \in I$$

erfüllt ist. Eine mathematische Aufgabe ist wohlgestellt, wenn eine eindeutige Lösung mit stetiger Abhängigkeit von den Daten existiert. So erhält man nun das erste theoretische Resultat, die lokale Existenz von Lösungen.

Satz 1.4.2 Satz von Peano

Sei $f\big(t,u(t)\big)$ stetig auf $D=\big\{\big(t,u(t)\big)\big\}\in\mathbb{R}^1\times\mathbb{R}^d$. Seien $|t-t_0|\le\alpha$ und $\|u-u_0\|\le\beta$. Dann existiert eine Lösung u der Anfangswertaufgabe auf dem Intervall $I=[t_0-T,\,t_0+T]$ mit $T:=\min\Big(\alpha,\frac{\beta}{M}\Big)$, $M:=\max_{(t,u)\in D}\|f(t,u)\|$.

Zu beachten ist, dass T unter Umständen sehr klein sein kann. Darum behandelt der Satz von Peano auch nur lokale Existenz.

Beweis zu Satz 1.4.2
Man nutze das Eulersche Polygonzugverfahren aus (1.3), also

$$u_n^h=u_{n-1}^h+h\,f(t_{n-1},u_{n-1}^h)\quad\Rightarrow\quad u^h(t)=u_{n-1}^h+(t-t_{n-1})\,f(t_{n-1},u_{n-1}^h)$$

für $t_{n-1}\le t\le t_n$. Wir zeigen nun, dass diese Lösungskonstruktion durchführbar ist, d. h. dass $\mathrm{Graph}(u^h)\subset D$ gilt. Dazu sei $\big(t,u^h(t)\big)\in D$ für $t_0\le t\le t_{k-1}$. Dann gilt per Konstruktion für $t\in[t_{k-1},\,t_k]$:

$$\begin{aligned}u^h(t)-u_0&=u^h(t)-u_{k-1}^h+\underbrace{\sum_{i=1}^{k-1}\overbrace{[u_i^h-u_{i-1}^h]}^{=h\,f(t_{i-1},u_{i-1}^h)}}_{\text{Teleskopsumme}}\\&=(t-t_{k-1})\underbrace{f(t_{k-1},u_{k-1}^h)}_{\|\cdot\|\le M}+h\sum_{i=1}^{k-1}f(t_{i-1},u_{i-1}^h)\end{aligned}$$

und folglich

$$\|u^h(t)-u_0\|\le(t-t_{k-1})M+(t_{k-1}-t_0)M=(t-t_0)M\le\beta.$$

Also ist $\big(t,u^h(t)\big)\in D$ für $t_0\le t\le t_k$. Durch Induktion folgt $\mathrm{Graph}(u^h)\subset D$.

Nun wird gezeigt, dass u stetig ist. Seien dazu $t,t'\in I$, $t'\le t$ beliebig mit $t\in[t_{k-1},\,t_k]$, $t'\in[t_{j-1},\,t_j]$ für gewisse $t_j\le t_k$. Im Fall $t,t'\in[t_{k-1},\,t_k]$, also für $j=k$, ist

$$\begin{aligned}u^h(t)-u^h(t')&=u_{k-1}^h+(t-t_{k-1})\,f\big(t_{k-1},u^h(t_{k-1})\big)-u_{k-1}^h-(t'-t_{k-1})\,f\big(t_{k-1},u^h(t_{k-1})\big)\\&=(t-t')\,f\big(t_{k-1},u^h(t_{k-1})\big)\end{aligned}$$

und somit $\|u^h(t) - u^h(t')\| \le M|t - t'|$. Im Fall $t_j < t_k$ ist

$$\begin{aligned} u^h(t) - u^h(t') &= u^h(t) - u^h_{k-1} + \sum_{i=j}^{k-1} [u^h_i - u^h_{i-1}] + u^h_{j-1} - u^h(t') \\ &= (t - t_{k-1})\, f(t_{k-1}, u^h_{k-1}) + h \sum_{i=j}^{k-1} f(t_{i-1}, u^h_{i-1}) + (t_{j-1} - t')\, f(t_{j-1}, u^h_{j-1}) \\ &= (t - t_{k-1})\, f(t_{k-1}, u^h_{k-1}) + h \sum_{i=j+1}^{k-1} f(t_{i-1}, u^h_{i-1}) + (h + t_{j-1} - t')\, f(t_{j-1}, u^h_{j-1}) \end{aligned}$$

und folglich $\|u^h(t) - u^h(t')\| \le M\big[(t - t_{k-1}) + (t_{k-1} - t_j) + (t_j - t')\big] \le M|t - t'|$. Damit ist die Stetigkeit gezeigt.

Es bleibt zu zeigen, dass die Grenzwertfunktion u die Integralgleichung erfüllt. Man setze $u^i(t) := u^{h_i}(t)$ für $t \in [t_{k-1},\, t_k] \subset I$. Für jedes i gilt zunächst

$$\begin{aligned} u^i(t) &= u^i_{k-1} + (t - t_{k-1})\, f(t_{k-1}, u^i_{k-1}) \\ &= u^i_{k-2} + (t_{k-1} - t_{k-2})\, f(t_{k-2}, u^i_{k-1}) + (t - t_{k-1})\, f(t_{k-1}, u^i_{k-1}) \\ &\;\;\vdots \\ &= u_0 + \sum_{j=1}^{k-1} (t_j - t_{j-1})\, f(t_{j-1}, u^i_{j-1}) + (t - t_{k-1})\, f(t_{k-1}, u^i_{k-1}) \\ &= u_0 + \sum_{j=1}^{k-1} \int_{t_{j-1}}^{t_j} f(t_{j-1}, u^i_{j-1})\, \mathrm{d}s + \int_{t_{k-1}}^{t} f(t_{k-1}, u^i_{k-1})\, \mathrm{d}s \\ &= u_0 + \sum_{j=1}^{k-1} \int_{t_{j-1}}^{t_j} \Big[f(t_{j-1}, u^i_{j-1}) - f\big(s, u^i(s)\big) \Big] \mathrm{d}s \\ &\qquad + \int_{t_{k-1}}^{t} \Big[f(t_{k-1}, u^i_{k-1}) - f\big(s, u^i(s)\big) \Big] \mathrm{d}s + \int_{t_0}^{t} f\big(s, u^i(s)\big)\, \mathrm{d}s. \end{aligned}$$

Auf der kompakten Menge D ist die stetige Funktion $f(t, u)$ auch gleichmäßig stetig. Ferner sind die Funktionen der Folge $(u^i)_i$ gleichgradig stetig. Zu beliebig gegebenen $\varepsilon > 0$ gibt es also $\delta, \varepsilon' > 0$, sodass für $|t - t'| < \delta$ und $\|u - u'\| < \varepsilon'$

$$\|u^i(t) - u^i(t')\| \le \varepsilon', \qquad \|f(t, u) - f(t', u')\| < \varepsilon$$

gilt. Für hinreichend großes $i \geq i_\varepsilon$, d. h. hinreichend kleines h_i, folgt damit

$$\max_{s \in [t_{k-1},\, t_k]} \left\| f(t_{k-1}, u^i(t_{k-1})) - f\big(s, u^i(s)\big) \right\| \leq \varepsilon,\ k = 1, \ldots, n.$$

Dies ergibt

$$\left| u^i(t) - u_0 - \int_{t_0}^{t} f\big(s, u^i(s)\big)\, \mathrm{d}s \right| \leq \varepsilon |t - t_0|.$$

Die gleichmäßige Konvergenz $u^i \to u$ auf I impliziert auch die gleichmäßige Konvergenz

$$f\big(\,\cdot\,, u^i(\,\cdot\,)\big) \to f\big(\,\cdot\,, u(\,\cdot\,)\big),\ i \to \infty.$$

Für $i \to \infty$ ergibt sich demnach

$$\left| u(t) - u_0 - \int_{t_0}^{t} f\big(s, u(s)\big)\, \mathrm{d}s \right| \leq \varepsilon |t - t_0|.$$

Wegen der beliebigen Wahl von ε folgt, dass die Integralgleichung von der Grenzwertfunktion u gelöst wird. □

Satz 1.4.3 Fortsetzungssatz

Falls $f(t, u)$ stetig auf D in $\mathbb{R}^1 \times \mathbb{R}^d$ ist, dann lässt sich die Lösung über das Intervall $[t_0 - T,\, t_0 + T]$ auf einen Bereich $[t_0 - T^*,\, t_0 + T^*]$ fortsetzen.

Der Beweis wird hier ausgelassen, kann bei Interesse aber in (Rannacher 2017, S. 17–18) nachgelesen werden.

Korollar 1.4.4 globale Existenz auf ganz $\mathbb{R}^1 \times \mathbb{R}^d$

Sei $f(t, u)$ auf ganz $\mathbb{R}^1 \times \mathbb{R}^d$ stetig. Es gelte die Abschätzung

$$\|u(t)\| \leq \beta(t),\ t \in [t_0 - T,\, t_0 + T]$$

mit festem und stetigem β. Dann lässt sich $u(t)$ auf ganz $\mathbb{R}$ fortsetzen.

Beweis zu Korollar 1.4.4
Wegen der zu erfüllenden Abschätzung kann keine der möglichen Lösungen der AWA eine unbeschränkte Lösung auf $D \subset \mathbb{R}^1 \times \mathbb{R}^d$ sein. Daher impliziert der Fortsetzungssatz, dass $u(t)$ global

auf $\mathbb{R}^1 \times \mathbb{R}^d$ existiert. $\square$

Beispiele

(a) Gegeben sei die skalare AWA $u'(t) = \sin\big(u(t)\big)$ mit $t \geq 0$ und $u(0) = 0$. Es gibt nach Peano lokale Lösungen, da $f(t,u) = \sin\big(u(t)\big)$ stetig ist. Ferner gibt es nach Korollar (1.4.4) sogar globale Lösungen, da $u(t)$ wie folgt beschränkt werden kann:

$$u(t) = u(0) + \int_0^t f\big(s, u(s)\big)\,\mathrm{d}s = u(0) + \int_0^t \sin\big(u(s)\big)\,\mathrm{d}s$$

$$\Rightarrow \quad |u(t)| \leq |u(0)| + \int_0^t \big|\sin\big(u(s)\big)\big|\,\mathrm{d}s \leq 0 + t$$

$$\Rightarrow \quad \|u(t)\| \leq t =: \beta(t).$$

(b) Gegeben sei $u'(t) = a\,u(t)$ für $t \geq 0$ mit $t_0 = 0$, $u(0) = u_0$, $a \in \mathbb{R}$. Es gibt eine eindeutige globale Lösung $u(t) = u_0 \exp(at)$. Hier unterscheide man drei Fälle:

- $a > 0$: $|u(t)| \to \infty$ $(t \to \infty)$.
- $a = 0$: $|u(t)| = u_0$ $(t \to \infty)$.
- $a < 0$: $|u(t)| \to 0$ $(t \to \infty)$.

1.4.2. Eindeutigkeit und Stabilität von Lösungen

Stabilität meint hier stabile Lösungen auf dem kontinuierlichen Level für $u(t)$. Hier ist also keine Numerik im Spiel! Das entscheidende Hilfsmittel ist die Lipschitz-Stetigkeit der rechten Seite $f(t,u)$ der AWA.

Definition 1.4.5 Lipschitz-Bedingung

(i) Die Funktion $f(t,u)$ genügt in $D \subset \mathbb{R}^1 \times \mathbb{R}^d$ einer gleichmäßigen Lipschitz-Bedingung, falls mit einer stetigen Funktion $L(t) > 0$ (Lipschitz-Konstante)

$$\|f(t,u) - f(t,\tilde{u})\| \leq L(t)\|u - \tilde{u}\|$$

mit $(t,u), (t,\tilde{u}) \in D$ gilt.

(ii) Die Funktion $f(t,u)$ genügt einer lokalen Lipschitz-Bedingung, falls diese in einer beschränkten Teilmenge von D gilt.

Satz 1.4.6 Lokaler Stabilitätssatz

Es seien $f(t,u)$ und $g(t,v)$ auf D definiert und stetig. Des Weiteren seien die folgenden AWAs gegeben:

$$u'(t) = f(t,u),\ t \in I,\ u(t_0) = u_0$$
$$v'(t) = g(t,v),\ t \in I,\ v(t_0) = v_0$$

Die Funktion $f(t,u)$ genügt der Lipschitz-Bedingung mit $\sup_{t\in I} L(t) < \infty$. Dann gilt für zwei beliebige Lösungen $u(t)$ und $v(t)$ die Abschätzung

$$\|u(t) - v(t)\| \leq \exp\big(L(t-t_0)\big) \cdot \left(\|u_0 - v_0\| + \int_{t_0}^{t} \varepsilon(s)\,\mathrm{d}s \right)$$

mit $\varepsilon(t) := \sup_{u\in\Omega} \|f(t,u) - g(t,u)\|$.

Bemerkung: Die Abschätzung aus dem lokalen Stabilitätssatz ist eine sogenannte a priori-Stabilitätsabschätzung. Das Ziel sollte immer $\|u(t) - v(t)\| \approx 0$ sein (dritte Bedingung von Hadamard).

Beispiel Sei $f \equiv g$. Dann gilt

$$\|u(t) - v(t)\| \leq \exp\big(L(t-t_0)\big) \cdot \|u_0 - v_0\|.$$

Die Fortsetzung des Unterschieds zwischen u_0 und v_0 wird daher wesentlich durch den Faktor $\exp\big(L(t-t_0)\big)$ bestimmt. Insbesondere gilt für $t \to \infty$ auch $\exp\big(L(t-t_0)\big) \to \infty$. D. h. für große t wird die Stabilitätsabschätzung unbrauchbar.

Beweis zu Satz 1.4.6
Setze $e(t) := u(t) - v(t)$. Dann ist

$$\begin{aligned}
e(t) &= \int_{t_0}^{t} \Big(f\big(s,u(s)\big) - g\big(s,v(s)\big) \Big)\,\mathrm{d}s + u_0 - v_0 \\
&= \int_{t_0}^{t} \Big(f\big(s,u(s)\big) - f\big(s,v(s)\big) + f\big(s,v(s)\big) - g\big(s,v(s)\big) \Big)\,\mathrm{d}s + u_0 - v_0 \\
&= \int_{t_0}^{t} \Big(f\big(s,u(s)\big) - f\big(s,v(s)\big) \Big)\,\mathrm{d}s + \int_{t_0}^{t} \Big(f\big(s,v(s)\big) - g\big(s,v(s)\big) \Big)\,\mathrm{d}s + u_0 - v_0
\end{aligned}$$

Daraus folgt nun

$$\|e(t)\| \leq \underbrace{\int_{t_0}^{t} |f(s,u) - f(s,v)| \,\mathrm{d}s}_{\leq L\|u-v\|} + \underbrace{\int_{t_0}^{t} |f(s,v) - g(s,v)| \,\mathrm{d}s}_{\leq \sup_{v\in\Omega}\|f(s,v)-g(s,v)\|} + \|u_0 - v_0\|$$

$$\leq \int_{t_0}^{t} L\|u - v\| \,\mathrm{d}s + \|u_0 - v_0\| + \int_{t_0}^{t} \sup_{v\in\Omega} \|f(s,v) - g(s,v)\| \,\mathrm{d}s.$$

Seien $w(t) := \|u - v\|$, $a := L$ und $b := \|u_0 - v_0\|$. Dann gilt das Gronwall-Lemma, wodurch der vordere Teil nach oben durch

$$\leq \exp\big(L(t - t_0)\big)\left(\|u_0 - v_0\| + \int_{t_0}^{t} \sup_{v\in\Omega} \|f(s,v) - g(s,v)\| \,\mathrm{d}s\right)$$

abgeschätzt werden kann. Damit ist alles gezeigt. □

Satz 1.4.7 Gronwall-Lemma

Die stückweise stetige Funktion $w(t) \geq 0$ genüge mit zwei Konstanten $a, b \geq 0$ der Integralungleichung

$$w(t) \leq a \int_{t_0}^{t} w(s) \,\mathrm{d}s + b, \; t \geq t_0.$$

Dann gilt $w(t) \leq \exp\big(a(t - t_0)\big) \cdot b$.

Beweis zu Satz 1.4.7
Man setze

$$\psi(t) := a \int_{t_0}^{t} w(s) \,\mathrm{d}s + b.$$

Es gilt $\psi'(t) = a\, w(t)$ und nach Voraussetzung $w(t) \leq \psi(t)$. D. h. es gilt $\psi'(t) \leq a\, \psi(t)$. Damit erhält man über die Multiplikation mit $\exp(-at)$

$$\big(\exp(-at)\, \psi(t)\big)' = -a \exp(-at)\, \psi(t) + \exp(-at)\, \psi'(t) = \underbrace{\exp(-at)}_{>0} \underbrace{\big[\psi'(t) - a\, \psi(t)\big]}_{<0} \leq 0.$$

Damit ist $\exp(-at)\,\psi(t)$ monoton fallend. Somit folgt aus der Voraussetzung $w(t) \leq \psi(t)$ und der Monotonie

$$\exp(-at)\,w(t) \leq \exp(-at)\,\psi(t) \leq \exp(-at_0)\,\underbrace{\psi(t_0)}_{=:b} = b\exp(-at_0),\ t \geq t_0.$$

Aus der Multiplikation mit $\exp(at)$ folgt dann

$$w(t) \leq b\exp\big(a(t-t_0)\big),\ t \geq t_0.$$ □

Korollar 1.4.8 Eindeutigkeit

Der Stabilitätssatz liefert Eindeutigkeit.

Beweis zu Korollar 1.4.8
Gäbe es zwei Lösungen $u(t)$ und $v(t)$, so gälten die beiden AWAs. Dann ist $f \equiv g$ und $u_0 \equiv v_0$. Stabilitätssatzabschätzung liefert

$$\|u(t) - v(t)\| \leq \exp\big(L(t-t_0)\big)\left(\|u_0 - v_0\| + \int_{t_0}^{t} \varepsilon(s)\,\mathrm{d}s\right) = 0.$$

Die Gleichheit zu 0 resultiert aus $\|u_0 - v_0\| = 0$, da $u_0 \equiv v_0$, und weil das Integral ebenfalls gleich 0 ist, da $f \equiv g$. □

Nun kommen wir zu dem Satz, den wir in Kapitel 1.3.1 bereits angesprochen hatten.

Satz 1.4.9 Picard-Lindelöf

Die stetige Funktion $f : D = I \times \Omega \to \mathbb{R}^d$ genüge einer lokalen Lipschitz-Bedingung. Dann existieren zu jedem Startwert $(t_0, u_0) \in D$ ein $T > 0$ und eine Lösung

$$u : [\,t_0 - T,\, t_0 + T\,] \to \mathbb{R}^d,$$

sodass die AWA

$$u'(t) = f(t,u),\ u(t_0) = u_0$$

erfüllt ist. Diese lokale Lösung ist eindeutig bestimmt.

Eigentlich liegt bereits alles zur Wohlgestelltheit von AWAs vor, die Existenz nach Peano, die Stabilität aus dem Stabilitätssatz und die Eindeutigkeit als Folgerung des Stabilitätssatzes. Der Satz von Picard-Lindelöf beweist alle drei Eigenschaften in einem einzigen Satz mit einer Beweistechnik, die auf dem Banach'schen Fixpunktsatz basiert.

Beweis zu Satz 1.4.9
Hier nutzen wir den Ansatz der Picard-Iteration, nämlich

$$u(t) = u_0 + \int_{t_0}^{t} f\big(s, u(s)\big)\,\mathrm{d}s,$$

und wollen zeigen, dass wir damit auf einem geschickt gewählten Banachraum eine Kontraktion definiert haben. Damit können wir dann den Banachschen Fixpunktsatz anwenden.

(i) Für $\delta > 0$ definiere man

$$K := \big\{(t,x) \in \mathbb{R} \times \mathbb{R}^d \;\big|\; |t - t_0| \le \delta,\ \|x - u_0\| \le \delta\big\}.$$

Nun gibt es ein $\delta > 0$ derart, dass $K \subset D$ gilt und f auf K eine Lipschitz-Bedingung mit Konstante L_K erfüllt. D. h.

$$\|f(t,x) - f(t,y)\| \le L_K \,\|x - y\|, \quad (t,x),(t,y) \in K.$$

Über Heine-Borel deduziert man, dass K eine kompakte Menge ist. Ferner ist f stetig auf K, somit gibt es ein $M > 0$ mit $\|f(t,x)\| \le M$. Es sei $T := \min\big\{\delta, \frac{\delta}{M}, \frac{1}{2L_K}\big\}$ und $I_0 := [\,t_0 - T,\, t_0 + T\,]$. Darüber definiere man

$$V := C\big([\,t_0 - T,\, t_0 + T\,]\big), \qquad \|\cdot\|_V := \max_{t \in I_0} \|u(t)\|,$$
$$V_0 := \big\{v \in V \;\big|\; \max_{t \in I_0} \|v(t) - u_0\| \le \delta\big\} \subset V$$

$(V, \|\cdot\|_V)$ ist damit ein Banachraum und V_0 ist eine abgeschlossene Teilmenge. Damit ist V_0 ein vollständiger metrischer Raum, welcher für den Fixpunktsatz genutzt wird.

(ii) Auf V_0 definiere man die Abbildung

$$g : V_0 \to V_0, \quad g(u)(t) = u_0 + \int_{t_0}^{t} f\big(s, u(s)\big)\,\mathrm{d}s.$$

Für ein $u \in V_0$ gilt

$$\|g(u)(t) - u_0\| \leq \int_{t_0}^{t} \big\|f\big(s, u(s)\big)\big\| \,\mathrm{d}s \leq (t - t_0)M \leq MT \leq \delta$$

Demnach ist $g(u) \in V_0$, daher ist g eine Selbstabbildung auf V_0. Über die Lipschitz-Stetigkeit von $f(t, \cdot\,)$ folgt zudem für $u, v \in V_0$

$$\begin{aligned}
\|g(u)(t) - g(v)(t)\| &= \left\| u_0 + \int_{t_0}^{t} f\big(s, u(s)\big) \,\mathrm{d}s - u_0 - \int_{t_0}^{t} f\big(s, v(s)\big) \,\mathrm{d}s \right\| \\
&= \left\| \int_{t_0}^{t} f\big(s, u(s)\big) - f\big(s, v(s)\big) \,\mathrm{d}s \right\| \\
&\leq \int_{t_0}^{t} \big\|f\big(s, u(s)\big) - f\big(s, v(s)\big)\big\| \,\mathrm{d}s \\
&\leq (t - t_0) L_K \|u - v\| \leq \underset{\substack{\uparrow \\ T = \min\left\{\cdots, \frac{1}{2L_K}\right\}}}{TL_K} \|u - v\| \leq \frac{1}{2} \|u - v\|.
\end{aligned}$$

Damit ist gezeigt: g ist eine Kontraktion.

(iii) Nach dem Banachschen Fixpunktsatz existiert für eine kontrahierende Funktion g auf einem vollständigen metrischen Raum V_0 genau ein $u^* \in V_0$ mit $g(u^*) = u^*$. Für diesen Fixpunkt impliziert der Hauptsatz der Differential- und Integralrechnung dann $u^{*\prime}(t) = f\big(t, u^*(t)\big)$ und weiterhin gilt $u^*(t_0) = u_0$. Der Fixpunkt ist also äquivalent zu der AWA, liefert also eine eindeutige lokale Lösung. □

1.5. Einschrittmethoden

Wir haben bereits vier Verfahren kennengelernt.

(a) Expliziter Euler (Polygonzug): $y_n = y_{n-1} + h_n \, f(t_{n-1}, y_{n-1})$

(b) Impliziter Euler: $y_n = y_{n-1} + h_n \, f(t_n, y_n)$

(c) Trapezregel: $y_n = y_{n-1} + \frac{h_n}{2}\big[f(t_n, y_n) + f(t_{n-1}, y_{n-1})\big]$

(d) Mittelpunktsregel: $y_n = y_{n-1} + h_n \, f\big(t_n + \frac{1}{2}h_n, \frac{1}{2}(y_n + y_{n-1})\big)$

1.5.1. Die Eulersche Polygonzugmethode

Man betrachte eine AWA der Form

$$u'(t) = f(t,u), \quad t \in I := [\,t_0,\, t_0+T\,], \quad u(t_0) = u_0.$$

Des Weiteren sei $f(t,u)$ stetig auf $I \times \mathbb{R}^d$ und genüge einer globalen Lipschitz-Bedingung

$$\|f(t,u) - f(t,\tilde{u})\| \le L\|u-\tilde{u}\|, \quad (t,u),\ (t,\tilde{u}) \in I \times \mathbb{R}^d.$$

Aus einem diskreten Startwert $y_0^h \in \mathbb{R}^d$ konstruiere man eine Folge $(y_n^h)_{n\in\mathbb{N}}$ mittels der Rekursion

$$y_n^h = y_{n-1}^h + h_n\, f(t_{n-1}, y_{n-1}^h), \quad n = 1,\dots,N, \qquad \text{(expliziter Euler)}$$

wobei $h_n := t_n - t_{n-1}$ die Schrittweite bezeichne. Das Ganze kann auch als Differenzengleichung

$$(L_h y^h)_n = 0, \quad n = 1,\dots,N$$

geschrieben werden mit dem Differenzenoperator

$$(L_h y^h)_n := \frac{1}{h_n}(y_n^h - y_{n-1}^h) - f(t_{n-1}, y_{n-1}^h)$$

für die sogenannten Gitterfunktionen $y^h := \{y_n^h\}_{n=1,\dots,N}$, welche insbesondere von h (der Gitterweite) abhängen. Zur Bestimmung — oder allgemeiner: zur Abschätzung — der Konvergenzgeschwindigkeit eines Diskretisierungsverfahrens (hier: Polygonzug) nutzen wir den sogenannten Abschneidefehler (auch lokaler Diskretisierungsfehler). Der Abschneidefehler beschreibt den Fehler, der gemacht wird, wenn die exakte Lösung u_n^h in das numerische Verfahren eingesetzt wird. Ausgehend vom Differenzenoperator erhält man

$$\tau_n^h := (L_h u^h)_n = \frac{1}{h_n}(u_n^h - u_{n-1}^h) - f(t_{n-1}, u_{n-1}^h). \tag{1}$$

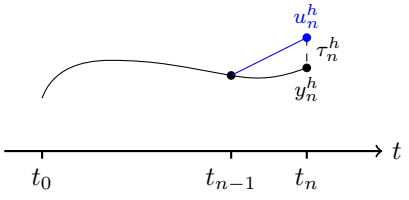


Abbildung 1.4.: Beispiel zum Abschneidefehler des Polygonzugverfahrens.

Die Bestimmung der Konvergenzgeschwindigkeit erfolgt dann mittels τ_n^h des Polygonzugver-

fahrens.[5] D. h.

$$\tau_n^h = \frac{1}{h_n}(u_n^h - u_{n-1}^h) - f(t_{n-1}, u_{n-1}^h) = \frac{1}{h_n}(u_n^h - u_{n-1}^h) - u'(t_{n-1})$$

und mit dem Hauptsatz der Integral- und Differentialrechnung erhält man

$$= \frac{1}{h_n} \int_{t_{n-1}}^{t_n} u'(t)\,\mathrm{d}t - u'(t_{n-1}) = \frac{1}{h_n} \int_{t_{n-1}}^{t_n} (t_n - t)\,u''(t)\,\mathrm{d}t.$$

Daraus folgt nun

$$\|\tau_n^h\| = \frac{1}{h_n} \left| \int_{t_{n-1}}^{t_n} (t_n - t)\,u''(t)\,\mathrm{d}t \right| \leq \frac{1}{h_n} \max_{t_n \in I} \|u''(t_n)\| \cdot \left| \int_{t_{n-1}}^{t_n} t_n - t\,\mathrm{d}t \right|.$$

Nun gilt

$$\left| \int_{t_{n-1}}^{t_n} t_n - t\,\mathrm{d}t \right| \leq \frac{1}{2}h_n^2,$$

womit sich durch Einsetzen

$$\|\tau_n^h\| \leq \frac{1}{h_n} \max_{t_n \in I} \|u''(t_n)\| \cdot \frac{1}{2}h_n^2 = \frac{1}{2}h_n \max_{t_n \in I} \|u''(t_n)\| = \mathcal{O}(h_n)$$

ergibt. D. h. der Abschneidefehler konvergiert mit der Ordnung 1. Im Folgenden werden wir das hochgestelle h oftmals weglassen, sofern Missverständnisse ausgeschlossen sind. Nun soll man von diesem lokalen Diskretisierungsfehler auf den globalen schließen. Dafür verfolgt man den Ansatz

$$e_n = y_n - u_n = e_{n-1} + h_n\big[f(t_{n-1}, y_{n-1}) - f(t_{n-1}, u_{n-1})\big] - h_n\tau_n$$

mit

$$u_n \overset{(1)}{=} u_{n-1} + h_n\, f(t_{n-1}, u_{n-1}) + h_n\tau_n.$$

Hier nutze man nun die Lipschitz-Stetigkeit von $f(t_{n-1}, \cdot\,)$:

$$\begin{aligned}\|e_n\| &\leq \|e_{n-1}\| + h_n L\|y_{n-1} - u_{n-1}\| + h_n\|\tau_n\| = \|e_{n-1}\| + h_n L\|e_{n-1}\| + h_n\|\tau_n\| \\ &= \underbrace{(1 + h_n L)\|e_{n-1}\|}_{\text{Stabilität}} + \underbrace{h_n\|\tau_n\|}_{\text{Konsistenz}}. \end{aligned} \tag{2}$$

Der globale Fehler besteht aus zwei Anteilen, dem Abschneidefehler $h_n\|\tau_n\|$ und der Fehler-

[5] Dieses τ_n^h muss individuell für jedes Verfahren neu nachgewiesen werden.

fortpflanzung $(1 + h_n L)\|e_{n-1}\|$, welche aussagt, wie sich der alte Fehler e_{n-1} auf den aktuellen Fehler e_n auswirkt. Die Fehlerfortpflanzung ist Teil der numerischen Stabilität des Verfahrens. Für $N \to \infty$, also $h \to 0$, ist $\|e_n\| \to 0$ bzw. $\|e_{n-1}\| \to 0$ nicht garantiert. D. h. es wird etwas wie $|1 + h_n L| < 1$ gebraucht. Daran sieht man, dass für $|L| \gg 1$, $L \in \mathbb{R}$ [6] sehr kleine Schrittweiten h_n benötigt werden. Dies ist eine inhärente Schwäche von expliziten Verfahren wie bspw. dem Polygonzugverfahren.

Für den Moment wird die ausführliche Diskussion des ersten Terms dadurch vermieden, dass nun mit der diskreten Variante des Gronwallschen Lemmas gearbeitet wird. Zunächst wird die Fehlerdarstellung (2) rekursiv auf sich selbst angewendet:

$$\underbrace{\|e_n\|}_{w_n} \leq \|e_0\| + L \sum_{k=0}^{n-1} h_{k+1}\|e_k\| + \sum_{k=1}^{n} h_n\|\tau_k\| = \sum_{k=0}^{n-1} \underbrace{L h_{k+1}}_{a_k} \underbrace{\|e_k\|}_{w_k} + \underbrace{\sum_{k=1}^{n} h_k\|\tau_k\| + \|e_0\|}_{b_n} \tag{3}$$

Satz 1.5.1 Diskretes Gronwallsches Lemma

Es seien $(w_n)_n$, $(a_n)_n$ und $(b_n)_n$ Folgen nicht-negativer Zahlen, für welche $w_0 \leq b_0$ und

$$w_n \leq \sum_{k=0}^{n-1} a_k w_k + b_n$$

gelten. Wenn die Folge $(b_n)_n$ monoton steigt, dann gilt

$$w_n \leq \exp\Big(\sum_{k=0}^{n-1} a_k \Big) b_n.$$

Man wende nun Gronwall auf (3) an[7]:

$$\|e_n\| \leq \exp\big(L \underbrace{(t_n - t_0)}_{= \sum_{k=0}^{n-1} h_{k+1}}\big) \left[\|e_0\| + \sum_{k=1}^{n} h_k\|\tau_k\| \right]$$

Das bedeutet allerdings nichts anderes als

$$\begin{aligned} \max_{1 \leq n \leq N} \|e_n\| &\leq \exp(LT) \left[\|e_0\| + T \max_{1 \leq n \leq N} \|\tau_n\| \right] \\ &\leq \exp(LT) \left[\|e_0\| + T \max_{1 \leq n \leq N} \left\{ \frac{1}{2} h_n \|u''(t_n)\| \right\} \right]. \end{aligned} \tag{4}$$

[6] L kann später auch negativ sein und kennzeichnet nicht immer die Lipschitz-Stetigkeit.

[7] Das ist hier zulässig, weil alle Folgen nicht-negativ sind und (b_k) monoton steigt. Das sieht man daran, dass $b_{k+1} = b_k + h_k\|\tau_k\|$ gilt und der Summand $h_k\|\tau_k\| \geq 0$ ist.

Aus der vorherigen a priori Fehlerabschätzung kann man demnach interpretieren, dass die globale Konvergenzordnung mindestens gleich der lokalen Konsistenzordnung (Abschneidefehler) ist. Weiterhin wird $\exp(LT)$ für $L \gg 1$ bzw. $T \to \infty$ sehr groß, wodurch die Fehlerabschätzung bis auf eine grobe obere Schranke eher weniger nützlich ist.

Satz 1.5.2 Konvergenzresultat (Polygonzug)

Für die AWA $u'(t) = f(t,u)$ mit allen bisherigen Voraussetzungen (insbesondere $L > 0$) gelte $\|e_0\| \to 0$ und sei mit dem Eulerschen Polygonzugverfahren diskretisiert. Dann liefert (4) globale Konvergenz, sprich bei festem L und T gilt

$$\max_{1 \le n \le N} \|e_n\| \to 0,$$

mit der Konvergenzordnung $\|e_n\| = \mathcal{O}(h)$, wobei $h := \max_{1 \le n \le N} h_n$.

1.5.2. Allgemeine Einschrittmethoden / Runge-Kutta-Methoden

Wie gezeigt, hat die Euler-Methode für gewöhnliche Differentialgleichungen eine geringe Genauigkeit und hat deshalb in der Praxis kaum Bedeutung. In der Numerik partieller Differentialgleichungen wird das implizite Euler-Verfahren allerdings häufig genutzt, was für diese Veranstaltung aber nicht von Interesse ist. Wofür es hier also eher genutzt wird, ist exemplarisch für Beweise einer großen Klasse von Methoden, die wir hier jetzt durch Verallgemeinerungen konstruieren wollen. Ziel ist die systematische Konstruktion expliziter Einschrittformeln höherer Ordnung, insbesondere Taylorartige-Formeln und Runge-Kutta-Formeln. Ausgangspunkt ist hierfür die Taylorentwicklung im skalaren Fall ($d = 1$) für das Problem $u'(t) = f(t,u)$:

$$u(t) = \sum_{r=0}^{R} \frac{h^r}{r!} u^{(r)}(t-h) + \underbrace{\frac{h^{R+1}}{(R+1)!}\, u^{(R+1)}(\xi)}_{\text{Restglied}}, \quad \xi \in [\, t-h,\, h\,]$$

Wegen $u' = f$ gilt auch $u^{(r)}(t) = \left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^{r-1} f\big(t, u(t)\big)$, falls $f\big(t,u(t)\big)$ hinreichend regulär ist. Die R-stufigen Taylor-Verfahren lauten dann wie folgt:

$$y_n = y_{n-1} + \underbrace{h_n \sum_{r=1}^{R} \frac{h_n^{r-1}}{r!} \underbrace{f^{(r-1)}(t_{n-1}, y_{n-1})}_{\text{explizit!}}}_{=:F(h_n, t_{n-1}, y_{n-1})} \tag{5}$$

Dies sind immer noch Einschrittverfahren, die lediglich vom vorherigen Schritt (t_{n-1}, y_{n-1}) abhängen. Für $R = 1$ ergibt sich das Eulersche Polygonzugverfahren. Abstrakt erhält man somit

$$y_n = y_{n-1} + h_n \, F(h_n, t_{n-1}, y_{n-1}).$$

Die Struktur dieser Formeln ist somit identisch zum Polygonzugverfahren von vorher. Die Funktion $F(h_n, t_{n-1}, y_{n-1})$ wird Verfahrensfunktion genannt. Analog zu vorher definiere man den Abschneidefehler

$$\tau_n := (L_h u^h)_h = \frac{1}{h_n}(u_n - u_{n-1}) - F(h_n, t_{n-1}, u_{n-1}).$$

Definition 1.5.3 Konsistenz (Ordnung des Verfahrens durch die Abschneidefehlerbetrachtung)

Die allgemeine Einschrittmethode (5) heißt konsistent (mit der zugehörigen AWA) bzw. konsistent mit der Konsistenzordnung m, wenn

$$\max_{t_n \in I} \|\tau_n\| \to 0$$

bzw. wenn

$$\max_{t_n \in I} \|\tau_n\| = \mathcal{O}(h^m) \ (h \to 0).$$

Für die R-stufigen Taylor-Verfahren erhält man, aufgrund der Konstruktion, genau Konsistenzordnung $R = m$. In der Praxis kann die Konstruktion höherer Ordnung aufwändig sein, da die höheren Ableitungen von $f\big(t, u(t)\big)$ mittels Kettenregel bestimmt werden müssen.

Beispiele

(a) Für $R = 2$ erhält man

$$f'(t,u) \approx \frac{1}{h} \underbrace{\Big[f\big(t+h, \overbrace{u(t+h)}^{\text{Taylor 1. Ordnung}}\big) - f(t,u) \Big]}_{\text{Differenzquotient}}$$

$$\approx \frac{1}{h} \Big[f\big(t+h, \, u(t) + h\, f(t,u)\big) - f(t,u) \Big]. \tag{6}$$

Daraus resultiert durch Einsetzen von (6) in die Taylorformel (5) das folgende Verfahren:

$$y_n = y_{n-1} + h_n \, f(t_{n-1}, y_{n-1}) + \frac{1}{2} h_n \Big[f\big(t_{n-1}, \, y_{n-1} + h_n \, f(t_{n-1}, y_{n-1})\big) - f(t_{n-1}, y_{n-1}) \Big]$$

Wie zuvor beim expliziten Euler (1) kann hier die Konsistenzordnung $R = m = 2$ gezeigt

werden, d. h. $\|\tau_n^h\| = \mathcal{O}(h^2)$. Wenn diese Konstruktion für R verallgemeinert wird, dann erhält man die sogenannten expliziten Runge-Kutta-Verfahren. Diese haben die Form

$$F(h,t,u) = \sum_{r=1}^{R} c_r k_r(h,t,u) \tag{7}$$

mit

$$\begin{aligned} k_1(h,t,u) &= f(t,u) \\ k_r(h,t,u) &= f\Bigg(t + ha_r,\ u + h\sum_{s=1}^{r-1} b_{rs} k_s(h,t,u)\Bigg),\ r = 2,\dots,R \end{aligned}$$

und passend bestimmten Koeffizienten c_r, a_r, b_{rs}. Bei einer systematischen Bestimmung sollte man eine möglichst hohe Konsistenzordnung m erhalten. Eine Möglichkeit ist ein Koeffizientenvergleich mit den Taylor-Formeln (5):

$$\sum_{r=1}^{R} c_r k_r(h,t,u) = \sum_{r=1}^{m} \frac{h^{r-1}}{r!}\, f^{(r-1)}(t,u) + \mathcal{O}(h^m) \tag{8}$$

Konstruktionsgemäß gilt dann die Konsistenzordnung $m = R$.

(b) Wir setzen bei (8) für $R = 2$ an:

$$\begin{aligned} & c_1\, f(t,u) + c_2\, f\big(t + ha_2,\ u + hb_{21}\, f(t,u)\big) \\ &= (c_1 + c_2)\, f(t,u) + c_2 a_2 h\, f'(t,u) + c_2 b_{21} h\, f(t,u)\, f'_u(t,u) + \mathcal{O}(h^2) \\ &\overset{!}{=} f(t,u) + \frac{1}{2} h\Big[f'(t,u) + f'_u(t,u)\, f(t,u)\Big] + \mathcal{O}(h^2) \end{aligned}$$

Falls c_1, c_2, a_2, b_{21} beliebig gewählt werden, so erhält man im Allgemeinen lediglich Konsistenzordnung $m = 1$. Ab hier besteht also die Aufgabe darin, Koeffizienten zu finden, für die die maximale Konsistenzordnung tatsächlich erreicht wird. Der Koeffizientenvergleich, den wir hier ausführen, liefert auch keine eindeutigen Werte. Über den ersten Term auf beiden Seiten schließen wir auf $c_1 + c_2 = 1$ und über die zweiten Terme auf $c_2 a_2 = c_2 b_{21} = \frac{1}{2}$. Man kann also bspw. $c_1 = c_2 = \frac{1}{2}$ und $a_2 = b_{21} = 1$ wählen (Heunsches Verfahren mit der Ordnung $m = 2$). Aus dieser Wahl von Koeffizienten folgt über (7):

$$\begin{aligned} F(h,t,u) &= c_1 k_1(h,t,u) + c_2 k_2(h,t,u) \\ &= c_1\, f(t,u) + c_2\, f\big(t + ha_2,\ u + hb_{21}\, f(t,u)\big) \\ &= \frac{1}{2}\, f(t,u) + \frac{1}{2}\, f\big(t + h,\ u + h\, f(t,u)\big) \end{aligned}$$

Mittels (5) folgt dann das entsprechende explizite Einschrittverfahren:

$$\begin{aligned} y_n &= y_{n-1} + h_n\, F(h_n, t_{n-1}, y_{n-1}) \\ &= y_{n-1} + \frac{1}{2}\, h_n \Big[f(t_{n-1}, y_{n-1}) + f\big(t_n,\, y_{n-1} + h_n\, f(t_{n-1}, y_{n-1})\big) \Big] \end{aligned}$$

Über die Koeffizientenwahl $c_1 = 0$, $c_2 = 1$, $a_2 = b_{21} = \frac{1}{2}$ erhält man das modifizierte Euler-Verfahren, welches ebenfalls Konsistenzordnung $m = 2$ hat. Hierfür rechnet man analog wie zuvor:

$$\begin{aligned} F(h,t,u) &= c_1\, f(t,u) + c_2\, f\big(t + h a_2,\, u + h b_{21}\, f(t,u)\big) \\ &= f\big(t + \frac{h}{2},\, u + \frac{h}{2}\, f(t,u)\big) \\ \Rightarrow \quad y_n &= y_{n-1} + h_n\, F(h_n, t_{n-1}, y_{n-1}) \\ &= y_{n-1} + h_n\, f\Big(t_{n-\frac{1}{2}},\, y_{n-1} + \frac{h_n}{2}\, f(t_{n-1}, y_{n-1})\Big) \end{aligned}$$

(c) Für $R = 4$ gibt es insgesamt 13 freie Parameter c_r, a_r, b_{rs} und 11 Bestimmungsgleichungen! Eine Lösung mit Konsistenzordnung $m = 4$ ist das klassische und berühmte Runge-Kutta-Verfahren 4. Ordnung, oft als $RK(4)$ bezeichnet:

$$y_n = y_{n-1} + \frac{1}{6}\, h_n \big(k_1 + 2k_2 + 2k_3 + k_4\big)(h_n, t_{n-1}, y_{n-1})$$

$$\begin{aligned} k_1(h,t,y) &= f(t,y) & k_2(h,t,y) &= f\big(t + \tfrac{h}{2},\, y + \tfrac{h}{2}\, k_1(h,t,y)\big) \\ k_3(h,t,y) &= f\big(t + \tfrac{h}{2},\, y + \tfrac{h}{2}\, k_2(h,t,y)\big) & k_4(h,t,y) &= f\big(t + h,\, y + h\, k_3(h,t,y)\big) \end{aligned}$$

Die Koeffizienten eines Runge-Kutta-Verfahrens werden überlicherweise in einem Schema angeordnet, den Butcher-Tableaus, worüber man die Verfahrensvorschrift systematisch ablesen kann:

$$\begin{array}{c|ccccc} 0 & & & & & \\ a_2 & b_{21} & & & & \\ a_3 & b_{31} & b_{32} & & & \\ \vdots & \vdots & & \ddots & & \\ a_R & b_{R,1} & b_{R,2} & \cdots & b_{R,R-1} & \\ \hline & c_1 & c_2 & \cdots & c_{R-1} & c_R \end{array}$$

Beispiele

(a) Heunsches Verfahren 2. Ordnung:

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

modifiziertes Polygonzugverfahren:

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

(b) $RK(4)$ (Runge-Kutta-Verfahren 4. Ordnung):

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

(c) Ein weiteres Verfahren mit maximaler Konsistenzordnung 4 ist die optimale Formel von Kuntzmann, die wir in diesem Skriptum nur über das dazugehörige Butcher-Tableau darstellen:

0				
$\frac{2}{5}$	$\frac{2}{5}$			
$\frac{3}{5}$	$-\frac{3}{20}$	$\frac{3}{4}$		
1	$\frac{19}{44}$	$-\frac{15}{44}$	$\frac{40}{44}$	
	$\frac{55}{360}$	$\frac{125}{360}$	$\frac{125}{360}$	$\frac{55}{360}$

1.5.3. Lokale Konvergenz und Fehlerabschätzungen

Konzeptionell werden wir hier dasselbe Vorgehen wählen, wie wir es mit dem Polygonzug-Verfahren gemacht hatten. Über die Lipschitz-Stetigkeit – hier von der Verfahrensfunktion F – gelangen wir zu einer Fehlerbetrachtung $\|e_n\|$ und erhalten über Gronwall ein finales, wenn auch bloß ein lokales, Konvergenzresultat. An dieser Stelle sei nochmal an den Differenzenoperator und die davon ausgehende Definition des Abschneidefehlers von Seite 33 erinnert. Diese lassen sich auch bzgl. der Verfahrensfunktion F schreiben:

$$(L_h y^h)_n := \frac{1}{h_n}(y_n - y_{n-1}) - F(h_n, t_{n-1}, y_n, y_{n-1}) \tag{9}$$
$$\tau_n := (L_h u^h)_n := \frac{1}{h_n}(u_n - u_{n-1}) - F(h_n, t_{n-1}, u_n, u_{n-1})$$

Die Verfahrensfunktion hat hier ein Funktionsargument mehr. Dieses ist für implizite Verfahren relevant, welche neben dem vorhergenden Schritt y_{n-1} auch den aktuellen Schritt y_n miteinbeziehen. Entsprechend entfällt dieses Argument für explizite Verfahrensfunktionen, d. h. wir hätten die von Runge-Kutta bekannte Form $F(h_n, t_{n-1}, y_{n-1})$.

Definition 1.5.4 Lipschitz-Stetigkeit der Verfahrensfunktion

Eine Einschrittformel heißt Lipschitz-stetig, wenn ihre Verfahrensfunktion F der Lipschitz-Bedingung

$$\|F(h,t,y_n,y_{n-1}) - F(h,t,\tilde{y}_n,\tilde{y}_{n-1})\| \leq L\big(\,\|y_n - \tilde{y}_n\| + \|y_{n-1} - \tilde{y}_{n-1}\|\,\big)$$

für $(t,y_n),\ (t,y_{n-1}),\ (t,\tilde{y}_n),\ (t,\tilde{y}_{n-1}) \in I \times \mathbb{R}^d$ genügt.

Satz 1.5.5 Diskreter Stabilitätssatz

Eine Lipschitz-stetige Differenzenformel $(L_h y^h)_n$ gemäß (9) ist diskret stabil, falls für beliebige Gitterfunktionen

$$y^h = \{y_n\}_{n\geq 0} \quad \text{und} \quad z^h = \{z_n\}_{n\geq 0}$$

mit hinreichend kleiner Schrittweise $h < \frac{1}{2L}$ die Abschätzung

$$\|y_n - z_n\| \leq \exp\big(\gamma L(t_n - t_0)\big) \cdot \left[\, \|y_0 - z_0\| + \sum_{k=1}^{n} h_k \big\|(L_h y^h - L_h z^h)_k\big\| \,\right]$$

gilt. Bei expliziten Verfahren ist $\gamma = 1$, bei impliziten Methoden ist $\gamma = 4$. Für explizite Verfahren entfällt die Schrittweitenbedingung von $h < \frac{1}{2L}$.

Beweis zu Satz 1.5.5

Für zwei Gitterfunktionen $L_h y^h$ und $L_h z^h$ erhält man die Fehlerabschätzung

$$y_n - z_n = y_{n-1} - z_{n-1} + h_n\big[F(h,t,y_n,y_{n-1}) - F(h,t,z_n,z_{n-1}) - (L_h y^h - L_h z^h)_n\big]. \quad (10)$$

Man setze $e_n := y_n - z_n$ und $\varepsilon_n := L_h y^h - L_h z^h$.

(i) Expliziter Fall: Hier entfällt das implizite Funktionsargument y_n bzw. z_n, d. h. man rechnet mit $F(h,t,y_{n-1})$, $F(h,t,z_{n-1})$. Aus (10) erhält man durch die Lipschitz-Stetigkeit dann

$$\|e_n\| \leq \|e_{n-1}\| + h_n L\|e_{n-1}\| + h_n\|\varepsilon_n\|,$$

weiter mit Hilfe von Rekursion

$$\leq \sum_{k=0}^{n-1} L h_{k+1} \|e_k\| + \sum_{k=1}^{n} h_k \|\varepsilon_k\| + \|e_0\|$$

und zuletzt aus dem Gronwall-Lemma (1.5.1)

$$\leq \exp\Big(L \sum_{k=0}^{n-1} h_{k+1} \Big) \cdot \Big[\|e_0\| + \sum_{k=1}^{n} h_k \|\varepsilon_k\| \Big] = \exp\big(L(t_n - t_0)\big) \cdot \Big[\|e_0\| + \sum_{k=1}^{n} h_k \|\varepsilon_k\| \Big].$$

Damit ist alles für den expliziten Fall gezeigt.

(ii) Impliziter Fall: Hier gibt es neben dem reinen impliziten Fall, bei dem y_{n-1} bzw. z_{n-1} entfällt (siehe implizites Euler-Verfahren) auch den gemischt expliziten/impliziten Fall (siehe Trapezregel). Von daher kann hier nichts fallen gelassen werden, aber analog berechnet man dennoch

$$\|e_n\| \leq \|e_{n-1}\| + h_n L\big(\|e_n\| + \|e_{n-1}\| \big) + h_n \|\varepsilon_n\|,$$

wobei in $h_n L\big[\|e_n\| + \|e_{n-1}\| \big]$ das $\|e_{n-1}\|$ direkt aus der Lipschitz-Stetigkeit der Verfahrensfunktion folgt und $\|e_n\|$ der implizite Anteil ist. Die Schwierigkeit ist, dass $\|e_n\|$ auf beiden Seiten der Ungleichung auftritt. Äquivalent umgeformt bringt man den $\|e_n\|$ Anteil komplett auf die linke Seite, erhält also

$$(1 - h_n L)\|e_n\| \leq (1 + h_n L)\|e_{n-1}\| + h_n \|\varepsilon_n\|.$$

Nun kann beidseitig mit $(1 - h_n L)^{-1}$ multipliziert werden. Das ist zulässig und ändert die Ungleichung nicht, denn aufgrund der Schrittweitenbedingung $h_n < \frac{1}{2L}$ ist $1 - h_n L > 0$ gewährleistet. Man erhält somit

$$\|e_n\| \leq \frac{1 + h_n L}{1 - h_n L} \|e_{n-1}\| + \frac{h_n}{1 - h_n L} \|\varepsilon_n\|.$$

Wie beim expliziten Fall folgt nun aus Rekursion und dem Gronwall-Lemma

$$\|e_n\| \leq \exp\big(4L(t_n - t_0)\big) \cdot \Big[\|e_0\| + \sum_{k=1}^{n} h_k \|\varepsilon_k\| \Big].$$

Damit ist auch alles für den impliziten Fall gezeigt. □

Satz 1.5.6 Lokale Konvergenz allgemeiner Einschrittmethoden

Die Differenzenformel $(L_h y^h)$ sei Lipschitz-stetig und konsistent mit der AWA. Für $\|y_0 - u_0\| \to 0$ konvergiert dann

$$\max_{t\in I} \|y_n - u_n\| \to 0 \quad (h \to 0), \quad h := \max_{1\le n\le N} h_n$$

und für hinreichend kleine h_n (im impliziten Fall $h_n < \frac{1}{2L}$) gilt die a priori-Fehlerabschätzung

$$\|y_n - u_n\| \le \exp\big(\gamma L(t_n - t_0)\big) \cdot \left[\|y_0 - u_0\| + \sum_{k=1}^{n} h_k \|\tau_k\|\right],\ 1 \le n \le N. \tag{11}$$

Beweis zu Satz 1.5.6
Man nutze den diskreten Stabilitätssatz und setze $y_n = y_n$ (diskrete Lösung) bzw. $z_n = u_n$ (exakte Lösung). Daraus erhält man dann $L_n y^h \equiv 0$ und $L_n u^h = \tau_n$. □

1.5.4. Globale Konvergenz

Aus den lokalen Abschätzungen kann man nun globale Resultate herleiten. Die Hauptschwierigkeit besteht darin, dass die lokale Abschätzung (11) nur auf verhältnismäßig kleinen Intervallen $I := [\,t_0,\, t_0 + T\,]$ mit kleinem T Sinn ergibt. Dies liegt am Term $\exp\big(\gamma L(t_n - t_0)\big)$. D. h. für Aussagen mit $T \to \infty$ muss man T in $\exp\big(\gamma LT\big)$ kontrollieren bzw. eliminieren. Neben der Lipschitz-Stetigkeit fordere man also eine weitere Strukturbedingung, nämlich die Monotoniebedingung, bei der

$$-\big\langle f(t,u) - f(t,\tilde u), u - \tilde u\big\rangle \ge \lambda(t)\,\|u - \tilde u\|^2$$

für alle $(t,u),(t,\tilde u) \in I \times \mathbb{R}^d$ mit stetigem $\lambda \ge 0$ erfüllt sein muss. So erreicht man in etwa für das implizite Euler-Verfahren eine globale Konvergenzaussage:

Satz 1.5.7 Globale Konvergenz des impliziten Euler-Verfahrens

Die AWA sei L-stetig und monoton. Dann sind die Lösungen des impliziten Euler-Verfahrens

$$y_n = y_{n-1} + h_n\, f(t_n, y_n), \quad n \ge 1, \quad y_0 = u_0,$$

für beliebige Schrittweiten h_n wohldefiniert und es gilt die globale Fehlerabschätzung

$$\|y_n - u_n\| \le \frac{1}{2}\,\min\big\{t_n - t_0,\, \lambda^{-1}\big\} \cdot \max_{1\le k\le n} \big\{h_k \max_{I_k} \|u''\|\big\}, \quad t_n \ge t_0.$$

Beweis zu Satz 1.5.7

(i) In jedem Schritt des impliziten Euler-Verfahrens ist ein Gleichungssystem

$$y_n - h_n\, f(t_n, y_n) = y_{n-1}$$

zu lösen. Aus der L-Stetigkeit und Monotonie ist die Abbildung $g(x) := x - h_n\, f(t_n, x)$ ebenfalls L-stetig und strikt monoton im Sinne

$$\big\langle g(x) - g(y),\, x - y \big\rangle \geq \gamma\, \|x - y\|^2, \quad x, y \in \mathbb{R}^d,$$

mit einer festen Konstante $\gamma > 0$. Vorerst nutzen wir $g(x) = c$ für ein beliebiges $c \in \mathbb{R}^d$ und betrachten die zur Gleichung äquivalente Fixpunktgleichung

$$G(x) := x - \theta\, \big(g(x) - c\big) = x$$

mit $\theta \in \big(0, \frac{2\gamma}{L^2}\big)$, wobei L die Lipschitz-Konstante von g sei. Für beliebige $x, y \in \mathbb{R}^d$ ergibt sich

$$\begin{aligned}
\|G(x) - G(y)\|^2 &= \|x - \theta\, g(x) - y + \theta\, g(y)\|^2 \\
&= \|x - y\|^2 - 2\theta \big\langle x - y,\, g(x) - g(y) \big\rangle + \theta^2 \|g(x) - g(y)\|^2 \\
&\leq (1 - 2\gamma\theta + L^2\theta^2)\, \|x - y\|^2 \\
&= \big(1 + \theta \underbrace{(L^2\theta - 2\gamma)}_{< \frac{2L^2\gamma}{L^2} - 2\gamma = 0}\big)\, \|x - y\|^2 < \|x - y\|^2.
\end{aligned}$$

Damit ist G eine Kontraktion, nimmt nach dem Banachschen Fixpunktsatz also einen eindeutig bestimmten Fixpunkt x^* an, der nach Konstruktion eine Lösung der Aufgabe $g(x^*) = c$ ist. Demnach besitzt $g(x) = x - h_n\, f(t_n, x)$ also ebenfalls eine eindeutig bestimmte Lösung, unabhängig von der Wahl von h_n, womit die Lösungen des impliziten Euler-Verfahrens schonmal wohldefiniert sind.

(ii) Für den Fehler $e_n = y_n - u_n$ gilt wieder die Differenzengleichung

$$e_n = e_{n-1} + h_n \big[f(t_n, y_n) - f(t_n, u_n)\big] - h_n \tau_n$$

mit dem Abschneidefehler τ_n, das für das implizite Euler-Verfahren analog wie beim Polygonzugverfahren auf Seite 34 durch

$$\|\tau_n\| \leq \frac{1}{2}\, h_n \max_{I_n} \|u''\| \tag{12}$$

abgeschätzt werden kann. Wir multiplizieren auf beiden Seiten der Differenzengleichung mit $\frac{e_n}{\|e_n\|}$

und nutzen die Monotoniebedingung, um

$$\|e_n\| \leq \|e_n\|^{-1}\langle e_{n-1}, e_n\rangle - \lambda_n h_n \|e_n\| + h_n \|e_n\|^{-1}\langle \tau_n, e_n\rangle$$

$$\Rightarrow \quad (1+\lambda_n h_n)\|e_n\| \leq \|e_{n-1}\| + h_n\|\tau_n\|$$

$$\Rightarrow \qquad \|e_n\| \leq \frac{1}{1+\lambda_n h_n}\|e_{n-1}\| + \frac{1}{1+\lambda_n h_n}\|\tau_n\| \tag{13}$$

zu erhalten.

(iii) Für schwache Monotonie ($\lambda \geq 0$) summiert man (13) über $k = 1, \ldots, n$ und über $e_0 = 0$ erhält man die T-abhängige Abschätzung

$$\|e_n\| \leq \sum_{k=1}^{n} h_k \|\tau_k\| \leq \left(\sum_{k=1}^{n} h_k\right) \cdot \max_{1\leq k\leq n} \|\tau_k\| \leq \frac{1}{2}\,(t_n - t_0) \max_{1\leq k\leq n} \left\{h_k \max_{I_k} \|u''\|\right\}.$$

Bei striker Monotonie ($\lambda > 0$) erschließt man per Induktion aus (13) die Abschätzung

$$\|e_n\| \leq \lambda^{-1} \max_{1\leq k\leq n} \|\tau_k\| \overset{(12)}{=} \frac{1}{2}\,\lambda^{-1} \cdot \max_{1\leq k\leq n} \left\{h_k \max_{I_k} \|u''\|\right\}$$

Für $n = 1$ gilt trivialerweise

$$\|e_1\| \leq \frac{h_1}{1+\lambda h_1}\,\|\tau_1\| \leq \lambda^{-1}\,\|\tau_1\|.$$

Gelte die Behauptung für $n-1$, dann folgt aus (13):

$$\begin{aligned}\|e_n\| &\leq \frac{1}{1+\lambda h_n}\,\|e_{n-1}\| + \frac{1}{1+\lambda h_n}\,\|\tau_n\| \\ &\leq \frac{1}{1+\lambda h_n}\,\lambda^{-1} \max_{1\leq k\leq n-1} \|\tau_k\| + \frac{h_n}{1+\lambda h_n}\,\|\tau_n\| \leq \lambda^{-1} \max_{1\leq k\leq n} \|\tau_k\|\end{aligned}$$

Dies vervollständigt den Beweis. □

Die Argumentation aus diesem Beweis lässt sich auf allgemeine Einschrittverfahren übertragen, für die die Verfahrensfunktion L-stetig und monoton ist. Leider ist das bei der Approximation monotoner AWAs mit Verfahren höherer Ordnung in der Regel nicht der Fall, weshalb ein anderer Zugang zur globalen Fehlerabschätzung allgemeiner Einschrittverfahren gefunden werden muss. Ausgangspunkt dazu ist die Beobachtung, dass monotone L-stetige AWAs exponentiell stabile Lösungen haben. Diese Überlegung führt tatsächlich zu einer globalen Fehlerabschätzung für allgemeine Einschrittmethoden, die wir ohne Beweis in Form eines Satzes angeben wollen. Der Beweis ist für interessierte Leser bspw. in (Rannacher 2017, S. 56–57) zu finden.

Satz 1.5.8 Globale Konvergenz allgemeiner Einschrittmethoden

Es gelten die Voraussetzungen aus Satz 1.5.6. Dann gilt

$$\max_{t_n \in I} \|y_n - u_n\| \leq K \cdot \max_{t_n \in I} \|\overline{\tau}_n\|$$

mit $\overline{\tau}_k$ als Maximum des Abschneidefehlers aller möglichen Lösungen der AWA und einer Konstanten K, die von T unabhängig ist.

1.5.5. Rundungsfehlereinfluss und adaptive Schrittweitenkontrolle

Rundungsfehler

In der Theorie geht $h \to 0$, wird also sehr klein. In der Praxis können beliebig kleine h nicht realisiert werden, da nur endliche Zahlendarstellungen möglich sind. Es gibt also eine Interaktion zwischen Maschinengenauigkeit eps und h. Ziel ist somit eine sinnvolle Wahl von h bzw. h_n. Selbst

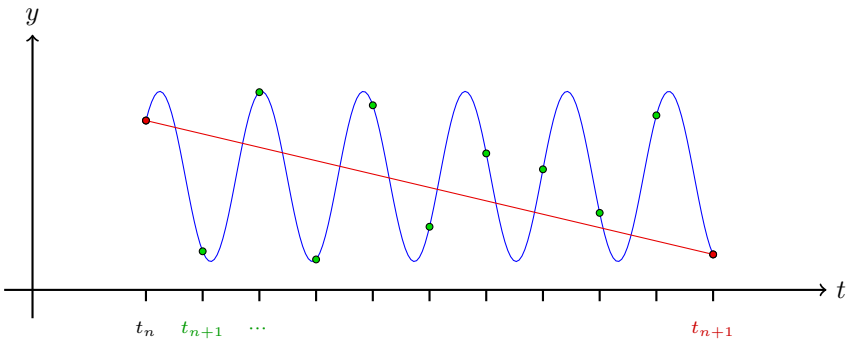


Abbildung 1.5.: Abhängig von der zu lösenden Aufgabe ist eine kleinere Schrittweite wichtig.

wenn eine gewählte Schrittweite etwa die physikalischen Bedingungen erfüllt und den Verlauf einer Lösung im Wesentlichen darstellt, so muss trotzdem die Numerik gut genug sein. Idealerweise basiert die Numerik auf bestmöglicher Approximation der zugrundeliegenden Physik, daher ist eine numerische Schrittweite nur dann sinnvoll, wenn sie mindestens so klein ist, wie die physikalische. Damit wurde aber noch nicht das Problem der Rundungsfehler addressiert. Wie muss die numerische Schrittweite denn in Bezug auf eps gewählt werden? Um das zu beantworten, sei $\tilde{y}_n$ eine gestörte diskrete Lösung, z. B. durch Rundungsfehler. Dann gilt

$$\tilde{y}_n = \tilde{y}_{n-1} + h_n \, F(h, t_{n-1}, \tilde{y}_{n-1}) + \varepsilon_n$$

mit $\|\varepsilon\| \sim \text{eps} \, \|y_{n-1}\|$. Setze $\tilde{e}_n = \tilde{y}_n - u_n$. Dann gilt

$$\tilde{e}_n = \tilde{e}_{n-1} + h_n \big[F(h, t_{n-1}, \tilde{y}_{n-1}) - F(h, t_{n-1}, u_{n-1})\big] - h_n \tau_n + \varepsilon_n$$

und somit folgt aus der Lipschitz-Stetigkeit von F

$$\begin{aligned}\|\tilde{e}_n\| &\leq \|\tilde{e}_{n-1}\| + h_n L\|\tilde{e}_{n-1}\| + h_n\|\tau_n\| + \|\varepsilon_n\| \\ &\leq \|\tilde{e}_0\| + L\sum_{k=0}^{n-1} h_{k+1}\|\tilde{e}_k\| + \sum_{k=1}^{n} h_k\|\tau_k\| + \sum_{k=1}^{n}\|\varepsilon_k\|\end{aligned}$$

und durch das Gronwall-Lemma erhält man

$$\begin{aligned}&\leq \exp\big(L(t_n - t_0)\big)\left[\|\tilde{e}_0\| + \sum_{k=1}^{n} h_k\|\tau_k\| + \sum_{k=1}^{n}\|\varepsilon_k\|\right] \\ &\leq \exp\big(L(t_n - t_0)\big)\left[\|\tilde{e}_0\| + \max_{1\leq n\leq N}\|\tau_n\|\sum_{k=1}^{n} h_k + \sum_{k=1}^{n} h_k h_k^{-1}\|\varepsilon_k\|\right] \\ &\leq \exp\big(L(t_n - t_0)\big)\left[\|\tilde{e}_0\| + (t_n - t_0)\left[\max_{1\leq n\leq N}\|\tau_n\| + \max_{1\leq n\leq N}\|\varepsilon_n\|h_n^{-1}\right]\right] \\ &= \underbrace{\exp\big(L(t_n - t_0)\big)}_{\text{Math. Aufgabenstellung}}\left[\|\tilde{e}_0\| + (t_n - t_0)\left[\underbrace{\max_{1\leq n\leq N}\|\tau_n\|}_{\text{Abschneidefehler}} + \underbrace{\max_{1\leq n\leq N}\operatorname{eps}\|y_{n-1}\|h_n^{-1}}_{\text{Neu: Diskretisierungsfehler}}\right]\right]. \qquad (14)\end{aligned}$$

Es gibt somit eine Relation $\operatorname{eps} \sim h_n^{-1}$. D. h. h_n darf nicht zu klein sein, um zu vermeiden, dass der Diskretisierungsfehler den Fehlerabschätzer dominiert. Für bspw. $\operatorname{eps} = 10^{-16}$ sind empirisch gesehen h_n bis 10^{-12} respektive dem Wert von $\max_{1\leq n\leq N}\|y_{n-1}\|$ noch möglich. Grundsätzlich sollte h_n mehrere Größenordnungen größer als eps sein.

Adaptive Schrittweitenkontrolle

Bisher lag immer eine äquidistante Verteilung der t_n vor, also waren die h_n uniform. Nun werden aber nicht-uniforme h_n betrachtet. Hier hat man nun zwei Möglichkeiten. Einmal die globale

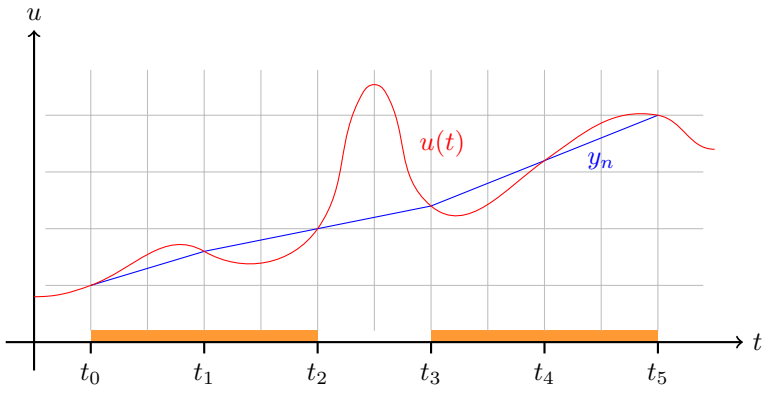


Abbildung 1.6.: Problem bei der Annäherung durch einen Polygonzug.

Verfeinerung, bei der ein feines Gitter über den gesamten Bereich gelegt wird. Das ist im orangenen Bereich aus Abbildung 1.6 jedoch nicht wirklich nötig und aufgrund der vielen Zeitschritte auch ressourcenintensiv. Die andere Möglichkeit ist die adaptive Wahl der h_n. Diese basiert auf Fehlerabschätzungen und ist a posteriori, passiert also während der Rechnung.

Ausgangspunkt ist wiederum die a priori-Fehlerabschätzung (14), die es erlaubt, die entscheidenden Terme zu identifizieren. In gekürzter Form betrachtet man also

$$\|\tilde{e}_n\| \leq \underset{\sim \exp(LT)}{KT} \|\tau_n\| + \varepsilon_n$$

mit der Annahme[8], dass $\|e_0\| \equiv 0$ (fehlerfreie Startwerte). Allerdings ist $\|\tau_n\|$ unbekannt und unter Umständen kann K sehr groß sein. Der Abschneidefehler ist letztendlich die einzige Größe, die tatsächlich direkt vom numerischen Verfahren abhängt und währenddessen kontrolliert werden kann, denn K und T sind bereits durch die mathematisch kontinuierliche Aufgabe festgelegt. Der Abschneidefehler kann durch

$$\tau_n = \tau^{(m)}(t_n)h^m + \mathcal{O}(h^{m+1}), \quad \tau^{(m)}(t_n) = \frac{1}{(m+1)!}\, u^{(m+1)}(t_{n-1}) \tag{15}$$

dargestellt werden. Hierbei steht $\tau^{(m)}(t_n)$ im Einklang mit der Definition der Konsistenz. Man nennt diesen Teil die Hauptabschneidefunktion. Für das globale Ziel $\|\tilde{e}_n\| \leq \mathrm{TOL}$ muss schlussendlich τ_n geeignet abgeschätzt werden. Weil die Hauptabschneidefunktion von der Aufgabenstellung festgelegt wird, ist lediglich h^m variabel.

Strategien zur Schrittweitensteuerung

Ausgangspunkt ist die a priori-Fehlerabschätzung, aus der man

$$Kh_n^m \|\tau_n^{(m)}\| \approx \frac{\mathrm{TOL}}{T} \quad \Leftrightarrow \quad h_n \approx \sqrt[m]{\frac{\mathrm{TOL}}{KT\,\|\tau_n^{(m)}\|}} \tag{16}$$

erhält. Diese Wahl erfüllt in der Tat

$$\begin{aligned}\max_n \|e_n\| &\leq K \sum_n h_n \|\tau_n\| = K \sum_n h_n h_n^m \|\tau^{(m)}\| \overset{(16)}{=} K \sum_n h_n \frac{\mathrm{TOL}}{KT} \frac{\|\tau_n^{(m)}\|}{\|\tau_n^{(m)}\|} \\ &= \sum_n h_n \frac{\mathrm{TOL}}{T} = \frac{T}{T}\,\mathrm{TOL} = \mathrm{TOL}\end{aligned}$$

Das Problem ist jedoch, dass $\tau_n^{(m)}$ in (16) nach wie vor unbekannt ist. Man führe darum eine Schätzschrittweite H ein, mit deren Hilfe zwei Proberechnungen durchgeführt werden.

[8] Dies ist eine sehr starke Annahme, die in der Praxis häufig nicht erfüllt werden kann. Aber für geeignete Klassen von AWAs, nämlich den monotonen, ist es möglich.

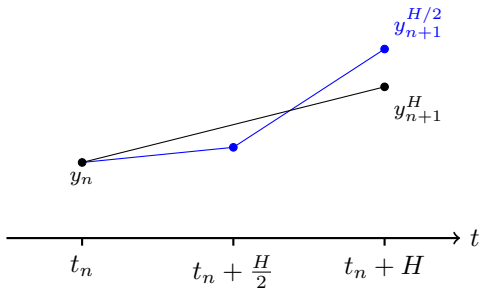


Abbildung 1.7.: Proberechnungen mit einer Schätzschrittweite H.

Man vergleiche davon die Fehler:

$$\begin{aligned} y_{n+1}^{H} - u_{n+1} &= e_n + H\big[F(H,t,y_n) - F(H,t,u_n)\big] - H\tau_{n+1}^{H} \\ &\overset{(15)}{=} \big(1 + \mathcal{O}(H)\big)e_n - H^{m+1}\,\tau_{n+1}^{(m)} + \mathcal{O}(H^{m+2}) \end{aligned}$$

Analog für

$$y_{n+1}^{H/2} - u_{n+1} = \big(1 + \mathcal{O}(H)\big)e_n - 2\big(\tfrac{1}{2}H\big)^{m+1}\,\tau_{n+1}^{(m)} + \mathcal{O}(H^{m+2}).$$

Man kann daraus nun den Fehler zu beiden Schätzungen errechnen:

$$y_{n+1}^{H/2} - y_{n+1}^{H} = \mathcal{O}(H)e_n - \tau_{n+1}^{(m)}\left[\,2\big(\tfrac{1}{2}H\big)^{m+1} - H^{m+1}\,\right] + \mathcal{O}(H^{m+2}).$$

Umstellen nach $\tau_{n+1}^{(m)}$ liefert

$$\tau_{n+1}^{(m)} = \frac{y_{n+1}^{H/2} - y_{n+1}^{H}}{H^{m+1}(1 - 2^{-m})} + \mathcal{O}(H) + \mathcal{O}(H^{-m})e_n.$$

Für $\mathcal{O}(H)$ kann $\mathcal{O}(H) \sim 0$ angenommen werden. Für den Summanden $\mathcal{O}(H^{-m})e_n$ gibt es zwei Möglichkeiten. Entweder ist $e_n \equiv 0$, oder aber $e_n = \mathcal{O}(H^{m+1})$. Im zweiten Fall ist für $H \to 0$ $\mathcal{O}(H^{-m})\mathcal{O}(H^{m+1}) = \mathcal{O}(H) = 0$. D. h. man erhält die finale Abschätzung

$$\bar{\tau}_{n+1}^{(m)} = \frac{y_{n+1}^{H/2} - y_{n+1}^{H}}{H^{m+1}(1 - 2^{-m})}$$

des Abschneidefehlers. Aus (16) kann man nun h_{n+1} wie folgt bestimmen:

$$h_{n+1} = \left(\frac{\mathrm{TOL}}{KT\,\|\bar{\tau}_{n+1}^{(m)}\|}\right)^{\frac{1}{m}}$$

Über die Wahl von TOL $\gg$ eps kann nun h_{n+1} adaptiv gesteuert werden. In jedem Schritt wird dafür $\bar{\tau}_{n+1}^{(m)}$ neu angepasst.

Algorithmus (Adaptive Schrittweitensteuerung) Es sei y_n als Approximation zu $u(t_n)$ mit der Schrittweite h_n berechnet.

Algorithmus 5.

1. *Setze:*

   $H = 2h_n$ *(Probe-Schrittweite)*

   $t_{n+1} = t_n + H,\ t_{n+\frac{1}{2}} = t_n + \frac{H}{2}$

2. *Berechne:*

   y_{n+1}^{H} *(1. Rechnung; 1 Zeit-Schritt)*

   $y_{n+1}^{H/2}$ *(2. Rechnung; 2 Zeit-Schritte)*

3. *Berechne:*

   $\bar{\tau}_{n+1}^{(m)}$

4. *Berechne:*

   $$h_{n+1} = \left(\frac{\mathrm{TOL}}{KT \, \|\bar{\tau}_{n+1}^{(m)}\|} \right)^{\frac{1}{m}}$$

5. *Prüfe:*

   $h_{n+1} \ll \frac{1}{2} H = h_n$

   Falls ja:

   Gehe zu Schritt 1 *mit* $H = 2h_{n+1}$

   Falls nein:

   Setze:

   $H := h_{n+1}$

   $t_{n+1} := t_n + \frac{H}{2}$

   $y_{n+1} := y_{n+1}^{H/2}$

Bemerkung: Setze in Schritt 5 ggf. minimale Schrittweite $h_{\min}$ und prüfe $H < h_{\min}$. Falls ja, beende Rechnung. Aufgrund der doppelten Berechnungen y_{n+1}^{H} und $y_{n+1}^{H/2}$ und einer möglichen Wiederholung in Schritt 5, falls die Schrittweite noch nicht gut genug war, ist der Algorithmus selbstverständlich teuer. Es ist aber im Allgemeinen so, dass numerische Quantifizierungen von Fehlern entsprechende numerische Kosten haben.

1.6. Numerische Stabilität

Zur Motivation sei an (2) auf Seite 34 erinnert:

$$\|e_n\| \leq \underbrace{(1 + h_n L)\|e_{n-1}\|}_{\text{Stabilität}} + \underbrace{h_n \|\tau_n\|}_{\text{Konsistenz}}$$

$$e_n \approx (1 + h_n L)e_{n-1} - h_n \tau_n$$

Bisher wurde der Konsistenzteil thematisiert, jetzt rückt die Stabilität in den Fokus. Grob gesagt wird hierbei untersucht, wie h_n und L miteinander zusammenhängen und welche Konsequenzen das für die Wahl des numerischen Verfahrens hat. Betrachte das Modellproblem

$$\begin{aligned} u'(t) &= f(t, u), \qquad \text{in } I = [\, t_0,\, t_0 + T\,] \\ u(t_0) &= u_0 \end{aligned}$$

mit $f(t, u) := \lambda\, u(t)$, $\lambda \in \mathbb{R}$. Hier muss zwischen drei Fällen unterschieden werden:

- $\lambda > 0$: exponentielles Wachstum
- $\lambda = 0$: $u(t) \equiv u_0$
- $\lambda < 0$: exponentieller Abfall

Für das Modellproblem gilt sofort $L = |\lambda|$. Für $\lambda < 0$ gilt als erstes Beispiel für die numerische Approximation mit dem Polygonzugverfahren

$$\frac{y_n - y_{n-1}}{h_n} = \lambda y_{n-1} \quad \Leftrightarrow \quad y_n = (1 + h_n\lambda)y_{n-1} = (1 + h_n\lambda)^2 y_{n-2} = \cdots \tag{1}$$

Somit gilt für $\lambda < 0$ (hoffentlich) $y_n \leq y_{n-1} \leq \ldots \leq y_0$.

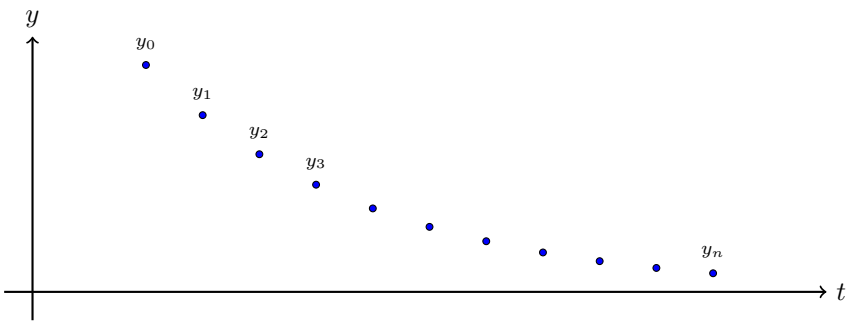


Abbildung 1.8.: Exponentieller Abfall.

Betrachte (1) genauer:

$$y_n \leq y_{n-1} \quad \Rightarrow \quad |1 + h_n\lambda| \leq 1$$

Hier gibt es wiederum zwei Fälle:

(i) $1 + h_n\lambda \leq 1$: Dann ist $h_n\lambda \leq 0$, was aufgrund von $h_n > 0$ and $\lambda < 0$ stets erfüllt ist.

(ii) $-(1 + h_n\lambda) \leq 1$: Dann ist $-h_n\lambda \leq 2$, also $h_n \leq \frac{2}{-\lambda}$.

In der Praxis ist es leider so, dass $\lambda \ll -1$ gilt, sprich dass $|\lambda|$ sehr groß ist. Damit $|1 + h_n\lambda| \leq 1$ erfüllt werden kann, muss entsprechend h_n sehr klein gewählt werden. Dementsprechend kommt die Frage auf, ob alle numerischen Verfahren derartige Bindungen haben, oder ob man sich davon lösen kann, immerhin bedeuten kleine h_n großen Rechenaufwand (und möglicherweise Rundungsfehler).

Beispiel Sei $u'(t) = \lambda\, u(t)$ für $\lambda = -1$ und mit $u(0) = 1$ als Anfangswert. Dann liefert das Polygonzugverfahren

$$y_n = (1 - h_n)y_{n-1}$$

Entsprechend ist nach obigen Berechnungen die Bedingung $|1 + h_n\lambda| \leq 1$ erfüllt, wenn $h_n \leq 2$. Für $h_n = \frac{3}{2}$ erhält man die ersten Iterierten

$$y_1 = -\frac{1}{2}, \quad y_2 = \frac{1}{4}, \quad y_3 = -\frac{1}{8}, \quad y_4 = \frac{1}{16}, \quad \dots$$

Für $h_n \ll \frac{1}{2}$ wären zudem alle Werte y_n positiv gewesen. D. h. das Verfahren wird umso stabiler,

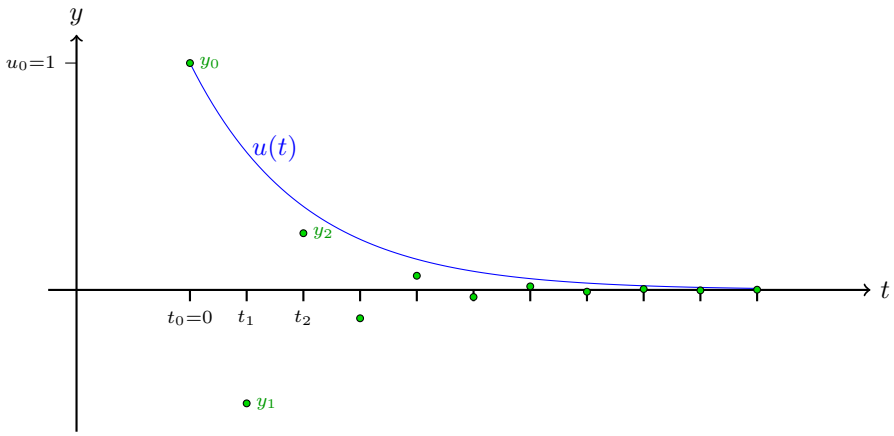


Abbildung 1.9.: Für $h_n = \frac{3}{2}$ erhält man eine stabile Lösung.

je kleiner die gewählte Schrittweite h_n ist.

Diese Analyse kann auch auf Systeme von Differentialgleichungen erweitert werden, zum Beispiel Räuber-Beute. Hierzu betrachtet man die sogenannte Jacobi-Matrix

$$f'_x(t,x),\ x=(x_1,\dots,x_d)$$

Bemerkung: Im Allgemeinen wird die folgende Stabilitätsanalyse für $\lambda \in \mathbb{C}$ geführt. Bereits für vermeintlich einfache Systeme (z. B. Räuber-Beute) können die führenden Koeffizienten (sogenannte Eigenwerte) komplexwertig sein. Für die Stabilitätsanalyse ist man also insbesondere am Verhalten der Eigenwerte interessiert.

1.6.1. Exkurs: Eigenwerte

In Vorgriff auf Kapitel 2 führen wir hier bereits Eigenwerte ein. Gegeben sei $A \in \mathbb{C}^{n\times n}$ mit einem Eigenvektor $x \neq 0$, sodass die Eigenwertgleichung

$$Ax = \lambda x$$

erfüllt ist. Hier ist $\lambda \in \mathbb{C}$ der sogenannte Eigenwert. Bekannte Resultate aus der linearen Algebra sind:

(1) Eigenvektoren sind nicht eindeutig bestimmt, denn z. B. ist für $\alpha \neq 0$ auch αx ein Eigenvektor.

(2) Rayleigh-Quotient:

$$Ax = \lambda x \Leftrightarrow x^T Ax = x^T \lambda x \Leftrightarrow \langle Ax, x\rangle = \lambda \|x\|^2 \Leftrightarrow \lambda = \frac{\langle Ax, x\rangle}{\|x\|^2}$$

(3) Die Eigenwerte λ sind Nullstellen des charakteristischen Polynoms. D. h.

$$p_A(\lambda) = \det(A - \lambda I) = 0.$$

D. h. zu $A \in \mathbb{C}^{n\times n}$ hat man n Eigenwerte (möglicherweise mit Vielfachheiten). Die Bestimmung von $p_A(\lambda)$ und den dazugehörigen Eigenwerten kann unter Umständen sehr aufwändig sein, wofür die numerische Approximation wiederum notwendig wird (siehe Kapitel 2 und Kapitel 3).

(4) Falls $A \in \mathbb{R}^{n\times n}$ (reell), dann können die Eigenwerte λ dennoch komplexwertig sein, aber dann sind sie komplex-konjugiert.

(5) Falls $A \in \mathbb{R}^{n\times n}$, $A = A^T$, dann sind alle Eigenwerte reell.

(6) Eine Matrix $A \in \mathbb{C}^{n\times n}$ ist diagonalisierbar, falls eine reguläre Matrix $U \in \mathbb{C}^{n\times n}$ existiert,

sodass

$$U^{-1}AU = D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$$

gilt. Hierbei wird U aus Eigenvektoren zusammengesetzt.

(7) Bei Dreiecks- und Diagonalmatrizen stehen die Eigenwerte auf der Hauptdiagonalen.

BEMERKUNG: Mit Hilfe der Eigenwerte können die Stabilitätseigenschaften dynamischer Systeme (wie z. B. GDGL) für $T \to \infty$ untersucht werden. Oft ist hierzu auch nur der größte Eigenwert von Interesse. In solchen Fällen sind numerische Methoden hinreichend, welche lediglich den größten Eigenwert approximieren.

BEISPIEL Man betrachte das DGL-System aus dem Räuber-Beute-Modell:

$$\begin{aligned} x'(t) &= \quad ax - bxy := f_1(t,x),\ x = (x,y) \\ y'(t) &= -cy + dxy := f_2(t,x) \end{aligned}$$

(a) Jacobi-Matrix:

$$f'_x(t,x) = \begin{pmatrix} \partial_x f_1 & \partial_y f_1 \\ \partial_x f_2 & \partial_y f_2 \end{pmatrix} = \begin{pmatrix} a - by & -bx \\ dy & -c + dx \end{pmatrix}$$

(b) Charakteristisches Polynom:

$$p_A(\lambda) = \det(A - \lambda I) = \det \begin{pmatrix} a - by - \lambda & -bx \\ dy & -c + dx - \lambda \end{pmatrix}$$

Hier ist schnell ersichtlich (die Rechnung wird dem Leser überlassen), dass $\lambda \in \mathbb{C}$ sein kann. Dies rechtfertigt, dass die Analyse der numerischen Stabilität gleich für komplexwertige λ durchgeführt wird.

1.6.2. Lineare Stabilitätsanalyse

Bei der linearen Stabilitätsanalyse werden modellhaft lineare Probleme betrachtet. Bekannt sind für

$$u'(t) = \lambda\, u(t), \quad u(t_0) = u_0$$

die drei Fälle

$$\begin{aligned} &\mathfrak{Re}(\lambda) < 0 \Rightarrow |u(t)| = |u_0| \exp\big(\mathfrak{Re}(\lambda)\big) \to 0, \\ &\mathfrak{Re}(\lambda) = 0 \Rightarrow |u(t)| = |u_0| \exp\big(\mathfrak{Re}(\lambda)\big) \equiv |u_0|, \end{aligned}$$

$$\mathfrak{Re}(\lambda) > 0 \Rightarrow |u(t)| = |u_0| \exp\big(\mathfrak{Re}(\lambda)\big) \to \infty.$$

Man interessiert sich hier insbesondere für den ersten Fall, wenn die kontinuierliche Lösung $u(t)$ beschränkt ist, i. e. $\sup_{t>0} \|u(t)\| < \infty$. Kann das numerische Verfahren dann ebenfalls beschränkte Lösungen liefern, sprich gilt $\sup_{n>0} \|y_n\| < \infty$? Ob das gilt, ist für den ersten Fall wesentlich kritischer, als für den dritten Fall, wo die kontinuierliche Lösung sowieso gegen ∞ strebt und ggf. Schrittweitenbeschränkungen aufgrund des Diskretisierungsfehlers notwendig sind.

Definition 1.6.1 Absolute Stabilität

Eine Einschrittmethode heißt für $\lambda h \neq 0$ absolut stabil, wenn diese, angewendet auf das Modellproblem, für $\mathfrak{Re}(\lambda) < 0$ beschränkte diskrete Lösungen $y_n,\ n = 1, \ldots, N$ erzeugt, d. h.

$$\sup_n \|y_n\| < \infty.$$

Der konkrete Nachweis erfolgt mit Hilfe des sogenannten Verstärkungsfaktors

$$\omega(z) = \omega(h\lambda).$$

Für das Polygonzugverfahren gilt

$$y_n = \underbrace{(1 + h\lambda)}_{=\omega(h\lambda)} y_{n-1}.$$

Für $\mathfrak{Re}(\lambda) < 0$ muss $0 < |\omega(z)| \leq 1$ sein, damit $y_n \leq y_{n-1} \leq \ldots \leq y_0$ gilt.

Definition 1.6.2 Gebiet absoluter Stabilität, Stabilitätsgebiet

Man bezeichnet mit

$$SG = \big\{ z = h\lambda \in \mathbb{C} \ \big|\ 0 < |\omega(z)| \leq 1 \big\}$$

das Gebiet der absoluten Stabilität.

Falls $\lambda \ll -1$ ist, so muss h entsprechend klein gewählt werden, um $|\omega(z)| \leq 1$ zu erfüllen. Damit lässt sich Taylor (Runge-Kutta) erweitern. Man betrachte das Taylor-Verfahren Stufe R:

$$y_n = y_{n-1} + h \sum_{r=1}^{R} \frac{h^{r-1}}{r!} \underbrace{f^{(r-1)}(t_{n-1}, y_{n-1})}_{f(t,u)=\lambda u} = y_{n-1} + \underbrace{h \sum_{r=1}^{R} \frac{h^{r-1}}{r!} \lambda^r}_{=:c}\, y_{n-1}$$

Nun ist die Frage, wann $|y_n| \leq c|y_{n-1}|$ gilt. Konkret ist der Verstärkungsfaktor $\omega(z)$ durch

$$\omega(z) = 1 + h \sum_{r=1}^{R} \frac{h^{r-1}}{r!} \lambda^r, \quad z = h\lambda$$

gegeben. Dadurch erhält man beispielsweise

$$R = 1: \quad \omega(z) = 1 + h\lambda = 1 + z,$$
$$R = 2: \quad \omega(z) = 1 + h\big(\lambda + \tfrac{h}{2}\,\lambda^2\big).$$

Nach längeren Rechnungen erhält man damit die folgenden Stabilitätsgebiete (graphisch):

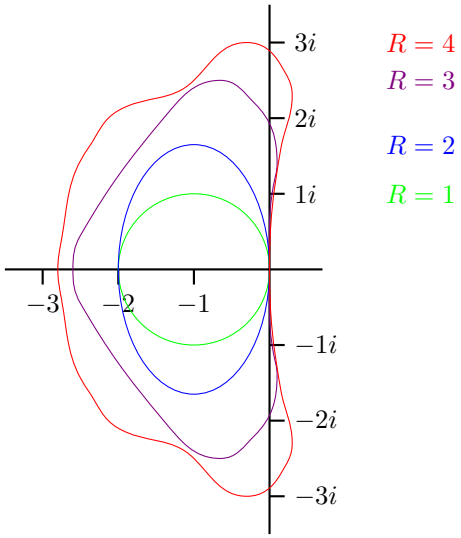


Abbildung 1.10.: Stabilitätsgebiete für explizite Runge-Kutta-Verfahren bis $R = 4$.

Bemerkung: Man sieht, dass ab $R \geq 3$ die Stabilitätsgebiete teilweise im positiven reellen Bereich liegen. Die Stabilitätsgebiete sind oft schwierig zu berechnen, doch in vielen Fällen reicht es zu wissen, wie sich der Realteil verhält. Dies führt auf das sogenannte Stabilitätsintervall SI.

Definition 1.6.3 Stabilitätsintervall

Das Stabilitätsintervall ist definiert durch

$$SI := \big\{ z \in \mathbb{R} \;\big|\; |\omega(z)| \leq 1 \big\}.$$

Beispiel

$$SI = \begin{cases} [\,-2,\,0\,], & R = 1 \text{ (Polygonzug)} \\ [\,-2,\,0\,], & R = 2 \\ [\,-2.51\ldots,\,0\,], & R = 3 \\ [\,-2.78\ldots,\,0\,], & R = 4 \end{cases}$$

Wie oben bemerkt, benötigt man für $\lambda \ll -1$ sehr kleine Schrittweiten h. In diesem Fall wären Formeln mit weit nach links reichendem SI bzw. SG wünschenswert. Offensichtlich ist dies jedoch bei den expliziten Taylor-/Runge-Kutta-Formeln nicht erfüllt. Hierfür wurde der stärkere Stabilitätsbegriff der A-Stabilität[9] eingeführt:

Definition 1.6.4 A-Stabilität

Eine Differenzenmethode heißt A-stabil, wenn für ihr SG

$$\left\{ z = \lambda h \in \mathbb{C} \;\middle|\; \mathfrak{Re}(z) \leq 0 \right\} \subset SG$$

gilt.

Insbesondere ist für A-stabile Verfahren $\mathfrak{Re}(z) \to -\infty$ möglich, wodurch selbst für $\lambda \ll -1$ die Schrittweite im Prinzip beliebig gewählt werden kann. Damit wären dann keine kleinen Schrittweiten notwendig, um numerische Stabilität zu gewährleisten.

Satz 1.6.5 Untersuchung der A-Stabilität von expliziten Differenzenverfahren

Explizite Verfahren können nicht A-stabil sein.

Beweis zu Satz 1.6.5
Hier soll der Beweis für $R = 1$ genügen. Sei also $\omega = 1 + \lambda h = 1 + z$. Für $|z| \to \infty$ gilt $|\omega| \to \infty$, d. h. $|\omega(z)| \leq 1$ ist sofort verletzt. Insbesondere sind für $\mathfrak{Re}(z) \to -\infty$ eben jene z nicht mehr im Stabilitätsgebiet enthalten. □

Bemerkung: Im Prinzip lässt sich die (skalare) lineare Stabilitätsanalyse auf Systeme erweitern. Allerdings gibt es einige Stellen in den entsprechenden Beweisen, die so nicht mehr gelten, wodurch gesonderte Betrachtungen notwendig werden. Bei Interesse kann das in (Rannacher 2017, S. 80–82) anhand eines ausführlichen Beispiels nachgeguckt werden.

[9] A-Stabilität ist in der Numerik für Differentialgleichungen ein zentraler Begriff.

1.6.3. Anwendungen der A-Stabilität

Die Anwendungen sind sogenannte steife Probleme[10]. Im Allgemeinen tritt Steifheit bei Systemen von mindestens zwei gekoppelten Differentialgleichungen auf, worin die Differentialgleichungen unterschiedliche zeitliche Entwicklungen aufweisen. In Abbildung 1.11 sieht man das einmal visualisiert.

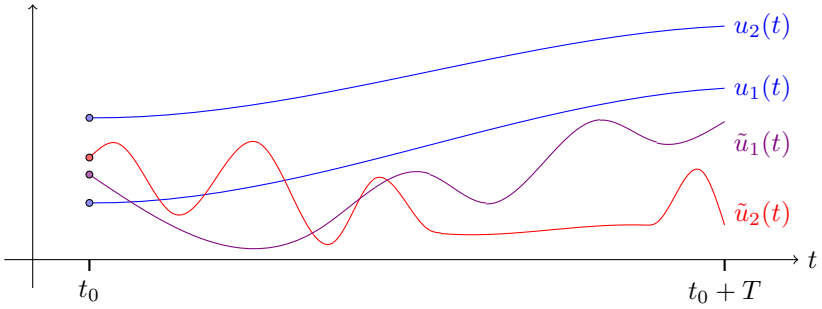


Abbildung 1.11.: Illustration von steifen $(\tilde{u}_i(t))$ und nicht steifen $(u_i(t))$ DGL-Systemen.

Die Konstruktionen A-stabiler Verfahren sind immer impliziter Natur und entstehen durch Schachtelung verschiedener numerischer Verfahren, bspw. muss in jedem t_n zur Bestimmung von y_n ein Fixpunktverfahren wie Newton verwendet werden. Man hat es dabei fast immer mit nicht-linearen Problemen zu tun.

Definition 1.6.6 Steifheit

Eine AWA heißt entlang einer Lösung $u(t)$ steif, wenn für die Eigenwerte $\lambda(t)$ der Jacobi-Matrix $f'_u\big(t, u(t)\big)$

$$\kappa(t) = \frac{\max\limits_{\mathfrak{Re}(\lambda)<0} |\mathfrak{Re}(\lambda)|}{\min\limits_{\mathfrak{Re}(\lambda)<0} |\mathfrak{Re}(\lambda)|} \gg 1$$

gilt. Die Größe $\kappa(t)$ wird Steifigkeitsrate genannt.

Die Realteile der Eigenwerte von $f'_u(t, u)$ stehen in enger Beziehung zur Lipschitz-Konstanten

$$\|f(t,u) - f(t,\tilde{u})\| \leq \max_{(t,\xi)\in D} \|f'_u(t,\xi)\| \, \|u - \tilde{u}\| \leq L_f \|u - \tilde{u}\|,$$

sowie

$$|\mathfrak{Re}(\lambda_{\max})| \leq \|f'_u(t,u)\|.$$

[10] Hierbei ist die Literatur nicht ganz eindeutig. Salopp sind das Probleme mit großen oder sehr unterschiedlichen Koeffizieten λ. Stringenter ist die hier im Skript gegebene Definition.

In der stringenten Interpretation wird Steifigkeit lediglich für Systeme von DGL definiert, wo die entsprechenden Eigenwerte stark unterschiedliche Abklingverhalten aufweisen, wo besondere numerische Stabilitätseigenschaften vonnöten sind. Dies kann z. B. beim Räuber-Beute-Modell der Fall sein. Im skalarwertigen Fall, also insbesondere bei

$$u'(t) = \lambda\, u(t), \quad u \in \Omega \subset \mathbb{R}^1,$$

spricht man im stringenten Sinne nicht von einem steifen Problem, nur weil $\lambda \ll -1$ ist. Hier muss bereits aus Konsistenzgründen, also um den Diskretisierungsfehler klein zu halten, mit kleinen Schrittweiten gerechnet werden.

BEISPIEL

$$u'(t) = A\, u(t), \quad u(0) = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \quad A = \begin{pmatrix} -21 & 19 & -20 \\ 19 & -21 & 20 \\ 40 & -40 & -40 \end{pmatrix}$$

Die Matrix-Einträge haben alle dieselbe Größenordnung, weshalb nicht unbedingt ein steifes System erwartet wird. Die Eigenwerte der Matrix sind allerdings $\lambda_1 = -2$ (reellwertig) und $\lambda_{2,3} = -40 \pm 40i$ (komplexwertig), sind also sehr unterschiedlich. Zu dieser AWA gibt es die Lösung

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}e^{-2t} + \frac{1}{2}e^{-40t}\big[\cos(40t) + \sin(40t)\big] \\ \frac{1}{2}e^{-2t} - \frac{1}{2}e^{-40t}\big[\cos(40t) + \sin(40t)\big] \\ -e^{-40t}\big[\cos(40t) - \sin(40t)\big] \end{pmatrix}.$$

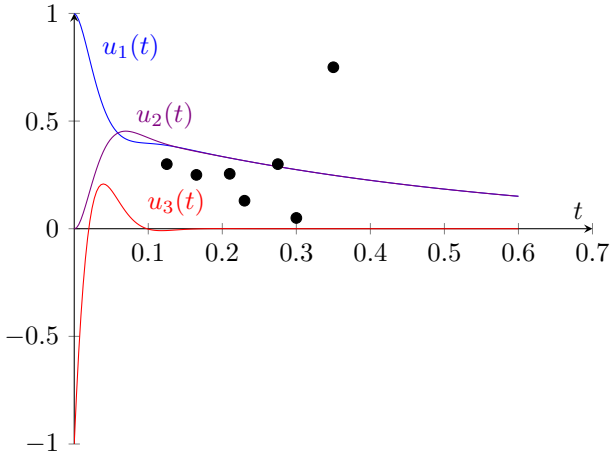


Abbildung 1.12.: Lösungskomponenten einer steifen AWA und instabile numerische Approximation "•".

Aus Abbildung 1.12 kann man ein paar Beobachtungen machen:

(a) Die Lösungskomponenten weisen keine starken Oszillationen auf.

(b) Für $t \in [\,0,\, 0.1\,]$ gibt es große Unterschiede. Intuitiv würde man also eine Schrittweitenrestriktion als notwendig empfinden.

(c) Ab $t \geq 0.1$ ist $u_1 \approx u_2$ und u_3 nähert sich asymptotisch der Null an.

Insgesamt ist aufgrund des Diskretisierungsfehlers wahrscheinlich keine Schrittweitenrestriktion notwendig. Wenn man mit dem Polygonzug-Verfahren rechnet, dann gilt $|\omega(z)| \leq 1$ zum Erhalt der numerischen Stabilität mit $\omega(z) = 1 + z$. D. h. mit $\lambda_1 = -2$ und $\mathfrak{Re}(\lambda_{2,3}) = -40$ gilt:

$$
\begin{aligned}
&|1 + h\lambda_1| = |1 - 2h| \leq 1 \\
&\qquad \rightarrow \quad 1 - 2h \leq 1 \quad \Leftrightarrow \quad -2h \leq 0 \quad \Rightarrow \quad h \geq 0 \\
&\qquad \rightarrow \quad -(1 - 2h) \leq 1 \quad \Leftrightarrow \quad 2h \leq 2 \quad \Rightarrow \quad h \leq 1
\end{aligned}
$$

$$
\begin{aligned}
&|1 + h\,\mathfrak{Re}(\lambda_{2,3})| = |1 - 40h| \leq 1 \\
&\qquad \rightarrow \quad 1 - 40h \leq 1 \quad \Leftrightarrow \quad -40h \leq 0 \quad \Rightarrow \quad h \geq 0 \\
&\qquad \rightarrow \quad -(1 - 40h) \leq 1 \quad \Leftrightarrow \quad 40h \leq 2 \quad \Rightarrow \quad h \leq \tfrac{1}{20}
\end{aligned}
$$

Also muss zum Erhalt der numerischen Stabilität mit $h \leq \frac{1}{20}$ gerechnet werden. Für $h > \frac{1}{20}$ treten mindestens für die Komponenten $y_{(2)}$ und $y_{(3)}$ physikalisch unerwünschte Oszillationen auf. Dies wird einerseits genauer in den Übungen untersucht. Andererseits verweisen wir auch auf Abschnitt 1.8.2.

1.6.4. Implizite Verfahren

Im Prinzip ist die Konstruktion impliziter Runge-Kutta-Verfahren höherer Ordnung analog zu den korrespondierenden expliziten Methoden möglich und wird in Anwendungen auch genutzt. Hier sollen zwei prominente Vertreter niedrigerer Ordnung vorgestellt werden: Impliziter Euler und die Trapezregel. Gegeben sei

$$u'(t) = \lambda\, u(t), \quad u(t_0) = u_0$$

Unter der Verfahrensvorschrift des impliziten Euler-Verfahrens schließt man dann

$$
\begin{aligned}
y_n = y_{n-1} + h\lambda y_n \quad &\Leftrightarrow \quad y_n - h\lambda y_n = y_{n-1} \quad \Leftrightarrow \quad (1 - h\lambda) y_n = y_{n-1} \\
&\overset{h\lambda \neq 1}{\Leftrightarrow} \quad y_n = \frac{y_{n-1}}{1 - h\lambda} \qquad \Rightarrow \quad \omega(z) = \frac{1}{1 - z}.
\end{aligned}
$$

Für $z = h\lambda \rightarrow -\infty$ ist die A-Stabilität also erfüllt. Ferner ist der implizite Euler sogar stark A-stabil. Allerdings dämpft der implizite Euler zu stark. Für gewisse positive λ ist die Metho-

de absolut-stabil. D. h. die kontinuierliche Lösung wächst an, während die numerische Lösung beschränkt bleibt. Dies ist insbesondere für Probleme ungeeignet, welche gewisse Erhaltungseigenschaften (z. B. Energie) aufweisen, und wo das implizite Euler-Verfahren zu stark dämpft, da dieses numerisch zu dissipativ ist. Für die Trapezregel erhält man auf dieselbe Weise

$$\omega(z) = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z}.$$

Hier ist $|\omega(z)| \leq 1$ für $\mathfrak{Re}(z) \leq 0$, womit die Trapezregel A-stabil ist. Weil für $z \to -\infty$ ggf. leichte Instabilitäten auftreten können, ist die Trapezregel allerdings nur (schwach) A-stabil, d. h. $|\omega(z)| \sim 1$. Hier sei nun noch für die Trapezregel zusammengefasst:

- $|\omega(z)| > 1$ für $h\lambda > 0$ (instabil, allerdings sowieso uninteressant in der linearen Stabilitätsanalyse)
- $|\omega(z)| = 1$ für $h\lambda = 0$
- $|\omega(z)| < 1$ für $h\lambda < 0$

Insgesamt gilt insbesondere für das implizite Euler-Verfahren und die Trapezregel im Vergleich:

- $|\omega_{\text{Euler}}(z)| \to 0$ für $z \to -\infty$
- $|\omega_{\text{Trapez}}(z)| \to 1$ für $z \to -\infty$

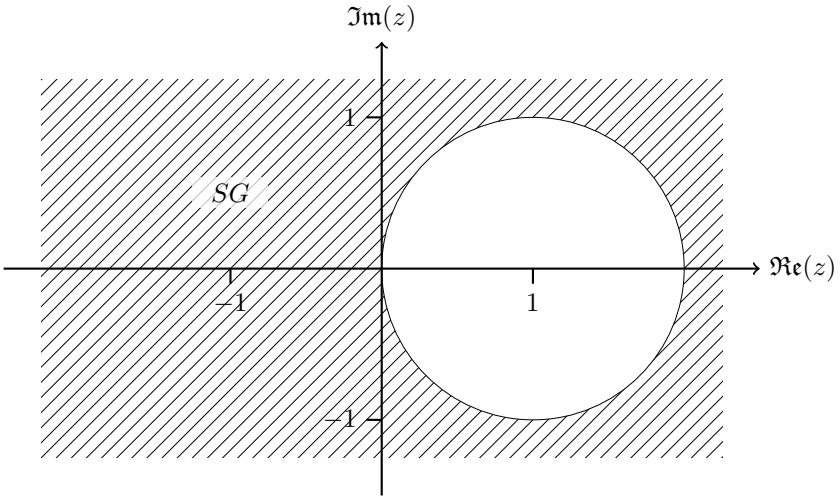


Abbildung 1.13.: Stabilitätsgebiet des impliziten Euler-Verfahrens.

1.7. Numerische Lösung impliziter Verfahren

Hier sei beispielhaft das implizite Euler-Verfahren

$$y_n = y_{n-1} + h_n\, f(t_n, y_n)$$

genutzt. Numerisch wird das Ganze als ein Nullstellenproblem

$$0 = -y_n + y_{n-1} + h_n\, f(t_n, y_n) \Rightarrow y_n = y_n \underbrace{-y_n + y_{n-1} + h_n\, f(t_n, y_n)}_{=-g(y_n)}$$

betrachtet. Wie man leicht sieht gilt $g(y_n) = 0$. Die Denkweise in Form eines Nullstellenproblems vereinfacht erheblich der Herleitung von Lösungsalgorithmen für komplexere Problemstellungen wie gekoppelte und nichtlineare Differentialgleichungen.

Als Iteration gefasst liegt die Gleichung

$$y_n^{(k)} = y_n^{(k-1)} - g(y_n^{(k-1)}) = y_n^{(k-1)} - y_n^{(k-1)} + y_{n-1} + h_n\, f(t_n, y_n^{(k-1)})$$

vor und mit Einführung eines Relaxationsparameters $\theta > 0$

$$y_n^{(k)} = \theta h_n\, f(t_n, y_n^{(k-1)}) + y_{n-1}.$$

Man hat somit eine Fixpunktiterationsvorschrift erhalten mit folgendem Algorithmus:

Algorithmus 6.

1. *Setze bei* t_n*:* $y_n^{(0)}$ *(oft* $y_n^{(0)} := y_{n-1}$*)*
2. *Für* $k = 1, 2, 3, \ldots$
3. *Berechne:*
   $$y_n^{(k)} = y_n^{(k-1)} - \theta\big[y_n^{(k-1)} - y_{n-1} - h_n\, f(t_n, y_n^{(k-1)})\big]$$
4. *Abbruch, falls* $\|y_n^{(k)} - y_n^{(k-1)}\| < \mathrm{TOL}$ *bzw. eines der u.g. Kriterien*

Bemerkung: Der Parameter θ ist ein typischer Relaxationsparameter, der geeignet gewählt werden muss, sodass die Fixpunktiteration konvergiert.

Aus Numerik I ist bekannt, dass Fixpunktverfahren maximal Konvergenzordnung 1 haben. Die Idee ist also, das Fixpunktverfahren durch das Newton-Verfahren zu ersetzen, welches superlineare Konvergenz (bzw. bei entsprechenden Voraussetzungen Ordnung 2) aufweist. Hier wird das Nullstellenproblem also in

$$g(y_n) = 0, \quad g(y_n) = y_n - y_{n-1} - h_n\, f(t_n, y_n)$$

umgeschrieben. Darauf lässt sich dann das Newton-Verfahren (Defekt-Korrektur) anwenden:

Algorithmus 7.

1. $\underbrace{g'(y_n^{(k-1)})}_{Jacobi\text{-}Matrix} \cdot \underbrace{\delta y}_{Update} = -g(y_n^{(k-1)})$
2. $y_n^{(k)} = y_n^{(k-1)} + \delta y$

   $\big[$*mit* $g'(y_n) = 1 - h_n\, f'_y(t_n, y_n)$ *(skalarwertig)*$\big]$
3. *Abbruch:*

$$\|y_n^{(k)} - y_n^{(k-1)}\| < \mathrm{TOL} \qquad \textit{(absolutes fehler-basiertes Kriterium)}$$

oder $\|g(y_n^{(k)})\| < \mathrm{TOL}$ *(absolutes residuen-basiertes Kriterium)*

oder $\|y_n^{(k)} - y_n^{(k-1)}\| < \mathrm{TOL}\,\|y_n^{(k)}\|$ *(relatives fehler-basiertes Kriterium)*

oder $\|g(y_n^{(k)})\| < \mathrm{TOL}\,\|g(y_n^{(0)})\|$ *(relatives residuen-basiertes Kriterium)*

Beispiele

(a) Man löse $u' = \lambda u$ mit dem impliziten Euler-Verfahren:

1. $f(t_n, y_n) = \lambda y_n$
2. $g(y_n) = y_n - y_{n-1} - h_n \lambda y_n$

   $g'(y_n) = 1 - h_n \lambda$

   Damit erhält man

$$(1 - h_n\lambda)\delta y = -\big(y_n^{(k-1)} - y_{n-1} - h_n\lambda y_n^{(k-1)}\big)$$

   als Defekt-Schritt. Auf der linken Seite taucht hierbei kein $y_n^{(k-1)}$ auf, weil $u' = \lambda u$ linear ist.

(b) Gegeben sei $u' = \lambda u^3$.

1. $g(y_n) = y_n - y_{n-1} - h_n \lambda y_n^3$
2. $g'(y_n) = 1 - h_n \lambda 3 y_n^2$

   Damit erhält man

$$\big(1 - 3h_n\lambda(y_n^{(k-1)})^2\big)\delta y = -\big(y_n^{(k-1)} - y_{n-1} - h_n\lambda(y_n^{(k-1)})\big)$$

   als Defekt-Schritt.

1.8. Numerische Beispiele

1.8.1. Exkurs (erweitertes Beispiel): Zeitabhängige Wäscheleine (Wärmeleitung)

Die Steifheit bedeutet, dass große Faktoren oder Konstanten auftreten. Das ist besonders in PDGL ein generisches Problem. Im Folgenden wird ein von zuvor (NM I / Alg. Math.) bekanntes Beispiel erweitert. Dafür sei zuerst auf das Wäscheleine-Modell

$$-u''(x) = f, \quad u(0) = u(1) = 0$$

erinnert. Dieses Modell soll nun um eine Schwingung erweitert werden:

$$\partial_t u(t,x) - u''(t,x) = f(t,x) \text{ [11]}$$
$$u(t,0) = u(t,1) = 0 \text{ (Randbedingungen)}$$
$$u(t_0,x) = u(t,x) = u_0 \text{ (Anfangsbedingung)}$$

Zuerst wird eine Diskretisierung von $u''(t,x)$ mit dem Differenzenquotienten im Ort x vorgenommen:

$$-u''(t,x) = -\frac{u(t,x-\Delta x) - 2u(t,x) + u(t,x+\Delta x)}{\Delta x^2} + \mathcal{O}(\Delta x^2), \quad \Delta x: \text{ Ortsgitterweite}$$

Als nächstes kommt die Zeitdiskretisierung ins Spiel, bspw. über das Polygonzugverfahren:

$$\partial_t u(t,x) \approx \frac{\overbrace{u(t_n,x)}^{u_n} - \overbrace{u(t_{n-1},x)}^{u_{n-1}}}{h}, \quad h = t_n - t_{n-1} \text{ (Zeitschrittweite)}$$

Zuletzt werden beide Diskretisierungen kombiniert:

$$\frac{u_n - u_{n-1}}{h} - \frac{u(t_{n-1},x-\Delta x) - 2u(t_{n-1},x) + u(t_{n-1},x+\Delta x)}{\Delta x^2} = f(t_{n-1},x)$$
$$\Rightarrow u_n = u_{n-1} + \underbrace{\frac{h}{\Delta x^2} \cdot \big(u(t_{n-1},x-\Delta x) - 2u(t_{n-1},x) + u(t_{n-1},x+\Delta x)\big) + h\, f(t_{n-1},x)}_{=:\tilde{f}(t_{n-1},x)}$$

Hier stellt sich nun die Frage, was zu L bzw. λ vom Vorhergehenden korrespondiert. Das ist gerade Term

$$\frac{h}{\Delta x^2} =: L.$$

[11] Diese Gleichung ist das Standardmodell der sogenannten Wärmeleitungsgleichung

Weil $L \to \infty$ für explizite Verfahren wegen der Schrittweitenbedingung in h ungünstig ist, sind die Konvergenzen in Ort Δx und Zeit h für $\Delta x \to 0$ und $h \to 0$ von wesentlichem Interesse. Allerdings gilt sowohl für festes wie auch für variables h (falls dieselben Nullfolgen für h und Δx verwendet werden) die Konvergenz $L \to \infty$ für $\Delta x \to 0$. Somit ist das hier ein steifes Problem. Für explizite Zeitschrittmethoden muss also h in Abhängigkeit von Δx^2 gewählt werden. D. h. für $\Delta x \to 0$ müssen die h sehr, sehr klein sein. Darum nutzt man hierbei oftmals implizite Methoden. Ganz konkret wird gerechnet:

$$\begin{aligned} u_n &= u_{n-1} - 2\frac{h}{\Delta x^2} u_{n-1} + \dots \\ &= \Big(1 - \frac{2h}{\Delta x^2}\Big) u_{n-1} \end{aligned}$$

Hieraus folgt

$$|\omega(z)| \leq 1 \quad \Leftrightarrow \quad \Big|1 - \frac{2h}{\Delta x^2}\Big| \leq 1.$$

Wie zuvor wird nun der Betrag aufgelöst. Dabei ist der erste Fall uninteressant, der zweite liefert hingegen

$$-\Big(1 - \frac{2h}{\Delta x^2}\Big) \leq 1 \quad \Rightarrow \quad h \leq \Delta x^2.$$

1.8.2. Numerische Simulationen mit Hilfe des Modellproblems

Hier folgend wird nun ein Teil aus (Amstutz und T. Wick 2022) übernommen, der im Zuge der SoSe2024-Vorlesung weiter modifiziert wurde.

Problem statement and discussion of results

Example 1.8.1. *Let $a = g - m$ be $a = 0.25$ or $a = -0.25$ or $a = -10$ (three test scenarios). Let the time interval be $(2011, 2014)$, i.e., $t_0 = 2011$ and $T = t_N = 2014$ and the length of the time interval is 3. The IVP is given by:*

$$y' = ay, \quad y_0 = y(t_0) = y(2011) = 2.$$

Use the forward Euler (FE), backward Euler (BE), and trapezoidal rule (TR) for the numerical approximation. Please observe the accuracy in terms of the discretization error and for (stiff) equations with a large negative coefficient $a = -10 \ll 1$ the behavior of the three schemes.

In order to calculate the convergence order α from numerical results, we make the following derivation. Let $P(k) \to P$ for $k \to 0$ be a converging process and assume that

$$P(k) - \tilde{P} = O(k^\alpha).$$

Therein, in this specific case k is the time step size with $k = t_n - t_{n-1}$. Here $\tilde{P}$ is either the exact limit P (in case it is known) or some 'good' approximation to it. Let us assume that three numerical solutions are known (this is the minimum number if the limit P is not known). That is

$$P(k), \quad P(k/2), \quad P(k/4).$$

Then, the convergence order can be calculated via the formal approach $P(k) - \tilde{P} = ck^\alpha$ with the following formula:

Proposition 1.8.2 Computationally-obtained convergence order

Given three numerically-obtained values $P(k), P(k/2)$ and $P(k/4)$, the convergence order can be estimated as:

$$\alpha = \frac{1}{log(2)} log\Big(\Big|\frac{P(k) - P(k/2)}{P(k/2) - P(k/4)}\Big|\Big). \tag{1}$$

The order α is an estimate and heuristic because we assumed a priori a given order, which strictly speaking we have to proof first.

Proof. We assume:

$$\begin{aligned} P(k) - P(k/2) &= (ck)^\alpha + O(k^{\alpha-1}), \\ P(k/2) - P(k/4) &= (ck/2)^\alpha + O((k/2)^{\alpha-1}). \end{aligned}$$

First, we have

$$P(k/2) - P(k/4) = \frac{1}{2^\alpha}(ck)^\alpha + O((k/2)^{\alpha-1}).$$

We insert the first approximation into the second one for $(ck)^\alpha$ and re-arrange:

$$\begin{aligned} P(k/2) - P(k/4) &= \frac{1}{2^\alpha}\Big(P(k) - P(k/2)\Big) + O(k^{\alpha-1}) \\ \Rightarrow \quad 2^\alpha &= \frac{P(k) - P(k/2)}{P(k/2) - P(k/4)} \\ \Rightarrow \quad \alpha &= \frac{1}{\log(2)} \log\Big(\Big|\frac{P(k) - P(k/2)}{P(k/2) - P(k/4)}\Big|\Big). \end{aligned}$$

□

Computations and their discussion

In the following table and explanations, we present our results for the actual value y_N at end time $T = 2014$, the relative and absolute errors for test case 1 ($a = 0.25$) on three mesh levels, i.e., three different time step sizes. Here, N is the end time index that equals the number of

(time) intervals. Three computations is the minimum number to apply the above formula for the EOC (expected order of convergence) in terms of α.

Scheme	N	k	y_N	e_N (rel.)	e_N (abs.)	EOC(alpha)
FE	3	1	3.9062	0.077409	0.32775	---
BE	3	1	4.7407	0.11968	0.50674	---
TR	3	1	4.2507	0.0039511	0.016729	---
FE	6	0.5	4.0546	0.042378	0.17943	---
BE	6	0.5	4.4564	0.052521	0.22237	---
TR	6	0.5	4.2381	0.00097934	0.0041465	---
FE	12	0.25	4.1398	0.022253	0.09422	0.7997
BE	12	0.25	4.3389	0.024764	0.10485	1.2748
TR	12	0.25	4.2350	0.00024431	0.0010344	2.0154

- The absolute error at the end time T is computed as

  $$e_N = |y(T) - y_N|,$$

  and the relative error is computed by

  $$e_N = |y(T) - y_N|/|y(T)|,$$

  where $y(T)$ is the exact solution and y_N the numerical approximation at the end time value at the final numerical step N.

- In the second column, i.e., $3, 6, 12$, the number of steps (= number of intervals, i.e., so called mesh elements - speaking in PDE terminology) are given. In the column after, the errors are provided.

- In order to compute numerically the expected convergence order α with the help of formula (1), we work with $k = k_{max} = 1.0$. Then we identify in the above table that $P(k_{max}) = P(1) = |y(T) - y_3|, P(k_{max}/2) = P(0.5) = |y(T) - y_6|$ and $P(k_{max}/4) = P(0.25) = |y(T) - y_{12}|$.

- We monitor that doubling the number of intervals (i.e., halving the step size k) reduces the error in the forward and backward Euler scheme by a factor of 2. This is (almost) linear convergence, which is confirmed by using Formula 1 yielding $\alpha = 0.79970$ and $\alpha = 1.2748$, respectively. The trapezoidal rule is much more accurate (for instance using $N = 12$ the error is much smaller) and we observe that the error is reduced by a factor of 4. Thus

quadratic convergence is detected. Here the 'exact' order on these three mesh levels is $\alpha = 2.0154$.

- A further observation is that the forward Euler scheme is unstable for $N = 16$ and $a = -10$ and has a zig-zag curve, whereas the other two schemes follow the exact solution and the decreasing exp-function. But for sufficiently small step sizes, the forward Euler scheme is also stable which we know from our A-stability calculations. These step sizes can be explicitely determined for this ODE model problem and shown below.

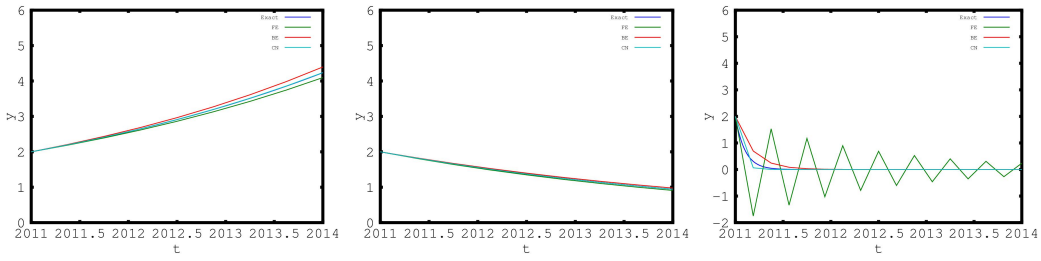


Abbildung 1.14.: plots of the solution to test 1, test 2 and test 3 with $N = 16$ [12](f. l. t. r.)

Treatment of the instability of the forward Euler method

With the help of definition (1.6.2) let us understand how to choose stable step sizes k for the forward Euler method. The convergence interval reads:

$$|1 + z| \leq 1 \quad \Rightarrow \quad |1 + ak| \leq 1$$

In test 3, $a = -10$, which yields:

$$|1 + z| \leq 1 \quad \Rightarrow \quad |1 - 10k| \leq 1$$

Thus, we need to choose a k that fulfills the previous relation. In this case this, $k < 0.2$ is calculated. This means that for all $k < 0.2$ we should have convergence of the forward Euler method and for $k \geq 0.2$ non-convergence (and in particular no stability!). We perform the following additional tests:

- Test 3a: $N = 10$, yielding $k = 0.3$;
- Test 3b: $N = 15$, yielding $k = 0.2$; exactly the boundary of the stability interval;

[12]Here, $N = 16$ corresponds to a step size $k = 0.18$ which is slightly below the critical step size for convergence (see Section 1.8.2). Thus we observe the instable behavior of the forward Euler method, but also see slow convergence towards the continuous solution.

- Test 3c: $N = 16$, yielding $k = 0.18$; from before;
- Test 3d: $N = 20$, yielding $k = 0.15$.

The results of test 3a,3b,3d are provided in Figure 1.15 and visualize very nicely the theoretically predicted behavior.

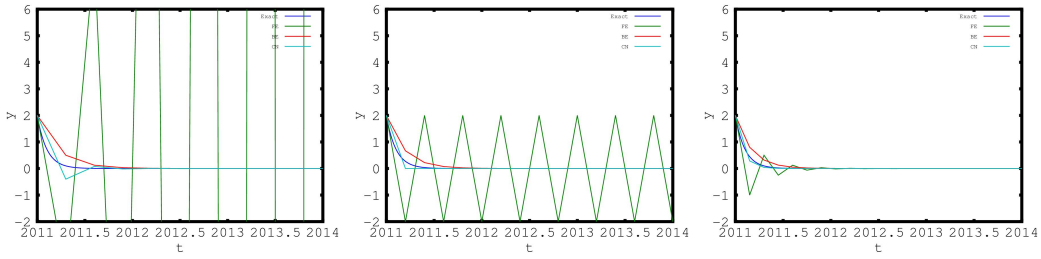


Abbildung 1.15.: Example 1.8.1: tests 3a,3b,3d: Blow-up, constant zig-zag non-convergence, and convergence of the forward Euler method.

Exercise 1. *Consider the ODE-IVP:*

$$y'(t) = ay(t), \quad y(t_0) = 7,$$

where $t_0 = 5$ *and* $T = 11$.

1. *Implement the backward Euler scheme. and set the model parameter to* $a = 2$
2. *Run a second test with* $a = -2$.
3. *Implement the forward Euler scheme. What is the critical step size (Hint: Determine the stability interval).*
4. *Implement the trapezoidal rule (also known as Crank-Nicolson).*
5. *Perform a computational analysis and detect the convergence order.*

1.9. Lineare Mehrschrittmethoden

Ziel ist es nun, mit möglichst niedrigen Ableitungen Verfahren höherer Ordnung zu konstruieren. In der Praxis werden diese in der Tat häufig eingesetzt. Die Idee ist hierbei, nicht nur den vorherigen Schritt (t_{n-1}, y_{n-1}) einzubeziehen, sondern auch weitere vorherige Schritte

$$(t_{n-2}, y_{n-2}), \ (t_{n-3}, y_{n-3}), \ldots$$

Damit erhofft man sich bessere Genauigkeit, Stabilität[13] und Effizienz im Vergleich zu Einschrittmethoden derselben Konvergenzordnung. Die Klasse von Verfahren werden (lineare) Mehrschrittmethoden (LMM) genannt. Es gibt zwei Arten der Konstruktion:

(1) Numerische Integration über Quadraturformeln

(2) Numerische Differentiation (BDF [14])

1.9.1. Konstruktion per Integration der rechten Seite

Für die Konstruktion per numerischer Integration mit Quadraturformeln sei für

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

und die Integralbeziehung

$$u(t_n) = u(t_{n-\sigma}) + \int_{t_{n-\sigma}}^{t_n} f\big(s, u(s)\big)\,\mathrm{d}s \tag{1}$$

für festes $\sigma \in \mathbb{N}$ erinnert. Die Idee ist somit die Approximation des Integrals

$$\int_{t_{n-\sigma}}^{t_n} f\big(s, u(s)\big)\,\mathrm{d}s$$

mittels einer Quadraturformel. Konkret heißt das, zunächst $f\big(s, u(s)\big)$ mittels Interpolationspolynomen zu approximieren und dieses Polynom dann zu integrieren. In Formeln lautet die Aufgabe dann, $f\big(s, u(s)\big)$ mit Polynomen $p_m(s)$ vom Grad m in den Gitterpunkten

$$t_{k-\mu}, \quad 0 \le \mu \le m$$

zu approximieren. Zuerst wird hier die Interpolationsbedingung

$$p_m(t_{k-\mu}) = f\big(t_{k-\mu}, u(t_{k-\mu})\big)$$

für $0 \le \mu \le m$ gestellt. Man baue ein Interpolationspolynom in Lagrangescher Darstellung

$$p_m(t) = \sum_{\mu=0}^{m} f\big(t_{k-\mu}, u(t_{k-\mu})\big) L_\mu^m(t), \quad L_\mu^m(t) = \prod_{\substack{l=0 \\ l \neq \mu}}^{m} \frac{t - t_{k-l}}{t_{k-\mu} - t_{k-l}}$$

[13] Lineare Mehrschrittmethoden weisen nicht immer bessere Stabilität auf. Diese muss man also nach charakterisieren des Problems beweisen.

[14] BDF kommt aus dem Englischen und steht für "Backward Differentiation Formulas"

mit den entsprechenden Restgliedern. Damit erhält man aus (1)

$$u(t_n) = u(t_{n-\sigma}) + \sum_{\mu=0}^{m} f\big(t_{k-\mu}, u(t_{k-\mu})\big) \cdot \int_{t_{n-\sigma}}^{t_n} L_\mu^m(s)\,\mathrm{d}s + \mathcal{O}(h^{m+2}) \tag{2}$$

Nun bleibt noch die Wahl eines σ, mit welchem das Integral aus (2) ausgewertet werden muss. Damit erhält man dann die LMM

$$y_n = y_{n-\sigma} + h \sum_{\mu=0}^{m} \beta_\mu\, f(t_{k-\mu}, y_{k-\mu}), \quad n \geq k$$

mit

$$\beta_\mu = \frac{1}{h} \int_{t_{n-\sigma}}^{t_n} L_\mu^m(s)\,\mathrm{d}s. \tag{3}$$

Beispiele

(a) Explizite Verfahren: Seien $\sigma = 1$ und $k = n-1$ (Adams-Bashforth-Formeln). Allgemein erhält man damit aus (3)

$$y_n = y_{n-1} + \sum_{\mu=0}^{m} f(t_{n-1-\mu}, y_{n-1-\mu}) \cdot \int_{t_{n-1}}^{t_n} L_\mu^m(t)\,\mathrm{d}t, \quad n \geq m-1.$$

So erhält man beispielsweise

$$m = 0: \quad y_n = y_{n-1} + h_n\, f(t_{n-1}, y_{n-1}) \qquad \text{(expliziter Euler)}$$

$$m = 1: \quad y_n = y_{n-1} + \tfrac{1}{2} h\,[\,3f_{n-1} - f_{n-2}\,]$$

$$m = 2: \quad y_n = y_{n-1} + \tfrac{h}{12}\,[\,23f_{n-1} - 16f_{n-2} + 5f_{n-3}\,]$$

Für $m = 0$ erhält man hier noch keine Mehrschrittmethode im stringenten Sinne, aber für $m \geq 1$ schon.

(b) Implizite Verfahren: Seien $\sigma = 1$ und $k = n$ (Adams-Moulton-Formeln). Damit erhält man aus (3)

$$y_n = y_{n-1} + \sum_{\mu=0}^{m} f_{n-\mu} \cdot \int_{t_{n-1}}^{t_n} L_\mu^m(t)\,\mathrm{d}t, \quad n \geq m.$$

Auch hierzu die ersten drei Verfahrensvorschriften:

$$m = 0: \quad y_n = y_{n-1} + h_n\, f(t_{n-0}, y_{n-0}) = y_{n-1} + h\, f(t_n, y_n) \qquad \text{(impliziter Euler)}$$

$$m = 1: \quad y_n = y_{n-1} + \tfrac{1}{2} h\, [\, f_n + f_{n-1}\,] \qquad \text{(Trapezregel)}$$

$$m = 2: \quad y_n = y_{n-1} + \tfrac{1}{12} h\, [\, 5f_n + 8f_{n-1} - f_{n-2}\,]$$

Hier sind jetzt sogar $m = 0, 1$ Einschrittmethoden und erst für $m \geq 2$ erhält man tatsächlich Mehrschrittverfahren.

Bemerkung: Bei LMMs müssen diese mit entsprechenden Startformeln initialisiert werden. Z. B. müssen bei der Adams-Moulton-Formel für $m = 2$ die Werte (t_{n-1}, y_{n-1}) und (t_{n-2}, y_{n-2}) vorliegen. Also müssen beim initialen Start für t_2 die entsprechenden Formeln für (t_1, y_1) und (t_0, y_0) bereits berechnet worden sein. Insbesondere ist (t_0, y_0) bereits durch den Startwert des Problems festgelegt. Demnach muss lediglich (t_1, y_1) berechnet werden. Dies kann mit einem Verfahren derselben Klasse (hier $m = 1$) geschehen (Selbststartprozedur).

1.9.2. Konstruktion per Differentiation der linken Seite

Nun zu den Konstruktionen mit BDFs: Konkret wird $u(t)$ in $u' = f(t, u)$ durch das Interpolationspolynom $p_m(t)$ der Ordnung m mit der Interpolationsbedingung

$$p_m(t_{k-\mu}) = u(t_{k-\mu}), \quad 0 \leq \mu \leq m$$

ersetzt. D. h. das Grundkonzept lautet

$$u'(t) \approx p'(t) = \sum_{\mu=0}^{m} u(t_{k-\mu}) \underbrace{(L_\mu^m)'(t)}_{\text{Ableitung}}$$

Damit erhält man

$$\sum_{\mu=0}^{m} u(t_{k-\mu})\, (L_\mu^m)'(t) = f(t, u) + \underbrace{\mathcal{O}(h^{m+1})}_{\text{Restglied der Lagrange-IP}}.$$

Hier sind jetzt k und m wählbar. Durch die natürliche Wahl $k = n$ erhält man die sogenannten Rückwärtsdifferenzenformeln (BDF)

$$\sum_{\mu=0}^{m} u(t_{n-\mu})\, (L_\mu^m)'(t) = f(t, u) \quad \overset{t = t_n}{\Rightarrow} \quad \sum_{\mu=0}^{m} y_{n-\mu}\, (L_\mu^m)'(t_n) = f(t_n, y_n). \tag{4}$$

Daraus ergibt sich ein Algorithmus zur Konstruktion eines BDF-Verfahrens:

1. `Wähle` m
2. `Baue` $L_\mu^m(t_n)$
3. `Leite` $(L_\mu^m)'(t_n)$ `ab`
4. `Werte` (4) `aus`

Beispiel Für $m = 1$ erhält man die Langrange-Polynome

$$L_0^1(t) = \frac{t - t_{k-1}}{t_k - t_{k-1}} \quad \Rightarrow \quad (L_0^1)'(t) = \frac{1}{t_k - t_{k-1}} = \frac{1}{h}$$
$$L_1^1(t) = \frac{t - t_k}{t_{k-1} - t_k} \quad \Rightarrow \quad (L_1^1)'(t) = \frac{1}{t_{k-1} - t_k} = -\frac{1}{h}$$

Die Ableitungen müssen nur noch in (4) eingesetzt werden, wodurch man

$$\frac{1}{h_n}(y_n - y_{n-1}) = f(t_n, y_n) \quad \Leftrightarrow \quad y_n = y_{n-1} + h_n\, f(t_n, y_n),$$

erhält, also den impliziten Euler. Für $m = 2, 3$ erhält man über dasselbe Vorgehen

$$m = 2: \quad y_n - \tfrac{4}{3}y_{n-1} + \tfrac{1}{3}y_{n-2} = \tfrac{2}{3}hf_n \qquad \text{(BDF 2)}$$

$$m = 3: \quad y_n - \tfrac{18}{11}y_{n-1} + \tfrac{9}{11}y_{n-2} - \tfrac{2}{11}y_{n-3} = \tfrac{6}{11}hf_n \qquad \text{(BDF 3)}$$

BDF 2 ist ein A-stabiles Verfahren mit der gleichen Ordnung wie die Trapezregel. BDF 3 hingegen ist nicht A-stabil und zeigt, dass implizite Formeln nicht automatisch A-stabil sein müssen. Das Gute an LMMs ist, dass sie sich problemlos auf DGL-Systeme erweitern lassen. Entsprechend der Einschrittformeln können nun die Verfahren auf Konsistenz, Stabilität sowie Konvergenz untersucht werden. Es stellt sich aufgrund der Beobachtung die Frage, welche Klassen von impliziten Verfahren überhaupt A-stabil sind. Es gilt der folgende Satz:

Satz 1.9.1 A-stabile LMM

(1) Explizite LMM können nicht A-stabil sein.

(2) Die Ordnung einer A-stabilen impliziten LMM kann nicht größer als 2 sein.

(3) Die A-stabile Trapezregel ist in dieser Hinsicht das optionale Verfahren, welches (z. B. im Vergleich zur BDF 2-Formel) die kleinste Fehlerkonstante besitzt.

Der Beweis kann in (Dahlquist 1963) nachgelesen werden.

BEMERKUNG: Nichtsdestotrotz sind aufgrund der höheren Konvergenzordnungen LMM mit $m \geq 3$ von Interesse. Hier wird dann mit schwächeren Stabilitätsbegriffen und Schrittweitenrestriktionen gearbeitet.

BEMERKUNG: Im Bereich der PDGL wird aufgrund der generischen Steifheit in der Tat sehr oft mit Trapezformel-ähnlichen Verfahren gearbeitet.

1.9.3. Prädiktor-Korrektor-Verfahren

Jetzt sind bereits implizite und explizite Verfahren bekannt, aber eines bleibt noch. Darum wird hier nun die Kombination von expliziten und impliziten Verfahren behandelt, die sogenannten Prädiktor-Korrektor-Verfahren. Doch wofür sind diese wichtig? Bei impliziten Verfahren löst man effektiv ein Nullstellenproblem, am Beispiel von einer LMM etwa

$$y_n - y_{n-\sigma} - h \sum_{\mu=0}^{m} \beta_\mu \, f(t_{k-\mu}, y_{k-\mu}) = 0.$$

Mit entsprechender Wahl von $k = n$ generiert man implizite Verfahren, die generell nicht anders lösbar sind, als durch Interpretation als Nullstellenprobleme. Für eine implizite LMM würde man also eine Fixpunktiteration

$$y_n^{(j+1)} = y_{n-\sigma} + h\beta_0 \, f\big(t_n, y_n^{(j)}\big) + h \sum_{\mu=1}^{m} \beta_\mu \, f(t_{n-\mu}, y_{n-\mu}).$$

lösen. Weil das Verfahren implizit ist, taucht y_n auf beiden Seiten der Gleichung auf, darum verwendet man die vorherige Iterierte auf der rechten Seite, bei der LMM also bei $\mu = 0$.

Man beachte, dass Startwerte unterschiedlich zu dem Anfangswert y_0 sind! Als Anfangswert ist y_0 immer fest und wird daher nicht über ein Fixpunktverfahren approximiert. Alle weiteren Werte $y_1, \ldots, y_N$ hingegen benötigen Startwerte $y_1^{(0)}, \ldots, y_N^{(0)}$, mit denen eine Approximation per Fixpunktiteration bestimmt werden kann.

Der Hintergrund der Kombination von expliziten und impliziten Verfahren ist, dass die Fehler des Anfangswertes bzw. der Anfangsiteration in den Fehlerabschätzungen, Fixpunktverfahren oder auch im Newton-Verfahren einsetzen. Beispielsweise gilt für das Fixpunktverfahren des Banachschen Fixpunktsatzes

$$\|y_n^{(j)} - y_n\| \leq q^j \|y_n^{(0)} - y_n\|$$

mit $q \sim hL\beta_\mu < 1$. Man sieht hier schnell den Fehler der Anfangsiteration $\|y_n^{(0)} - y_n\|$ in der Abschätzung. Jetzt steht die Frage im Raum, ob dieser Fehler minimiert werden kann? Die Idee ist, einen Anfangswert $y_n^{(0)}$ mit möglichst einfacher Formel zu erzeugen, und diesen dann iterativ bis $y_n^{(j^*)}$ zu verbessern, $j^* \geq 1$. Die einfachsten Formeln für die Startwertbestimmung sind die expliziten (Prädiktor), für die Verbesserung durch Iteration empfehlen sich die impliziten Formeln

(Korrektor). Ein Beispiel für so ein kombiniertes Verfahren wäre mit Adams-Bashforth für $m = 3$ als Prädiktor und Adams-Moulton für $m = 3$ als Korrektor – beides Verfahren 4. Ordnung. Oder man nimmt für $m = 1$ den expliziten Euler als Prädiktor und die Trapezregel als Korrektor. In der Anwendung dieser Idee gibt es drei Möglichkeiten:

(i) $j^* = 1$, d. h. einmal Prädiktor und dann einmal verbessern durch Korrektor.

(ii) $j^* \in \mathbb{N}$ fest, d. h. einmal Prädiktor und dann j^* Iterationen vom Korrektor zur Verbesserung des Wertes.

(iii) Den über den Prädiktor festgelegten Startwert $y_n^{(0)}$ so häufig mit dem Korrektor iterieren, bis $\|y_n^{(j+1)} - y_n^{(j)}\| < \text{TOL}$ gilt, d. h. die Zahl der Korrektorschritte j^* ist variabel.

Beispiele

(a) Man versucht bei impliziten Verfahren, gute Startwerte zu initialisieren, welche anschließend mit Fixpunkt-Verfahren oder Newton gelöst werden. Ein Klassiker ist hierbei, die Trapezregel mittels Fixpunktverfahren zu lösen:

$$y_n^{(k)} = y_{n-1} + \tfrac{1}{2} h_n \left[f(t_{n-1}, y_{n-1}) + f\big(t_n, y_n^{(k-1)}\big) \right] \tag{5}$$

Anstatt die Fixpunkt-Iteration vollständig durchzuführen, wird ein kombiniertes Verfahren vorgeschlagen, also Prädiktor-Korrektor. Mit dem expliziten Euler als Prädiktor soll nun (5) realisiert werden:

$$\begin{aligned} &\text{P} : y_n^* = y_{n-1} + h_n f(t_{n-1}, y_{n-1}) \\ &\text{K} : y_n = y_{n-1} + \tfrac{1}{2} h_n \left[f(t_{n-1}, y_{n-1}) + f\big(t_n, y_n^*\big) \right] \end{aligned}$$

Um es auf (ii) und (iii) anzupassen, muss lediglich $y_n^{(0)}$ durch den Prädiktor initialisiert werden, der dann als Startwert in der Iteration des Korrektors verwendet wird, d. h.:

$$\begin{aligned} &\text{P} : \quad y_n^{(0)} = y_{n-1} + h_n f(t_{n-1}, y_{n-1}) \\ &\text{K} : y_n^{(j+1)} = y_{n-1} + \tfrac{1}{2} h_n \left[f(t_{n-1}, y_{n-1}) + f\big(t_n, y_n^{(j)}\big) \right], \quad j = 0, 1, \ldots \end{aligned}$$

(b) Statt der Trapezregel soll nun Adams-Moulton für $m = 3$ mittels Fixpunktverfahren gelöst werden. Als Prädiktor dient diesmal Adams-Bashforth, ebenfalls mit $m = 3$. Gegeben seien y_{n-r} für $r = 1, \ldots, 4$. Man nutze den Prädiktor, um den Startwert $y_n^{(0)}$ zu bestimmen:

$$y_n^{(0)} = y_{n-1} + \tfrac{1}{24} h \left[55\, f_{n-1} - 59\, f_{n-2} + 37\, f_{n-3} - 9\, f_{n-4} \right]$$

Im nächsten Schritt wird der Korrektor angewendet, um durch sukzessive Approximation den Startwert $y_n^{(0)}$ an den tatsächlichen Wert y_n anzunähern:

$$y_n^{(j)} = y_{n-1} + \tfrac{1}{24}h\,\big[\underbrace{9\,f_n^{(j-1)}}_{\text{P für } j=1} + \underbrace{19\,f_{n-1} - 5\,f_{n-2} + f_{n-3}}_{\text{komplett explizit}}\big], \quad f_n^{(j-1)} := f\big(t_n,\, y_n^{(j-1)}\big)$$

Entweder wird (K) vollständig ausiteriert oder frühzeitig beendet. In der Praxis wird häufig mit festem Iterationsindex k gerechnet, sodass in wenigen Schritten die Konvergenz verbessert wird. Es stellt sich die Frage, welche Ordnung das Gesamtverfahren besitzt, wenn der Korrektor nach einer fest vorgegebenen Anzahl an Iterationen beendet wird?

Satz 1.9.2 Ordnung des PK-Verfahrens

Es sei $m^{(\mathrm{P})}$ die Ordnung vom Prädiktor. Ferner sei $m^{(\mathrm{K})}$ die Ordnung vom Korrektor. Dann ist die Gesamtordnung des PK-Verfahrens durch

$$m = \min\{m^{(\mathrm{K})}, m^{(\mathrm{P})} + j^*\} \tag{6}$$

bestimmt, wobei j^* die Anzahl der Iterationen von K bezeichnet. Im Fall $m^{(\mathrm{K})} < m^{(\mathrm{P})} + j^*$ ist die Fehlerkonstante des Gesamtverfahrens gleich der des Korrektors.

Die Ordnung des Prädiktors kann durchaus geringer sein, als die des Korrektors. Dennoch wird durch hinreichend großes j^* die Ordnung des Gesamtverfahrens aufgrund von (6) durch den Korrektor bestimmt. Um auf das erste Beispiel für PK-Verfahren zurückzukommen, der explizite Euler hat $m^{(\mathrm{P})} = 1$, die Trapezregel hat $m^{(\mathrm{P})} = 2$. Mit nur einem Korrektorschritt $j^* = 1$ erhält man somit $m = 2$.

1.10. Implizit-gestellte Differentialgleichungen

Bisher war es so, dass die Problemstellung immer explizit[15] war, d. h. $u'(t) = f(t, u)$. Die jetzt betrachtete Problemstellung wird impliziter[16] Natur, also der Form $F(t, u, u') = 0$ sein. Im Allgemeinen sind implizite Problemstellungen wesentlich schwieriger zu lösen, als explizite, weil diese grundsätzlich implizite Verfahren benötigen, um gelöst zu werden. Allerdings wird es hier auch aus mathematischer Sicht erst so richtig spannend, wobei dieser Abschnitt lediglich eine grobe erste Idee eines reichhaltigen Forschungsfeldes geben kann.

Gegeben sei also

$$F\big(t, u(t), u'(t)\big) = 0, \quad t \geq t_0, \quad u(t_0) = u_0$$

[15] Damit ist gemeint, dass die höchste Ableitung explizit aufgelöst ist.
[16] Die höchste Ableitung ist implizit in $F(\,\cdot\,)$ enthalten und nicht explizit aufgelöst.

mit einer Vektorfunktion

$$F(t, x, \eta) : D \subset \mathbb{R}^1 \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d.$$

Falls $F'_{u'}(t, u, u')$ entlang einer Lösung $u(t)$ regulär ist, dann kann zumindest prinzipiell wieder nach u' aufgelöst werden. Ein trivialer Fall ist etwa $F(t, u, u') := u' - f(t, u)$.

Beispiel (Linear-implizite Gleichung): Gegeben sei die Differentialgleichung

$$M(t, u)\, u'(t) = f(t, u), \quad M(t, u) : \mathbb{R}^1 \times \mathbb{R}^d \to \mathbb{R}^{d \times d}.$$

Hierfür gibt es unterschiedliche Fälle, die zu entsprechenden Lösungsverfahren führen.

(i) Falls $M(t, u)$ regulär ist, dann kann man wie folgt nach u' auflösen:

$$u'(t) = M^{-1}(t, u)\, f(t, u)$$

Damit können dann die üblichen Methoden angewendet werden. Etwa für $M(t, u) = 3$ kann nach $u'(t)$ gelöst werden, für $M(t, u) = t^2$ kann für $t \neq 0$ nach $u'(t)$ gelöst werden und für $M(t, u) = u(t)$ kann genau dann nach $u'(t)$ gelöst werden, wenn $u(t)$ für jedes $t \in I$ invertierbar ist, was aber für Näherungslösungen in verwendeten Verfahren nicht garantiert werden kann!

(ii) Ist $u(t) \in \mathbb{R}$, also u skalarwertig, dann gilt $M(t, u) = 1$ und somit ist $M^{-1}(t, u) = M = 1$, woraus $u'(t) = f(t, u)$ folgt.

(iii) Für $u(t) \in \mathbb{R}^2$, $u = (u_1, u_2)$ liefert

$$F(t, u, u') = \begin{pmatrix} u'_1(t) \\ u'_2(t) \end{pmatrix} - \begin{pmatrix} f_1(t, u_1) \\ f_2(t, u_2) \end{pmatrix} \quad \Rightarrow \quad F'_{u'}(t, u, u') = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Damit ist $M(t, u) = 1$, also ist M regulär.

(iv) Abschließend betrachte man nun

$$F(t, u, u') = (u')^2 + uu' - 3(u')^5.$$

Bei diesem Problem ist keine explizite Auflösung nach $u'(t)$ mehr möglich.

1.10.1. Differentiell-algebraische Gleichungen

Interessant wird nun der Fall, dass $F'_{u'}$ nicht regulär entlang einer Lösung $u(t)$ ist. Dann zerfällt nämlich $F(t, u, u')$ häufig in einen differentiellen Anteil und einen algebraischen Anteil. Der differentielle enthält dabei u' (resp. die höchste Ableitung), der algebraische hingegen nicht. Ein

solches System wird Differential Algebraic Equation (DAE) genannt. Die Grundstruktur ist durch

$$u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \quad M = \underbrace{\begin{pmatrix} M_{11} & 0 \\ 0 & 0 \end{pmatrix}}_{\text{nicht regulär}}, \qquad \begin{aligned} M_{11}u_1' &= f_1(t, u_1, u_2) \quad \text{(Differentielle Gleichung)} \\ 0 &= f_2(t, u_1, u_2) \quad \text{(Algebraische Gleichung)} \end{aligned}$$

gegeben, wobei u_1 den differentiellen und u_2 den algebraischen Anteil bezeichne.

Eine Anwendung ist bspw. die Strömungsmechanik. Dafür sind die sogenannten inkompressiblen Navier-Stokes-Gleichungen ein Standardmodell[17]. So ein Problem könnte dann wie folgt lauten: Finde Geschwindigkeit v und Druck p, sodass

$$\begin{aligned} \partial_t v + v'v - v'' + p' &= f \\ v' &= 0 \end{aligned}$$

mit

$$v = v(t, x), \quad v' := \frac{\partial}{\partial x} v(t, x).$$

In einer Dimension folgt aus $v' = 0$ auch $v'v = 0$ und somit vereinfacht sich dabei die Aufgabe zu

$$\begin{aligned} \partial_t v - v'' + p' &= f \\ v' &= 0. \end{aligned}$$

Zur Klarheit wird hier die von zuvor bekannte Notation

$$\begin{aligned} \partial_t v - \frac{\partial^2 v}{\partial x^2} + \frac{\partial p}{\partial x} &= f \\ \frac{\partial v}{\partial x} &= 0 \end{aligned}$$

verwendet. Für v liegt in der ersten Gleichung die Zeitableitung $\partial_t v$ vor, d. h. ein differentieller Anteil, während in der zweiten Gleichung keine Zeitableitung vorkommt und das einen algebraischen Anteil darstellt.

Die Frage ist, wie aus dem System die fehlende Ableitung rekonstruiert werden kann. Das motiviert die folgende

[17] Im eindimensionalen Fall machen die Navier-Stokes-Gleichungen in der Praxis keinen Sinn, reichen aber aus, um eine DAE zu erklären.

Definition 1.10.1 Index einer DAE

Der differentielle Index einer DAE ist die kleinste Zahl $k \in \mathbb{N}$, für die der Ableitungsvektor $u'(t)$ durch die $k+1$ Gleichungen

$$F(t,u,u') = 0, \quad \frac{\mathrm{d}^i}{\mathrm{d}t^i} F(t,u,u') = 0, \quad i = 1,\ldots,k$$

eindeutig in Ausdrücken von $u(t)$ bestimmt ist.

Beispiele

(a) Explizite DGL haben trivialerweise den Index 0.

(b) Gegeben sei

$$Mu'(t) = u(t) - b, \quad u = (u_1, u_2)^T$$

mit

$$M = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

Nun versucht man, die fehlende Ableitung u_1' zu rekonstruieren. Man hat

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u_1' \\ u_2' \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \Rightarrow \begin{cases} u_2' = u_1 - b_1 \to u_1 = u_2' + b_1 \\ \qquad\qquad\quad \to u_1' = u_2'' + b_1' = b_2'' + b_1' \\ 0 = u_2 - b_2 \to u_2 = b_2 \\ \qquad\qquad\quad \to u_2' = b_2' \\ \qquad\qquad\quad \to u_2'' = b_2'' \end{cases}$$

Es sind hierbei zwei Ableitungen nötig, um u_1' darstellen zu können, wodurch der Index 2 ist.

(c) Als nächstes soll ein wichtiger Spezialfall betrachtet werden, die linear-explizite Form

$$\begin{aligned} M(t,u,v)\, u' &= f(t,u,v), \quad t \geq t_0, \quad u(t_0) = u_0 \\ 0 &= g(t,u,v) \end{aligned}$$

mit Vektorfunktionen

$$f(\,\cdot\,),\ g(\,\cdot\,) : \mathbb{R}^{1+d+d} \to \mathbb{R}^d,$$

sowie einer Matrixfunktion

$$M(\,\cdot\,) : \mathbb{R}^{1+d+d} \to \mathbb{R}^{d\times d}.$$

Vorausgesetzt wird, dass $f(\,\cdot\,), g(\,\cdot\,)$ Lipschitz-stetig sind. Des Weiteren muss $M(\,\cdot\,)$ hinreichend regulär sein. Außerdem wird noch vorausgesetzt, dass $g'_v(t,u,v)$ regulär ist. Zur Bestimmung des Index muss herausgefunden werden, wie oft abgeleitet werden muss, sodass v' explizit dargestellt werden kann. Hier setzt man mit der Differentiation von $g(\,\cdot\,)$ in t an. Aus der Kettenregel folgt

$$0 = g'_t(t,u,v)\,\frac{\partial t}{\partial t} + g'_u(t,u,v)\,\frac{\partial u}{\partial t} + g'_v(t,u,v)\,\frac{\partial v}{\partial t} = g'_t(\,\cdot\,) + g'_u(\,\cdot\,)\,u' + g'_v(\,\cdot\,)\,v'.$$

Nun ist $u' = M^{-1}(\,\cdot\,)\,f(\,\cdot\,)$ und somit kann man durch Umstellen v' via

$$v' = -(g'_v)^{-1}(\,\cdot\,)\big[g'_t(\,\cdot\,) + g'_u(\,\cdot\,)\,u'\big] = -(g'_v)^{-1}(\,\cdot\,)\big[g'_t(\,\cdot\,) + g'_u(\,\cdot\,)\,M^{-1}(\,\cdot\,)\,f(\,\cdot\,)\big]$$

darstellen. Es ist also nur eine Ableitung zur Darstellung von v' nötig, also ist der Index 1.

(d) Dieses Beispiel hat strukturelle Ähnlichkeiten mit den Navier-Stokes-Gleichungen. Gegeben sei das Problem

$$\begin{aligned} u' &= f(t,u,v),\\ 0 &= g(t,u). \end{aligned}$$

Im Unterschied zu Beispiel (c) taucht hier die algebraische Variable v nicht in der algebraischen Nebenbedingung auf und entsprechend ist in den Navier-Stokes-Gleichungen der Druck p in der zweiten Gleichung nicht vorhanden. Für die Indexbestimmung wird vorbereitend $g(\,\cdot\,)$ abgeleitet:

$$0 = g'_t(\,\cdot\,) + g'_u(\,\cdot\,)\,u'$$

Das lässt sich nun mit dem differentiellen Teil kombinieren:

$$g'_u(\,\cdot\,)\,u' = g'_u(\,\cdot\,)\,f(\,\cdot\,) = -g'_t(\,\cdot\,)$$

Dies nun ein zweites Mal nach t differenziert ergibt nun durch Ketten- und Produktregel

$$\underbrace{\big[g''_{ut}(t,u) + g''_{uu}(t,u)\,u'\big]\,f(t,u,v)}_{=:\alpha_1} + g'_u(t,u)\big[f'_t(t,u,v) + f'_u(t,u,v)\,u' + f'_v(t,u,v)\,v'\big]$$

$$= \underbrace{-g''_{tt}(t,u) - g''_{tu}(t,u)\,u'}_{=:\alpha_2}.$$

Durch Auflösen nach v' erhält man

$$v' = \big(-\alpha_1 + \alpha_2 - g'_u(t,u)\, f'_t(\,\cdot\,) - \underbrace{g'_u(\,\cdot\,)\, f'_u(\,\cdot\,)\, u'}_{=:\alpha_3}\big)\big[g'_u(\,\cdot\,)\, f'_v(\,\cdot\,)\big]^{-1}.$$

In dem man nun in α_i, $i \in \{1,2,3\}$, die erste Gleichung des Problems einsetzt, so hat man eine explizite Darstellung von v' gefunden, also ist der Index gleich 2, weil $g(\,\cdot\,)$ zweimal differenziert werden musste.

1.10.2. Numerische Aspekte zum Lösen von DAEs

Man betrachte die Problemstellung

$$\begin{aligned} u' &= f(t,u,v), \quad t \in [\,t_0,\, t_0+T\,], \quad u(t_0) = u_0, \\ 0 &= g(t,u,v). \end{aligned}$$

Hierbei sei hinreichende Glattheit (Regularität) von $f(\,\cdot\,)$ und $g(\,\cdot\,)$ angenommen. Was können für numerische Schwierigkeiten auftreten?

(a) Der differentielle Teil ist explizit nach u' aufgelöst. Daher kann man diesen Teil mit expliziten oder impliziten numerischen Verfahren lösen.

(b) Der algebraische Teil ist immer implizit. Daher sind immer implizite numerische Lösungskomponenten des Gesamtsystems notwendig.

(c) Im differentiellen Teil gilt $\|f'_u(t,u,v)\| \gg 1$.

(d) Im algebraischen Teil gilt $\|g'_v(t,u,v)^{-1}\| \gg 1$ [18].

Im Weiteren soll das System mit dem Newton-Verfahren gelöst werden, obwohl Fixpunktverfahren ebenso möglich sind. Zunächst diskretisiert man den differentiellen Teil mit einem frei gewählten Zeitschrittverfahren. Bei vorausgesetzter Steifheit empfiehlt sich die Nutzung eines impliziten Verfahrens wie den impliziten Euler:

$$\frac{y_n - y_{n-1}}{h_n} = f(t, y_n, z_n)$$

Damit ist dann der algebraische Teil

$$0 = g(t, y_n, z_n).$$

[18] Weshalb Steifheit für die Inverse von g und nicht für g selbst wichtig ist, sollte aus den vorherigen Betrachtungen klar sein, denn wenn nach v' aufgelöst wird, muss mit $(g'_v)^{-1}$ durchmultipliziert werden.

Daraus formuliere man nun ein Nullstellenproblem

$$\begin{aligned} y_n - \big(y_{n-1} + h_n\, f(t, y_n, z_n)\big) &= 0, \\ g(t, y_n, z_n) &= 0. \end{aligned}$$

Dieses Nullstellenproblem kann dann mit dem Newton-Verfahren gelöst werden:

1. `Setze:`

   $(y_n^{(0)}, z_n^{(0)})$ [19]

2. `Für` $k = 1, 2, \ldots$

$$J \begin{pmatrix} \delta y \\ \delta z \end{pmatrix} = -g(y_n^{(k-1)}, z_n^{(k-1)}) \qquad \text{(Defekt-Schritt)}$$

$$\begin{pmatrix} y_n^{(k)} \\ z_n^{(k)} \end{pmatrix} = \begin{pmatrix} y_n^{(k-1)} \\ z_n^{(k-1)} \end{pmatrix} + \begin{pmatrix} \delta y \\ \delta z \end{pmatrix} \qquad \text{(Korrektur-Schritt)}$$

Hierbei ist

$$g(y_n^{(k-1)}, z_n^{(k-1)}) = \begin{pmatrix} y_n^{(k-1)} - y_{n-1} - h_n\, f(t, y_n^{(k-1)}, z_n^{(k-1)}) \\ g(t, y_n^{(k-1)}, z_n^{(k-1)}) \end{pmatrix}$$

das Residuum und

$$J := g'(y_n^{(k-1)}, z_n^{(k-1)}) = \begin{pmatrix} 1 - h_n\, f'_y(t, y_n^{(k-1)}, z_n^{(k-1)}) & -h_n\, f'_z(t, y_n^{(k-1)}, z_n^{(k-1)}) \\ g'_y(t, y_n^{(k-1)}, z_n^{(k-1)}) & g'_z(t, y_n^{(k-1)}, z_n^{(k-1)}) \end{pmatrix}$$

die Jacobi-Matrix. Hierbei wurde nun implizit vorausgesetzt, dass $y_n, z_n \in \mathbb{R}$ sind, also die Differentialgleichung skalarwertig ist. Formal kann man für DGL-Systeme identisch vorgehen.

Bemerkung: Wenn J positiv definit ist, wird das Newton-Verfahren problemlos funktionieren. Allerdings kann J (leider) auch leicht indefinit werden. Hier sind dann spezielle Tricks notwendig.

1.10.3. Voll-implizite Problemstellungen und deren numerische Lösung

Bisher wurden hier Aufgabenstellungen vorgestellt, bei denen die höchste Ableitung explizit aufgelöst werden konnte. Bei den DAE konnte zumindest noch ein Teil des System explizit nach der höchsten Ableitung aufgelöst werden. Jetzt kommen mit den voll impliziten Aufgaben Systeme der Form

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

mit vektorwertigen u, u'. Das Schema lautet dann:

[19] Die Initialisierung des Newton-Startwerts kann z. B. durch $y_n^{(0)} := y_{n-1}$ und $z_n^{(0)} := z_{n-1}$ erfolgen.

1. Führe diskrete Variable y ein:

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

2. Definiere

$$g(y_n) := F(t, y_n, y_{n-1}, y'_n, y'_{n-1})$$

3. Setze Nullstellenproblem

$$g(y_n) = 0$$

und löse mit Fixpunktverfahren oder Newtonverfahren.

Beispiele

(a) Gegeben sei

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

mit z. B. $f(t, u) = t^2 u^2$. Man löse hier

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

d. h. $u' - f(t, u) = 0$. Mit dem impliziten Euler erhält man

$$\frac{y_n - y_{n-1}}{h_n} - f(t_n, y_n) = 0.$$

Diese Gleichung kann wie üblich mit Fixpunkt- oder Newton-Verfahren gelöst werden.

(b) Gegeben sei

$$F(t, u, u') = (u')^3 + (u\,u')^2 - f(t, u).$$

Hier ist kein Auflösen nach u' möglich. Auch wird hier gar nicht erst versucht, irgendwelche Vereinfachungen vorzunehmen. Wie geht man an so ein Problem heran? Für gewöhnlich diskretisiert man alle u' mit Differenzenquotienten, woraus man die diskrete Differentialgleichung

$$g(y_n) := \left(\frac{y_n - y_{n-1}}{h_n}\right)^3 + \left(y_n \cdot \frac{y_n - y_{n-1}}{h_n}\right)^2 - f(t_n, y_n) = 0$$

erhält. Als nächstes formuliert man daraus ein Fixpunkt- oder das Newton-Verfahren

1. `Setze:`

   $y_n^{(0)}$

2. `Für` $k = 1, 2, \ldots$

   $g'(y_n^{(k-1)})\delta y = -g(y_n^{(k-1)})$

   $y_n^{(k)} = y_n^{(k-1)} + \delta y$

mit

$$g'(y_n) = 3\left(\frac{y_n - y_{n-1}}{h_n}\right)^2 \frac{1}{h_n} + 2\Big(y_n \cdot \frac{y_n - y_{n-1}}{h_n}\Big)\Big(\frac{y_n - y_{n-1}}{h_n} + \frac{y_n}{h_n}\Big) - f'_y(t_n, y_n).$$

Beispielsweise wäre $f'_y(t, y_n) = 2t^2 y_n$ für $f(t, y_n) = t^2 y_n^2$.

(c) Gegeben sei

$$F(t, u, u') = (u')^2 + u\,u' - 3(u')^5 = 0.$$

Im vorherigen Beispiel wurde $u \to y_n$ und $u' \to \frac{y_n - y_{n-1}}{h_n}$ genutzt. Hierzu gibt es aber auch Alternativen. Statt $u \to y_n$ ist auch $u \to y_{n-1}$ möglich und für den Differenzenquotienten könnte man explizitieren, d. h. $u' \to \frac{y_{n-1} - y_{n-2}}{h_n}$. Hier soll nur $u \to y_{n-1}$ verändert werden, woraus sich

$$g(y_n) := \left(\frac{y_n - y_{n-1}}{h_n}\right)^2 + y_{n-1} \cdot \frac{y_n - y_{n-1}}{h_n} - 3\left(\frac{y_n - y_{n-1}}{h_n}\right)^5 = 0$$

ergibt. Das resultiert dann für $k = 1, 2, \ldots$ in dem Newton-Verfahren

$$g'(y_n^{(k-1)})\delta y = -g(y_n^{(k)}), \quad y_n^{(k)} = y_n^{(k-1)} + \delta y,$$

worin

$$g'(y_n) = 2\left(\frac{y_n - y_{n-1}}{h_n}\right)\frac{1}{h_n} + \frac{y_{n-1}}{h_n} - 15\left(\frac{y_n - y_{n-1}}{h_n}\right)^4 \frac{1}{h_n}.$$

1.11. Galerkin-Verfahren

Bisher wurden nahezu ausschließlich "Differenzenverfahren" verwendet, dessen Ziel es war, Ableitungen mit Hilfe von Differenzenquotienten zu approximieren. Daraus resultierten Verfahren niedriger und höherer Ordnungen und teilweise A-stabile implizite Verfahren. Die Schrittweiten-

steuerungen und die Adaptivität basierten dabei auf a priori-Fehlerabschätzungen, die Informationen über den Hauptabschneidefehler lieferten und Abschätzungen der Lipschitz-Konstanten ermöglichten. Jetzt sollen die Galerkin-Verfahren vorgestellt werden. Diese basieren auf der integralen Darstellung der Differentialgleichung und auf passend gewählten Funktionenräumen. Dafür gibt es einige Vorteile:

(i) Konstruktion von höherer Ordnungs-impliziter Verfahren

(ii) Generisch gute numerische Stabilitätseigenschaften

(iii) Das Verfahren erbt die Monotonieeigenschaften der AWA

(iv) Schrittweitensteuerung und Adaptivität erfolgen aufgrund von a posteriori-Fehlerinformationen und erlaubt somit die Einbeziehung der bereits gemachten Fehler während der Rechnung

(v) 'Globale' Sichtweise auf die Problemstellung

1.11.1. Variationelle Formulierung der AWAs

Gegeben ist die wohl bekannte Problemstellung

$$u'(t) = f(t,u), \quad u(t_0) = u_0. \tag{1}$$

Bisher nahm man die Diskretisierung mit dem Differenzenquotienten vor, was eine Lösung y_n in diskreten Zeitpunkten $t_0, t_1, \ldots, t_N$ lieferte. Nun sei wie üblich $I = [\, t_0,\, t_0 + T\,]$. Sei ferner $u \in C^1(I)^d$. Aus (1) folgt

$$\begin{aligned} & u'(t) - f(t,u) = 0 && \text{(Starke Form)} \\ \Rightarrow \quad & \int_I \big(u'(t) - f(t,u)\big)\varphi(t)\,\mathrm{d}t = 0, \quad \forall \varphi \in C^1(I) && \text{(Schwache Form)} \end{aligned}$$

in einer Dimension. Im Allgemeinen ist der Integrand durch

$$\langle u' - f(t,u), \varphi \rangle$$

gegeben, wobei $\langle \cdot, \cdot \rangle$ das euklidische Skalarprodukt (siehe S. 89) bezeichnet. Beispielsweise gilt bei $d = 2$

$$u' = f(t,u) \quad \Leftrightarrow \quad \begin{pmatrix} u_1' \\ u_2' \end{pmatrix} = \begin{pmatrix} f_1(t,u_1) \\ f_2(t,u_2) \end{pmatrix}$$

und somit

$$\begin{aligned}\int \langle u' - f(t,u), \varphi \rangle \,\mathrm{d}t &= \int \begin{pmatrix} u_1' - f_1(t,u_1) \\ u_2' - f_2(t,u_2) \end{pmatrix} \cdot \begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix} \mathrm{d}t \\ &= \int \Big(\big[(u_1' - f_1(t,u_1)) \cdot \varphi_1 \big] + \big[(u_2' - f_2(t,u_2)) \cdot \varphi_2 \big] \Big) \,\mathrm{d}t.\end{aligned}$$

Bei der variationellen Form der DGL darf also beliebig mit den sogenannten Testfunktionen $\varphi \in C^1$ variiert werden.

Bemerkung: Die Rückrichtung, also die Erzeugung der starken Form aus der schwachen, ist etwas delikater. Dafür wähle man die Dirac-Funktion mit entsprechendem Träger. Stichwort ist hier das Fundamentallemma der Variationsrechnung. Dieses kann vom interessierten Leser in (Ciarlet 2013) nachgeschlagen werden.

1.11.2. Diskretisierung der schwachen Form

Zunächst sei wie üblich

$$t_0 < t_1 < \ldots < t_n < \ldots < t_N = t_0 + T.$$

Insbesondere wird nun das Gesamtzeitintervall I in links-halboffene Teilintervalle $I_n := (\, t_{n-1},\, t_n \,]$ unterteilt. Man setze

$$h_n := t_n - t_{n-1}, \quad h := \max_n \, h_n.$$

Ziel des Galerkin-Verfahrens ist es, auf jedem der Intervalle I_n mit "einfachen" (hier polynomialen) Funktionen zu arbeiten. D. h. man berechnet

$$\int_{I_n} \big(u'(t) - f(t,u) \big) \, \varphi(t) \,\mathrm{d}t = 0,$$

wobei φ Testfunktionen aus einem geeigneten Funktionenraum sind. Abbildung 1.16 zeigt, wie auf einigen Intervallen konstante bzw. lineare Testfunktionen benutzt werden, mit denen die Integraldarstellung intervallweise gelöst wird. Im Vergleich dazu stehen in blau markiert die Annäherungen y_n, die mit bisher studierten Einschritt- und Mehrschrittverfahren berechnet werden können. Während also die bereits bekannten Methoden stets punktweise Lösungen in diskreten Zeitpunkten t_n ermitteln, ist der Sinn hinter den Galerkin-Verfahren, auf jedem Teilintervall durch Variationen eine Bestapproximation im stetigen Funktionenraum zu finden. Das führt nun zu den Fragen, welche Ansatzfunktionen für u und was für Testfunktionen φ gut sind, um numerisch eine möglichst präzise Lösung zu bestimmen? Dieser Grundgedanke ist die Motivation hinter den sogenannten FEM. Wir besprechen diese ausführlich in Numerik III (T. Wick 2022).

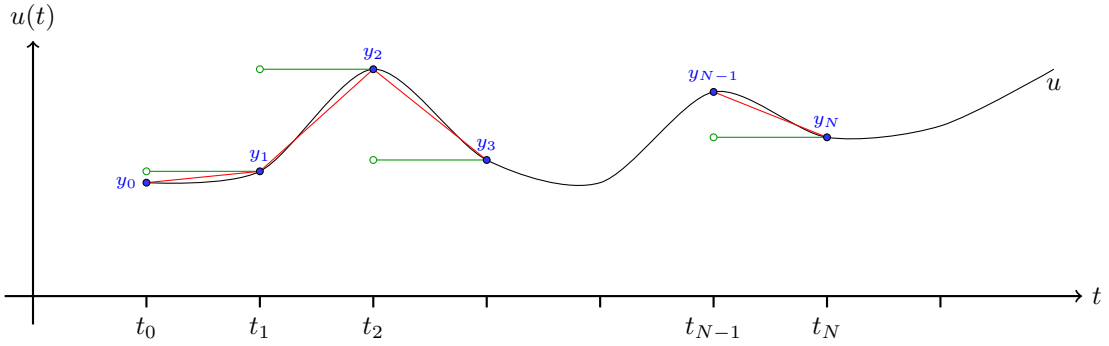


Abbildung 1.16.: Diskretisierung einer variationellen Differentialgleichung mit konstanten und linearen Testfunktionen.

Definition 1.11.1

Finite-Elemente-Methoden (FEM) Eine FEM besteht aus der Festlegung der folgenden Eigenschaften:

(1) Geometrie

(2) Ansatzfunktionen (z.B. Polynome; also 'einfache' Funktionen)

(3) DoFs (Degrees of Freedom, Freiheitsgrade), um Ansatzfunktionen eindeutig festzulegen

Das klingt noch sehr abstrakt. Für unsere Diskretisierung wurde ein Intervall in Teilintervalle zerlegt, das wäre hier die Geometrie der FEM. Bei zweidimensionalen Aufgaben werden häufig Dreiecke oder Vierecke verwendet, in drei Dimensionen nutzt man Tetraeder und Parallelepipede. Die Ansatzfunktionen φ sind in der Regel Polynome. In Abbildung 1.16 wurden in etwa Polynome vom Grad 0 und 1 verwendet. Weil man sich für eindeutige Lösungen interessiert, sind Rahmenbedingungen notwendig. Polynome r-ten Grades haben insgesamt $r+1$ Freiheitsgrade. Darum sind auch entsprechend viele Rahmenbedingungen notwendig, um Eindeutigkeit garantieren zu können. Häufig nutzt man hier Werte in Punkten aus einem Element der Geometrie, aber Integrale über die Geometrie (oder Teile der Geometrie) sind ebenfalls Möglichkeiten, Rahmenbedingungen festzulegen.

1.11.3. Konkrete Konstruktion der Funktionenräume

Man formuliere eine Unterteilung

$$\mathbb{T}_h = \{\, I_n = (t_{n-1}, t_n) \,|\, n = 1, \ldots, N \,\}.$$

$\mathbb{T}_h$ wird als Gitter bezeichnet und ist für die FEM, auf die hier hingearbeitet wird, die Geometrie. Für den Funktionenraum definiere man den Raum der stückweise glatten Funktionen:

$$V(I) := \left\{ v : I \to \mathbb{R}^d \;\middle|\; v(t_0) \in \mathbb{R}^d,\ v|_{I_n} \in C_c^1(I_n)^d,\ n = 1, \ldots, N \right\}$$

Hierbei bezeichnet $C_c^1(I_n)$ den Raum der auf den I_n stetig-differenzierbaren und stetig zum linken Randpunkt t_{n-1} fortsetzbaren Funktionen[20]. Nun wird die variationelle Form der DGL mit Hilfe des Raumes $V(I)$ weiter konkretisiert. Bisher war diese Form noch immer global bzgl. I formuliert. Nun soll eine Form bzgl. der I_n für $n = 1, \ldots, N$ betrachtet werden. Bei der reinen intervallweisen Betrachtung wird bei der entsprechenden Realisierung

$$\int_{I_n} \big(u'(t) - f(t,u)\big)\, \varphi(t)\, \mathrm{d}t = 0, \quad \forall \varphi \in V(I)$$

keine Information von I_n auf I_{n+1} übertragen. Daher muss diese Formulierung entsprechend erweitert werden. Für eine Funktion $v \in V(I)$ definiere man dazu

$$v_n^+ = \lim_{t \searrow t_n} v(t), \quad v_n^- = \lim_{t \nearrow t_n} v(t), \quad [v]_n = v_n^+ - v_n^-.$$

Man nennt $[v]_n$ den Sprung von v. Trivialerweise ist $[v]_n \equiv 0$, sofern $v \in C(I)$.

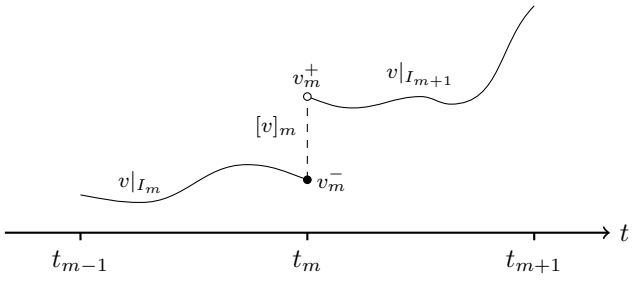


Abbildung 1.17.: Beispiel zum Sprung von v

Damit kann nun die variationelle Form der DGL in der folgenden äquivalenten Form geschrieben werden:

[20] Stetig fortsetzbar heißt nicht, dass die Funktion global stetig ist. Tatsächlich kann die Funktion global unstetig sein, ist aber auf den einzelnen Teilintervallen stetig.

Problemformulierung Unstetiges Galerkin-Verfahren für AWAs

Finde $u \in V(I)$ mit den Eigenschaften $u(t_0) = u_0^- = u_0$, sowie

$$\sum_{n=1}^{N} \left[\int_{I_n} \langle u'(t) - f(t,u), \varphi(t) \rangle \, \mathrm{d}t + \underbrace{\langle [u]_{n-1}, \varphi_{n-1}^+ \rangle}_{\text{Kopplungsterm}} \right] = 0 \tag{2}$$

für alle $\varphi \in V(I)$ mit dem euklidischen Skalarprodukt

$$\langle x, y \rangle := \sum_{k=1}^{d} x_k y_k$$

für die Fälle $d > 1$. Für $d = 1$ gilt direkt

$$\langle u'(t) - f(t,u), \varphi(t) \rangle = [u'(t) - f(t,u)] \, \varphi(t).$$

Bemerkung Es gibt auch stetige Galerkin-Verfahren, die in Partiellen Differentialgleichungen behandelt werden. Das unstetige Galerkin-Verfahren ist auch also dG-Verfahren bekannt, worin das dG für "discontinuous Galerkin" steht. Das Galerkin-Verfahren liefert eine globalere Sichtweise auf die zugrundeliegende Problemstellung. Im Prinzip ist eine simultane (parallele) Lösung aller I_n möglich, anstatt sequentiell von $(t_0, y_0) \to (t_1, y_1) \to \ldots \to (t_N, y_N)$ zu gehen.

In der bisherigen Problemstellung war der Anfangswert noch explizit gegeben:

$$u(t_0) = u_0 = u_0^-$$

Dabei ist u_0 der exakte Wert, u_0^- spielt später die Rolle des diskreten Anfangswertes. In starker Form gilt

$$u_0^- - u_0 = 0.$$

Dann folgt wie vorher die schwache Form

$$\langle u_0^- - u_0, \, \varphi_0^- \rangle = 0.$$

Dann wird (2) zu

$$\sum_{n=1}^{N} \left[\int_{I_n} \langle u'(t) - f(t,u), \varphi(t) \rangle \, \mathrm{d}t + \langle [u]_{n-1}, \varphi_{n-1}^+ \rangle \right] + \langle u_0^- - u_0, \, \varphi_0^- \rangle = 0. \tag{3}$$

Da u_0 durch die Problemstellung bekannt ist, wird dieser Teil auf die rechte Seite geschrieben[21].

[21] Wie üblich schreibt man Unbekanntes links und Bekanntes rechts.

Damit erhält man aus (3)

$$\sum_{n=1}^{N}\Big[\int_{I_n}\langle u'(t)-f(t,u),\varphi(t)\rangle\,\mathrm{d}t+\langle[u]_{n-1},\varphi_{n-1}^{+}\rangle\Big]+\langle u_0^{-},\,\varphi_0^{-}\rangle=\langle u_0,\,\varphi_0^{-}\rangle. \tag{4}$$

Man definiert nun

$$A(u)(\varphi):=\sum_{n=1}^{N}\Big[\int_{I_n}\langle u'(t)-f(t,u),\varphi(t)\rangle\,\mathrm{d}t+\langle[u]_{n-1},\varphi_{n-1}^{+}\rangle\Big]+\langle u_0^{-},\,\varphi_0^{-}\rangle.$$

Dann kann die vorherige Problemstellung in

$$A(u)(\varphi)=\langle u_0,\varphi_0^{-}\rangle \tag{5}$$

umformuliert werden. Man nennt $A(u)(\varphi)$ eine Semi-Linearform, welche bzgl. des ersten Arguments, hier u, nicht-linear und bzgl. des zweiten Arguments, hier φ, linear ist. Ist u selbst linear, so ist (5) ein lineares Gleichungssystem, andernfalls ist das Gleichungssystem wie auch u nicht-linear.

In anderen Worten ist das Galerkin-Verfahren eine Approximation des (immer noch) unendlich-dimensionalen Raumes $V(I)$ durch eine Folge endlich-dimensionaler Teilräume $S_h^{(r)}$, sodass für $h\to 0$ die unten genannte endlich-dimensionale Problemstellung gegen (5) konvergiert. In $S_h^{(r)}$ bezeichnet das r den Polynomgrad der Basisfunktion der I_n und das h den Diskretisierungsparameter, hier also die Schrittweite. Konkret werden die $S_h^{(r)}$ wie folgt definiert:

$$S_h^{(r)}(I)=\big\{\,\varphi\in V(I)\;\big|\;\varphi(t_0)\in\mathbb{R}^d,\;\varphi|_{I_n}\in P_r(I_n)^d,\;n=1,\dots,N\,\big\}$$

Bemerkung: Die Ansatzfunktionen $\varphi\in S_h^{(r)}(I)$ können im Allgemeinen unstetig sein und im Anfangszeitpunkt t_0 den Wert $\varphi(t_0)\neq\lim\limits_{t\searrow t_0}\varphi(t)$ annehmen.

Man erhält daraus die

Problemformulierung Galerkin-Verfahren (diskrete Version)

Finde $U\in S_h^{(r)}(I)$, sodass

$$A(U)(\varphi)=\langle u_0,\,\varphi_0^{-}\rangle \tag{6}$$

für alle $\varphi\in S_h^{(r)}(I)$ erfüllt ist.

Bemerkung (Notation): Bisher wurde die diskrete Funktion y genannt. Bei der Galerkin-Methode heißt die Galerkin diskrete Funktion U.

Das oben genannte Verfahren wird im Allgemeinen dG(r)-Verfahren genannt. Die abstrakte Form (6) kann bei geschickter Wahl von $S_h^{(r)}(I)$ wieder als konventionelles Zeitschrittverfahren geschrieben werden. Der Trick ist hierbei, die Basisfunktionen aus $S_h^{(r)}(I)$ mit möglichst kleinem, lokalen Träger zu wählen. Die einfachste Wahl ist

$$\varphi = \begin{cases} 1 & \text{auf } I_n \\ 0 & \text{auf } I_n \neq I_m,\ n \neq m \end{cases} \tag{7}$$

die weiter unten (Abschnitt 1.11.5) auf verschiedene Weisen konkretisiert werden kann.

Zunächst folgt aus (4) und der Wahl aus $S_h^{(r)}(I)$

$$\sum_{n=1}^{N} \left[\int_{I_n} \langle U'(t) - f(t,U), \varphi(t) \rangle \,\mathrm{d}t + \langle [U]_{n-1}, \varphi_{n-1}^+ \rangle \right] + \langle U_0^-,\, \varphi_0^- \rangle = \langle u_0,\, \varphi_0^- \rangle.$$

Insbesondere kann entkoppelt werden:

$$\int_{I_n} \langle U'(t) - f(t,U), \varphi(t) \rangle \,\mathrm{d}t + \langle [U]_{n-1}, \varphi_{n-1}^+ \rangle + \langle U_0^-,\, \varphi_0^- \rangle = \langle u_0,\, \varphi_0^- \rangle, \quad n = 1, \ldots, N \tag{8}$$

Zunächst können die Anfangswerte eliminiert werden für $\varphi_0^- \equiv 1$:

$$\langle U_0^-,\, \varphi_0^- \rangle = \langle u_0,\, \varphi_0^- \rangle \quad \Rightarrow \quad U_0^- = u_0.$$

Danach folgt aus (8)

$$\int_{I_n} \langle U'(t) - f(t,U), \varphi(t) \rangle \,\mathrm{d}t + \langle [U]_{n-1},\, \varphi_{n-1}^+ \rangle = 0 \tag{9}$$

$$\Leftrightarrow \quad \int_{I_n} \langle U'(t) - f(t,U), \varphi(t) \rangle \,\mathrm{d}t + \langle U_{n-1}^+ - U_{n-1}^-,\, \varphi_{n-1}^+ \rangle = 0$$

und dadurch, weil $\langle \cdot, \cdot \rangle$ eine Bilinearform ist,

$$\Rightarrow \quad \int_{I_n} \langle U'(t), \varphi(t) \rangle \,\mathrm{d}t + \langle U_{n-1}^+,\, \varphi_{n-1}^+ \rangle = \int_{I_n} \langle f(t,U), \varphi(t) \rangle \,\mathrm{d}t + \langle U_{n-1}^-,\, \varphi_{n-1}^+ \rangle \tag{10}$$

für $n = 1, \ldots, N$.

Bemerkung: Trotz, dass die Startwerte in (9) weggefallen sind, werden diese dennoch in (10) berücksichtigt. Für $n = 1$ arbeitet man nämlich mit $\langle U_0^+,\, \varphi_0^+ \rangle$ sowie $\langle U_0^-,\, \varphi_0^+ \rangle$. Der "echte" Startwert ist durch U_0^- repräsentiert.

Beispiel (Modellproblem):

Gegeben sei $u' = \lambda u = f(t,u)$ (kontinuierliches Problem). Diskret mit der Galerkin-Notation erhält man dann $U' = f(t,U)$ mit $f(t,U) = \lambda U$. Damit wird (10) konkret zu

$$\int_{I_n} U' \cdot \varphi \,\mathrm{d}t + U_{n-1}^{+} \cdot \varphi_{n-1}^{+} = \int_{I_n} \lambda U \cdot \varphi \,\mathrm{d}t + U_{n-1}^{-} \cdot \varphi_{n-1}^{+}.$$

1.11.4. Galerkin-Orthogonalität (Satz des Pythagoras)

Die Galerkin-Orthogonalität ist äquivalent zur Bestapproximationseigenschaft, in Worten: Bei der Nutzung des Galerkin-Verfahrens zur Lösung von DGLn ist der Fehler zwischen kontinuierlicher Lösung $u \in V(I)$ und $U \in S_h^{(r)}(I)$, gemessen in den jeweiligen Semi-Linearformen, minimal. Konkret steht dieser Fehler senkrecht zum diskreten Funktionenraum $S_h^{(r)}(I)$. Als mathematische Relation betrachtet hat man zwei Probleme:

(i) Finde $u \in V(I) : A(u)(\varphi) = \langle u_0, \varphi_0^- \rangle \quad \forall \varphi \in V(I)$ (kontinuierlich)

(ii) Finde $U \in S_h^{(r)}(I) : A(U)(\varphi) = \langle U_0, \varphi_0^- \rangle \quad \forall \varphi \in S_h^{(r)}(I)$ (diskret)

Nun gilt bei spezieller Wahl von $\varphi \in S_h^{(r)}$ in (i):

$$\text{Finde } u \in V(I) : A(u)(\varphi) = \langle u_0, \phi_0^- \rangle \quad \forall \varphi \in S_h^{(r)}.$$

Daraus erhält man

$$A(u)(\varphi) - \langle u_0, \varphi_0^- \rangle - A(U)(\varphi) + \langle U_0, \varphi_0^- \rangle = 0 \quad \forall \phi \in S_h^{(r)},$$

d. h.

$$A(u)(\varphi) - A(U)(\varphi) = 0.$$

Hierbei wurde ausgenutzt, dass $\langle u_0, \varphi_0^- \rangle + \langle U_0, \varphi_0^- \rangle = \langle U_0 - u_0, \varphi_0^- \rangle$ gilt, was für $U_0 = u_0$ identisch Null ist. Dies ist eine Orthogonalitätsbeziehung im Funktionenraum $S_h^{(r)}$. Oftmals kann $A(\,\cdot\,)(\varphi)$ als Skalarprodukt identifiziert werden. Wenn ein Skalarprodukt gleich Null ist, so herrscht Orthogonalität.

Für die geometrische Interpretation seien $u \in V(I), U \in S_h^{(r)}(I)$ gegeben. Es sei $S_h^{(r)}(I) \subset V(I)$ sowie

$$A(u)(\varphi) - A(U)(\varphi) = 0 \quad \forall \varphi \in S_h^{(r)}(I).$$

1.11.5. Konstruktion konkreter Verfahren

Ausgehend von (10) sollen nun konkrete Ansatz- und Testfunktionen festgelegt werden, die die Eigenschaft (7) besitzen. Genutzt wird hier $d = 1$, aber analog kann das auch in höheren Dimen-

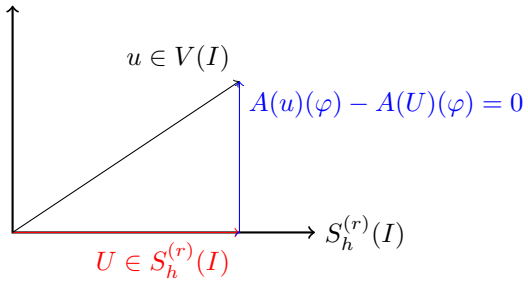


Abbildung 1.18.: Geometrische Interpretation der Galerkin-Orthogonalität: Bestapproximation, wo der Fehler gemessen in der Bilinearform senkrecht auf dem kontinuierlichen Funktionenraum steht.

sionen umgesetzt werden. Betrachtet wird also

$$\int_{I_n} U'(t) \cdot \varphi(t)\,\mathrm{d}t + U_{n-1}^+ \cdot \varphi_{n-1}^+ = \int_{I_n} f(t,U) \cdot \varphi(t)\,\mathrm{d}t + U_{n-1}^- \cdot \varphi_{n-1}^+. \tag{11}$$

Fall $r = 0$: Für diesen Fall kann $U_n := U_n^-$ auf I_n gesetzt werden, sprich man wählt eine konstante Fortsetzung von U_n^-. O. b. d. A. ist $\varphi \equiv 1$ auf I_n und 0 sonst. Weil φ konstant ist, könnte die Konstante ohnehin beidseitig dividiert werden, darum folgt aus (11):

$$\begin{aligned}
&\int_{I_n} U'(t)\,\mathrm{d}t + U_{n-1}^+ = \int_{I_n} f(t,U)\,\mathrm{d}t + U_{n-1}^- \\
\Leftrightarrow\quad & U_n^- - U_{n-1}^+ + U_{n-1}^+ = \int_{I_n} f(t,U)\,\mathrm{d}t + U_{n-1}^- \\
\Leftrightarrow\quad & U_n^- - U_{n-1}^- = \int_{I_n} f(t,U)\,\mathrm{d}t
\end{aligned}$$

Hier wird jetzt das Integral über numerische Quadratur gelöst (siehe (Richter und Thomas Wick 2017)). Weil die Testfunktionen konstant sind, reicht hier eine Quadraturformel niedrigerer Ordnung (sprich eine QF, die konstante Polynome exakt integriert), etwa die rechtsseitige Boxregel:

$$\begin{aligned}
\Rightarrow\quad & U_n^- - U_{n-1}^- = h_n\, f(t_n, U_n^-) \\
\overset{U_n = U_n^-}{\Leftrightarrow}\quad & U_n - U_{n-1} = h_n\, f(t_n, U_n)
\end{aligned}$$

Das dG(0)-Verfahren resultiert also im impliziten Euler-Verfahren.

Fall $r = 1$: Für dG(1) hat man zwei DoF's (Freiheitsgrade) auf I_n. Diese DoF's sind hier U_{n-1}^+ und U_n^-. Das Ziel ist die Konstruktion einer linearen Funktion auf I_n. Man verwendet dafür den Ansatz, für $U(t)$ auf I_n mit der Lagrangschen Darstellung eine Polynom-Interpolation durchzuführen:

$$U(t) = h_n^{-1}(t - t_{n-1})\, U_n^- - h_n^{-1}(t - t_n)\, U_{n-1}^+$$

Insbesondere ist dann

$$U(t_n) = h_n^{-1}(t_n - t_{n-1})\, U_n^- = \frac{h_n}{h_n}\, U_n^- = U_n^-.$$

Nun wird (11) jeweils mit den Basispolynomen (Newtonsche Darstellung der Lagrange-Interpolation (Richter und Thomas Wick 2017))

$$\begin{aligned} &\varphi_1 \equiv 1 && \text{(1. Basisfunktion)} \\ &\varphi_2 = h_n^{-1}(t - t_{n-1}) && \text{(2. Basisfunktion)} \end{aligned}$$

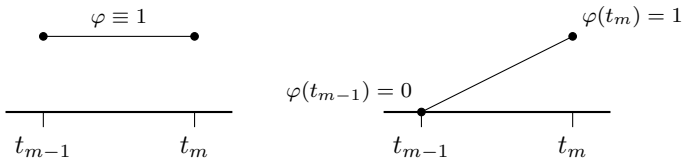


Abbildung 1.19.: Gestalt der Basispolynome.

getestet. So erhält man:

$$\varphi_1 : \quad U_n^- - U_{n-1}^- = \int_{I_n} f(t,u)\,\mathrm{d}t$$

$$\varphi_2 : \quad U_n^- - U_{n-1}^+ = 2h_n^{-1} \cdot \int_{I_n} f(t,U)\,(t - t_{n-1})\,\mathrm{d}t$$

Da hier mit $r = 1$ gearbeitet wird, müssen zur Erhaltung einer optimalen Konvergenzordnung des Galerkin-Verfahrens die Integrale der rechten Seite mit einer höherwertigen Quadraturformel approximiert werden, z. B. mit der Trapezregel, da nun lineare Basisfunktionen zugrunde liegen und daher die Integrale mit einer linearen Quadraturformel exakt integriert werden sollten. Es gibt also einen Zusammenhang zwischen der Ordnung der polynomialen Ansatzfunktion auf I_n und der Ordnung der numerischen Quadratur. Dann folgt

$$\varphi_1 : \quad U_n^- - U_{n-1}^- = \frac{h_n}{2}\left[f(t_{n-1}, U_{n-1}^+) + f(t_n, U_n^-)\right]$$

$$\varphi_2: \quad U_n^- - U_{n-1}^+ = h_n\, f(t_n, U_n^-)$$

Das ist ein System von zwei Gleichungen mit den zwei Unbekannten U_{n-1}^+ und U_n^-. Man erhält eine Lösung durch das Lösen des nicht-linearen Gleichungssystems (wiederum mit Newton oder Fixpunkt) oder indem man die zweite Gleichung in die erste einsetzt. Daraus erhält man den impliziten Runge-Kutta:

$$U_n^- - U_{n-1}^- = \frac{1}{2} h_n \Big[f\big(t_{n-1}, U_n^- - h_n\, f(t_n, U_n^-)\big) + f(t_n, U_n^-) \Big]$$

1.11.6. A $dG(r)$ implementation in deal.II

In this section, we consider again the model problem

$$\begin{aligned} u'(t) &= \lambda u(t) \quad \text{in } I := (0, T), \\ u(0) &= 1, \end{aligned}$$

where $\lambda = 5$ and $T = 1$. Using the $dG(r)$ from the previous sections, an implementation in the modern finite element package deal.II, version 9.3.0, Arndt, Wolfgang Bangerth, Blais u. a. 2021; Arndt, Wolfgang Bangerth, Davydov u. a. 2021; W. Bangerth, Hartmann und Kanschat 2007 has been done by Jan Philipp Thiele and Julian Roth and is published on github[22]. A graphical comparison of $dG(r)$ with $r = 0, 1, 2$ is displayed in Figure 1.20.

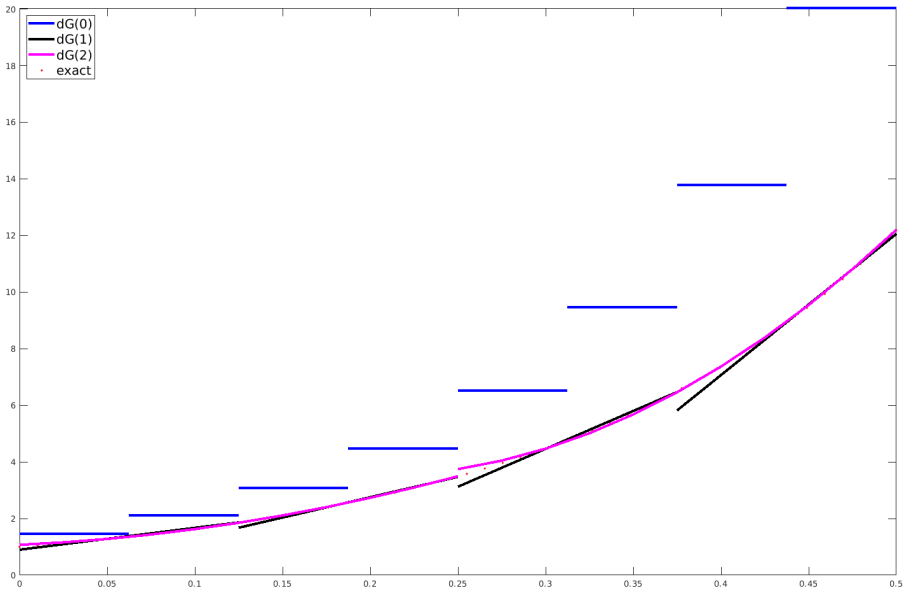


Abbildung 1.20.: Comparison of $dG(r)$ with $r = 0, 1, 2$ for the ODE model problem.

[22] https://github.com/jpthiele/ode_dg

1.11.7. Computational convergence analysis for the end time error

For the computational convergence analysis we follow our steps outlined before and compute discrete solutions u_k on at least three different (temporal) meshes. Moreover, we test different polynomial orders r and confirm these findings from the literature in which superconvergence is observed. As quantity of interest, we adopt the end time solution value $u_k(T = 1)$.

Computationally-obtained convergence order

In order to calculate the convergence order α we follow Abschnitt 1.8.2. Let P be some numerical process with $P(k) \to P$ for $k \to 0$, where k is the discretization parameter, i.e., the time step size $k = t_m - t_{m-1}$. Moreover, $\tilde{P}$ is either the exact limit P (in case it is known) or some 'good' approximation to it. We assume convergence and

$$P(k) - \tilde{P} = O(k^\alpha),$$

with the convergence order $\alpha > 0$. Let us assume that three numerical solutions are known (this is the minimum number if the limit P is not known). That is

$$P(k), \quad P(k/2), \quad P(k/4).$$

A specific example for $P(k)$ is the end time value $u_k(T)$. Then as before:

1. $P(k) := u_k(T)$, which means that $u_k(T)$ is evaluated at T using in the computation time step sizes k;
2. $P(k/2) := u_{k/2}(T)$, which means that $u_{k/2}(T)$ is evaluated at T using in the computation time step sizes $k/2$;
3. $P(k/4) := u_{k/4}(T)$, which means that $u_{k/4}(T)$ is evaluated at T using in the computation time step sizes $k/4$.

Further smaller time step sizes are obtained by continuing bisection. Of course, the more data we have, the more precise the result becomes. Thus, $P(k/8), P(k/16), P(k/32)$ can be utilized as well. Then, the convergence order can be calculated via the formal approach $P(k) - \tilde{P} = ck^\alpha$ that was introduced before.

Remark 1.11.2. *The superconvergence of higher order $dG(r)$ schemes at discrete time points t_m holds for any $r \geq 1$* **Ra17_1***; Satz 7.4 and Bemerkung 7.1. It holds for the error in the discrete time points:*

$$\max_{1 \leq m \leq M} \|u(t_m) - u_k(t_m)\| = O(k_m^{2r+1}),$$

yielding order $\alpha = 1$ for $dG(0)$, order $\alpha = 3$ for $dG(1)$, order $\alpha = 5$ for $dG(2)$, and so forth. This is computationally confirmed in the following subsubsection.

Computational convergence analysis for Section 1.11.6

We take the code from Section 1.11.6, and modify two parts:

- The number of mesh elements is too small (i.e., time step size k too large) for our purposes, and therefore, we increase the number of mesh elements;
- For the computational convergence analysis, the values u_k (etc.) must be of sufficient accuracy in their output, for which we add
  `std::setprecision(16) << std::scientific <<`
  in the
  `output_results()` function.

In the code, the number r is changed in the `int main()`
function with `int fe_degree`. The number of mesh elements M is changed in the `void ODE::make_grid()`
function with `triangulation.refine_global(...);`.
We run nine computations that are summarized in Table 1.1. On these results, we now apply

r	Mesh level	M (mesh elem.)	DoFs	u_{k,k/2,k/4}(T=1)
0	6	64	64	1.8238519836868412e+02
0	7	128	128	1.6406723183673446e+02
0	8	256	256	1.5594031985663744e+02
1	6	64	128	1.4840813893702008e+02
1	7	128	256	1.4841253828679140e+02
1	8	256	512	1.4841308191020389e+02
2	6	64	192	1.4841315940666405e+02
2	7	128	384	1.4841315911202136e+02
2	8	256	768	1.4841315910286346e+02

Tabelle 1.1.: Goal functional evaluations (end time value $u_k(T = 1)$) for different polynomial degrees and different meshes.

(1) for $u_k, u_{k/2}, u_{k/4}$ separately for each set $r = 0, 1, 2$ and obtain for each $dG(r)$ scheme the convergence orders presented in Table 1.2.

These convergence orders are confirmed by Remark 1.11.2. Finally, we notice that even for the case $r = 1$ the temporal order of 3 for PDEs (not ODEs) is already high, but rather unusual to be utilized in the literature. The main reason being that the spatial solution per time interval I_m increases as previously discussed.

r	alpha
0	1.1725
1	3.0166
2	5.0078

Tabelle 1.2.: Numerically obtained convergence orders.

2. Numerik zu Eigenwertproblemen (EWP) und Singulärwertzerlegungen (SVD)

Für eine kurze Einführung sei auf Abschnitt 1.6.1 auf Seite 53 verwiesen. Das Eigenwertproblem ist gegeben durch

$$Aw = \lambda w$$

mit $A \in \mathbb{C}^{n\times n}$, $w \in \mathbb{C}^n \setminus \{0\}$, $\lambda \in \mathbb{C}$. Eigenwerte spielen in der Stabilitätsanalyse eine maßgebliche Rolle. Man betrachte etwa Eigenschwingungen, z. B. bei einer Brücke mit 8 Positionen $z_i(t) \in \mathbb{R}^2$ (siehe Abschnitt 22 in (Hanke-Bourgeois 2008)). Dann sind $z_i'(t)$ die zugehörigen Geschwindigkeitsvektoren und $v_i''(t)$ die Beschleunigungsvektoren. Zusammengefasst erhält man $z(t) \in \mathbb{R}^{16}$. Sei nun $x(t) = u(t) - z^{(0)}$ die Auslenkung. Dann erhält man

$$m\, x''(t) = -A\, x(t).$$

Man nehme $m = 1$ an. Dann erhält man die spezielle Lösung

$$x(t) = \cos\bigl(\sqrt{\lambda}t\bigr)v$$

mit einem Eigenvektor v zum Eigenwert λ von A. Die Schwingung ist periodisch mit Dauer $T = \frac{2\pi}{\sqrt{\lambda}}$, d. h. je kleiner der Eigenwert λ ist, desto länger ist die Schwingung.

Weitere Beispiele und auch Programmiercodes in MATLAB/Octave sind in (Quarteroni, Saleri und P.Gervasio 2014) zu finden.

Nun aber die Frage, wofür das Ganze? Einerseits sind Eigenwerte in der Stabilitätsanalyse von Differentialgleichungen hilfreich. Anhand von ihnen lässt sich die Hauptdynamik von Systemen bestimmen und bei einer Verallgemeinerung zu singulären Werten lässt sich sogar eine Modellreduktion erwirken. All diese Aspekte werden in diesem Kapitel betrachtet.

2.1. Geometrische Lokalisierung von Eigenwerten

Aus der Numerik I / Algorithmische Mathematik ist bereits bekannt, dass für eine verträgliche Matrixnorm $\|\cdot\|$ die Abschätzung

$$|\lambda| \leq \rho(A) \leq \|A\|, \quad \forall \lambda \in \sigma(A)$$

gilt. Hierbei beschreibt $\sigma(A)$ das Spektrum, d. h. die Menge aller (komplexen) Eigenwerte der Matrix A und $\rho(A) = \max\limits_{\lambda \in \sigma(A)} |\lambda|$ den Spektralradius. In Abbildung 2.1 ist das einmal illustriert.

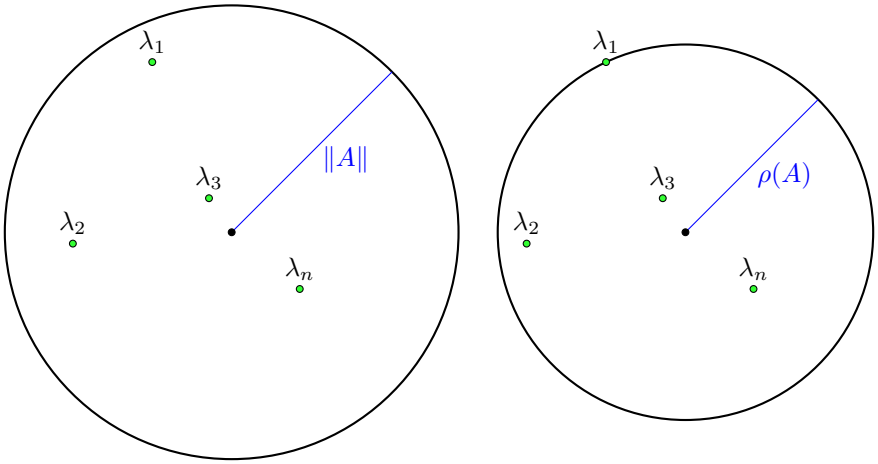


Abbildung 2.1.: Gröbst mögliche Abschätzungen der Eigenwerte.

Der Beweis dieser Aussage ist einfach. Sei dafür $Aw = \lambda w$ für $w \neq 0$. Dann gilt

$$Aw = \lambda w \quad \Rightarrow \quad |\lambda| \, \|w\| = \|\lambda w\| = \|Aw\| \leq \|A\| \, \|w\| \quad \Rightarrow \quad |\lambda| \leq \|A\|.$$

Um die geometrische Lokalisierung weiter zu verfeinern, als auch eine Stabilitätsanalyse durchzuführen, wird das folgende Lemma benötigt.

Lemma 2.1.1

Seien $A, B \in \mathbb{C}^{n \times n}$, dann gilt

$$\forall \lambda \in \sigma(A) \setminus \sigma(B) : \|(\lambda I - B)^{-1}(A - B)\| \geq 1.$$

Hier bezeichnet $\|\cdot\|$ irgendeine verträgliche Matrixnorm.

Beweis zu Lemma 2.1.1

Sei $\lambda \in \sigma(A) \setminus \sigma(B)$ mit dazugehörigem Eigenvektor $w \neq 0$. Dann gilt

$$(A - B)w = (\lambda I - B)w.$$

Weil $\lambda \notin \sigma(B)$, ist $\lambda I - B$ regulär, also invertierbar. Somit folgt

$$(\lambda I - B)^{-1}(A - B)w = w,$$

woraus sich bereits die Behauptung ergibt:

$$\begin{aligned} 1 &= \frac{\|(\lambda I - B)^{-1}(A - B)w\|}{\|w\|} \\ &\leq \sup_{w \neq 0} \frac{\|(\lambda I - B)^{-1}(A - B)w\|}{\|w\|} \overset{\|A\| = \sup\limits_{x \neq 0} \frac{\|Ax\|}{\|x\|}}{=} \|(\lambda I - B)^{-1}(A - B)\| \end{aligned}$$

□

Satz 2.1.2 Gerschgorin-Kreise

Sei $A \in \mathbb{R}^{n \times n}$. Man setze

$$K_i := \left\{ z \in \mathbb{C} \;\middle|\; |z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \right\}, \quad i = 1, \ldots, n.$$

Dann gelten folgende Aussagen:

(i) $\forall \lambda \in \sigma(A) : \lambda \in \bigcup_{i=1}^{n} K_i$

(ii) Sind $I_m = \{i_1, \ldots, i_m\}$ und $I'_m = \{1, \ldots, n\} \setminus I_m$ Indexmengen mit

$$\underbrace{\bigcup_{i \in I_m} K_i}_{=:U_m} \cap \overline{\underbrace{\bigcup_{i \in I'_m} K_i}_{=:U'_m}} = \emptyset,$$

dann sind, jeweils mit algebraischer Vielfachheit gezählt, m Eigenwerte in U_m und $n - m$ Eigenwerte in U'_m.

Beweis zu Satz 2.1.2

(i) Sei $D = \operatorname{diag}(a_{ii}) \in \mathbb{R}^{n \times n}$. Sei ferner $\lambda \in \sigma(A)$ derart, dass $\lambda \neq a_{ii}$ gilt. Aus Lemma 2.1.1 und

$\|\cdot\|_\infty$ als gewählte Matrixnorm erhält man

$$1 \leq \|(\lambda I - D)^{-1}(A - D)\|_\infty = \max_{i=1,\ldots,n} \left\{ \sum_{\substack{j=1 \\ j\neq i}}^{n} |(\lambda - a_{ii})^{-1} a_{ij}| \right\}$$

$$= \max_{i=1,\ldots,n} \left\{ |\lambda - a_{ii}|^{-1} \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \right\}.$$

Das Maximum wird in einem Index i^* angenommen, womit $\lambda \in K_{i^*}$ gilt. Ist $\lambda = a_{ii}$ für einen Index $i = 1, \ldots, n$, so gilt direkt $\lambda \in K_i$. Damit folgt der erste Teil.

(ii) Seien I_m und I'_m gemäß Aufgabenstellung und sei $B(s) := (1-s)D + sA$ für $s \in [0,1]$. Dann sind die Gerschgorin-Kreise $K_{i,s}$ bezüglich $B(s)$ gleich

$$K_{i,s} = \left\{ z \in \mathbb{C} \;\middle|\; |z - a_{ii}| \leq s \sum_{j\neq i} |a_{ij}| \right\}.$$

Sie sind also dieselben wie von A, bloß mit Radien um den Faktor s gedämpft. Ferner seien:

$$U_{m,s} := \bigcup_{i\in I_m} K_{i,s}, \quad U'_{m,s} := \bigcup_{i\in I'_m} K_{i,s}$$

Es gelten also $U_{m,s} \subseteq U_{m,s'}$ und $U'_{m,s} \subseteq U'_{m,s'}$ für $s < s'$. Daher gilt auch $U_{m,s} \cap U'_{m,s} = \emptyset$ für alle $s \in [0,1]$. Aufgrund der Inklusionen und weil die $U_{m,s}$, $U'_{m,s}$ jeweils abgeschlossene, disjunkte Mengen sind, ist $d(U_{m,s}, U'_{m,s}) > 0$ eine wohldefinierte, monoton fallende Funktion. Sie nimmt daher ihr Minimum $d_{\min}$ in $s = 1$ ein, wofür $B(1) = A$ gilt.

Für $B(0) = D$ sind $K_{i,0}$ Punktemengen, die nur die a_{ii} umschließen. Die a_{ii} sind gerade die Eigenwerte von $B(0)$, entsprechend sind in $U_{m,s}$ genau m Eigenwerte, in $U'_{m,s}$ genau $n-m$ Eigenwerte. Man bemerke, dass B die Eigenwerte von $B(s)$ stetig aufeinander abbildet. Es wird also für jeden Eigenwert λ von A eine stetige Abbildung $\lambda(s)$ induziert, worin $\lambda(s)$ Eigenwert von $B(s)$ ist. Sei o. B. d. A. $\lambda(0) \in U_{m,0}$. Damit gilt

$$d\big(\lambda(0), U'_{m,0}\big) \geq d(U_{m,0}, U'_{m,0}) \geq d_{\min}.$$

Angenommen, $\lambda(1)$ läge in $U'_{m,1}$, dann wäre $d\big(\lambda(1), U'_{m,1}\big) = 0$. Die Stetigkeit von $\lambda(s)$ implizierte die Existenz eines $0 < s_0 < 1$, für das

$$0 < d\big(\lambda(s_0), U'_{m,s_0}\big) < d_{\min} \leq d(U_{m,s_0}, U'_{m,s_0})$$

gälte. Dann wäre aber $\lambda(s_0) \notin U_{m,s_0} \cup U'_{m,s_0} = \bigcup_{i=1}^{n} K_{i,s_0}$, im Widerspruch zu (i). Daher war die Annahme falsch, $\lambda(1)$ liegt somit in $U_{m,1}$. Demnach sind genau m Eigenwerte von $B(1) = A$ in $U_{m,1}$ und $n-m$ Eigenwerte in $U'_{m,1}$. □

Weil sich die Eigenwerte einer Matrix beim Transponieren nicht ändern, können die Radien der Gerschgorin-Kreise sowohl zeilen-, als auch spaltenweise berechnet werden. Es ist aber wichtig, dass man für eine Matrix konsistent bei der Wahl bleibt. Im Allgemeinen gilt nämlich nicht

$$K_i = \left\{ z \in \mathbb{C} \;\middle|\; |z - a_{ii}| \leq \min \left\{ \sum_{j \neq i} |a_{ij}|, \sum_{j \neq i} |a_{ji}| \right\} \right\},$$

wie man beispielsweise anhand der Matrix

$$A = \begin{pmatrix} 1 & 10 \\ 0.01 & 1 \end{pmatrix}$$

sehen kann. Diese hat die Eigenwerte $\lambda_1 = 1.316$ und $\lambda_2 = 0.6838$, und würde man immer das Minimum nehmen, gälte $K_1 = K_2 = \{ z \in \mathbb{C} \mid |z - 1| \leq 0.1 \}$, worin λ_1 nicht mehr enthalten wäre! Letztendlich berechnen sich die Radien ähnlich zur Zeilen- respektive Spaltensummennorm einer Matrix, weshalb oft von Gerschgorin-Kreisen bezüglich $\| \cdot \|_\infty$ respektive $\| \cdot \|_1$ gesprochen wird.

Beispiel: Für eine illustrative Anschauung der Gerschgorin-Kreise, betrachte man die Matrix

$$A = \begin{pmatrix} 2 & 0.1 & -0.5 \\ 0.2 & 3 & 0.5 \\ -0.4 & 0.1 & 5 \end{pmatrix}.$$

Für diese ist $\sigma(A) = \{ 1.91,\ 3.01,\ 5.08 \}$, woraus man folgende Gerschgorin-Kreise bzgl. $\| \cdot \|_\infty$ (links) und bzgl. $\| \cdot \|_1$ (rechts) erhält:

$$K_1^z = \{ z \in \mathbb{C} \mid |z - 2| \leq 0.6 \} \qquad K_1^s = \{ z \in \mathbb{C} \mid |z - 2| \leq 0.6 \}$$
$$K_2^z = \{ z \in \mathbb{C} \mid |z - 3| \leq 0.7 \} \qquad K_2^s = \{ z \in \mathbb{C} \mid |z - 3| \leq 0.2 \}$$
$$K_3^z = \{ z \in \mathbb{C} \mid |z - 5| \leq 0.5 \} \qquad K_3^s = \{ z \in \mathbb{C} \mid |z - 5| \leq 1 \}$$

Man erkennt, dass mit $|\lambda| \leq \|A\|_1 = 6$ eine sehr grobe Abschätzung für die Eigenwerte existiert. In diesem Fall ist die Abschätzung per $|\lambda| \leq \|A\|_\infty = 5.5$ zwar geringfügig besser, aber bei weitem nicht so gut, wie die Abschätzung über die Gerschgorin-Kreise. Ebenfalls nennenswert ist hier insbesondere die Partition $\lambda_{1,2} \in K_1^z \cup K_2^z$, $\lambda_3 \in K_3^z$, die in Punkt (ii) des Satzes erwähnt wurde. In Bezug auf $\| \cdot \|_1$, sprich K_i^s, ist je ein Eigenwert in einem Gerschgorin-Kreis, weil alle drei Kreise disjunkt sind.

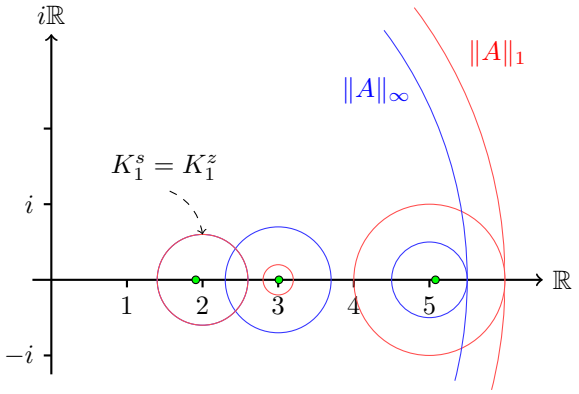


Abbildung 2.2.: Gerschgorin-Kreise der Matrix A bzgl. $\|\cdot\|_\infty$ und $\|\cdot\|_1$.

2.2. Konditionierung des Eigenwert-Problems

Satz 2.2.1 Bauer-Fike / Stabilität des Eigenwertproblems

Sei $A \in \mathbb{R}^{n\times n}$ mit n linear unabhängigen Eigenvektoren $(w_1, \ldots, w_n) = W$, d. h. W ist regulär. Man setze $\tilde{A} = A + \delta A$, wobei $\delta A \in \mathbb{R}^{n\times n}$ eine Störung, bspw. durch Rundungsfehler, darstellt. Dann existiert zu jedem Eigenwert $\tilde{\lambda} \in \sigma(\tilde{A})$ ein $\lambda \in \sigma(A)$ mit

$$|\lambda - \tilde{\lambda}| \leq \mathrm{cond}_2(W) \cdot \|\delta A\|_2, \quad \mathrm{cond}_2(W) = \|W\|_2 \, \|W^{-1}\|_2 = \frac{\lambda_{\max}}{\lambda_{\min}}$$

Beweis zu Satz 2.2.1

Sei $\lambda_i \in \sigma(A)$ mit Eigenvektor w_i. Dann ist $Aw_i = \lambda_i w_i$. Insgesamt ist dann

$$AW = W \,\mathrm{diag}(\lambda_i).$$

Da W regulär ist, folgt daraus

$$A = \mathrm{diag}(\lambda_i)\, W^{-1} \tag{1}$$

Sei nun $\tilde{\lambda} \in \sigma(\tilde{A}) \setminus \sigma(A)$. Dann ist

$$\begin{aligned} \left\|(\tilde{\lambda} I - A)^{-1}\right\|_2 &\overset{(1)}{=} \left\|\left(\tilde{\lambda} I - W\,\mathrm{diag}(\lambda_i)\, W^{-1}\right)^{-1}\right\|_2 \overset{I = WIW^{-1}}{\downarrow} = \left\|W\left(\tilde{\lambda} I - \mathrm{diag}(\lambda_i)\right)^{-1} W^{-1}\right\|_2 \\ &\leq \|W\|_2 \cdot \left\|\left(\tilde{\lambda} I - \mathrm{diag}(\lambda_i)\right)^{-1}\right\|_2 \cdot \|W^{-1}\|_2 \end{aligned}$$

$$= \operatorname{cond}_2(W) \cdot \max_{i=1,\dots,n} |\tilde{\lambda} - \lambda_i|^{-1}. \tag{2}$$

Aus Lemma 2.1.1 folgt somit

$$1 \leq \left\|(\tilde{\lambda} I - A)^{-1}(\tilde{A} - A)\right\|_2 \leq \left\|(\tilde{\lambda} I - A)^{-1}\right\|_2 \cdot \|\delta A\|_2$$

$$\overset{(2)}{\leq} \operatorname{cond}_2(W) \cdot \max_{i=1,\dots,n} |\tilde{\lambda} - \lambda_i|^{-1} \cdot \|\delta A\|_2,$$

also erhält man nichts Geringeres als

$$\max_{i=1,\dots,n} |\tilde{\lambda} - \lambda_i| \leq \operatorname{cond}_2(W) \cdot \|\delta A\|_2. \qquad \square$$

BEMERKUNG: Falls $A = A^T$ (symmetrisch) oder $A = \overline{A}^T$ (hermitsch), dann ist W eine Orthonormalbasis. Ferner ist W unitär, d. h. $W\,\overline{W}^T = I$. Demnach gilt dann $\|W\| = \|W^{-1}\| = 1$, womit das Eigenwert-Problem laut Bauer-Fike gut konditioniert ist. Allgemein hängt die Kondition des Eigenwert-Problems zu A von W ab, kann also abhängig von der Größe von $\operatorname{cond}_2(W)$ beliebig schlecht sein.

2.3. Direkte Methode

In der direkten Methode sucht man zu einer gegebenen Matrix $A \in \mathbb{R}^{n\times n}$ ein $z \in \mathbb{C}$, sodass

$$\chi_A(z) = \det(zI - A) = 0$$

gilt. Man erhält dadurch Polynome vom Grad n, dessen Nullstellen zu finden sind, z. B. mit Newton-Verfahren, Bisektion, oder Ähnlichem. Für die Nullstellensuche können die Startwerte aus den Gerschgorin-Kreisen gewählt werden. Diese Aufgabe ist jedoch schlecht konditioniert. Man betrachte etwa eine Matrix $A \in \mathbb{R}^{5\times 5}$ mit $\sigma(A) = \{1, 2, 3, 4, 5\}$. Dann ist

$$\chi_A(z) = \prod_{i=1}^{5} (z - \lambda_i) = z^5 - 15z^4 + 85z^3 - 255z^2 + 274z - 120.$$

Man stören nun den Koeffizienten von z^4 mit 0.1%, dann erhält man

$$\tilde{\chi}_A(z) = z^5 - 0.999 \cdot 15z^4 + 85z^3 - 255z^2 + 274z - 120$$

mit den dazugehörigen Nullstellen

$$\lambda_1 \approx 0.999, \quad \lambda_2 \approx 2.05, \quad \lambda_3 \approx 2.75, \quad \lambda_{4,5} \approx 4.59 \pm 0.43i \in \mathbb{C}.$$

Der Fehler liegt hiermit bei

$$\frac{|4.59 + 0.430i - 5|}{5} \approx 0.1,$$

d. h. der Fehler ist $\sim 10\%$. Das entspricht einer Fehlerverstärkung vom Faktor 100, weshalb die im Folgenden vorgestellten iterativen Methoden gebraucht werden.

2.4. Potenzmethode nach Richard von Mises

Das Lösen von Eigenwertproblemen ist eine nicht-lineare Aufgabenstellung, da bei $Aw = \lambda w$ zwei Unbekannte, der Eigenwert λ und der dazugehörige Eigenvektor w, bestimmt werden müssen. Das Lösen solcher Aufgaben besteht prinzipiell aus der Verschachtelung zweier numerischer Verfahren, ähnlich wie beim Newton-Verfahren. D. h. man hat eine äußere Iteration und innerhalb dieser müssen weitere Teilprobleme gelöst werden. Die hier behandelte Potenzmethode ist lediglich partiell, soll heißen, sie bestimmt bloß einen einzigen Eigenwert, nicht das gesamte Spektrum.

2.4.1. Voraussetzungen

Es sei $A \in \mathbb{R}^{n \times n}$ und es existiere eine Basis von Eigenvektoren $\{w_1, \ldots, w_n\}$, d. h. die Matrix A ist diagonalisierbar. Es gelte

$$|\lambda_n| > |\lambda_{n-1}| \geq \ldots \geq |\lambda_1| > 0$$

für die Eigenwerte λ_i. Man sieht, dass der betragsmäßig größte Eigenwert λ_n separiert ist. Im Allgemeinen verbessert sich die Konvergenz, je besser die Eigenwerte separiert sind. Für ein $x \in \mathbb{R}^n$ gilt in Basisdarstellung

$$x = x^{(0)} = \sum_{j=1}^{n} \alpha_j w_j, \quad \alpha_j \in \mathbb{R},\ j = 1, \ldots, n. \tag{1}$$

Dieses $x^{(0)}$ wird später der Startwert der Iteration sein. Man nehme an, dass $\alpha_n \neq 0$ ist, da man sonst den Eigenwert λ_n nicht bekommen könnte. Das muss später im Startwert berücksichtigt werden.

2.4.2. Konstruktion der Iteration

Der Grundgedanke ist hier, erst einen Eigenvektor zu approximieren, und dann den dazugehörigen Eigenwert zu berechnen. Man definiere

$$x^{(i)} := \frac{Ax^{(i-1)}}{\|Ax^{(i-1)}\|}, \quad i \geq 1.$$

Wiederholte Nutzung der Rekursionsvorschrift liefert daraus:

$$x^{(i)} = \frac{Ax^{(i-1)}}{\|Ax^{(i-1)}\|} = \frac{A \cdot \dfrac{Ax^{(i-2)}}{\|Ax^{(i-2)}\|}}{\left\| A \cdot \dfrac{Ax^{(i-2)}}{\|Ax^{(i-2)}\|} \right\|} = \frac{A^2 x^{(i-2)}}{\|A^2 x^{(i-2)}\|} = \ldots = \frac{A^i x^{(0)}}{\|A^i x^{(0)}\|} \tag{2}$$

Unter Anwendung von (1) folgt

$$A^i x^{(0)} = \sum_{j=1}^{n} \alpha_j \lambda_j^i w_j = \alpha_n \lambda_n^i \cdot \left(w_n + \sum_{j=1}^{n-1} \frac{\alpha_j}{\alpha_n} \cdot \left(\frac{\lambda_j}{\lambda_n} \right)^i \cdot w_j \right). \tag{3}$$

Da nach Voraussetzung $|\lambda_n| > |\lambda_j|$ für $j = 1, \ldots, n-1$ gilt, ist

$$1 > \frac{|\lambda_j|}{|\lambda_n|} \quad \Rightarrow \quad \left| \frac{\lambda_j}{\lambda_n} \right|^i \underset{i \to \infty}{\longrightarrow} 0 \quad \Rightarrow \quad \left(\frac{\lambda_j}{\lambda_n} \right)^i \underset{i \to \infty}{\longrightarrow} 0.$$

Somit gilt insbesondere

$$\gamma_i := \sum_{j=1}^{n-1} \frac{\alpha_j}{\alpha_n} \cdot \left(\frac{\lambda_j}{\lambda_n} \right)^i \cdot w_j \underset{i \to \infty}{\longrightarrow} 0.$$

Insgesamt ergibt sich damit also

$$\frac{A^i x^{(0)}}{\|A^i x^{(0)}\|} = \underbrace{\frac{\alpha_n \lambda_n^i}{|\alpha_n \lambda_n^i|}}_{=:\beta_i} \cdot \frac{w_n + \gamma_i}{\|w_n + \gamma_i\|}$$

Hier wurde zu Beginn angenommen, $A \in \mathbb{R}^{n \times n}$ wäre diagonalisierbar, womit alle Eigenwerte reell sind. Das alles funktioniert aber auch für diagonalisierbare $A \in \mathbb{C}^{n \times n}$, wo Eigenwerte komplex sein können. In beiden Fällen sind β_i Elemente auf dem Einheitskreis[1], im reellen Fall sind das nur ± 1. Das Problem ist, dass β_i für $i \to \infty$ nicht konvergiert, wenn $\beta_1 \neq 1$. Dieses Problem ist glücklicherweise irrelevant, denn egal wie sich β_i verhält, für hinreichend große i ist $\gamma_i \sim 0$, wodurch

$$x^{(i)} = \frac{A^i x^{(0)}}{\|A^i x^{(0)}\|} \approx \frac{\beta_i}{\|w_n\|} w_n$$

[1] Die β_i haben offensichtlich Betrag 1, aber sie sind nur wohldefiniert, weil initial $\alpha_n \neq 0$ angenommen wurde.

ein Element des Eigenraums von λ_n ist, d. h. $Ax^{(i)} \approx \lambda_n x^{(i)}$. Das gilt insbesondere in jeder k-ten Komponente $[x^{(i)}]_k \neq 0$, woraus sich

$$\lambda_n \approx \frac{[Ax^{(i)}]_k}{[x^{(i)}]_k}$$

ergibt. Dies führt zu folgendem

Satz 2.4.1 Potenzmethode nach von Mises

Es sei $A \in \mathbb{R}^{n\times n}$ mit n linear unabhängigen Eigenvektoren $\{w_1, \ldots, w_n\}$. Seien der betragsmäßig größte Eigenwert λ_n separiert und $x^{(0)}$ ein Startwert mit nicht-trivialer Komponente bezüglich w_n ($\alpha_n \neq 0$). Für einen beliebigen Index k mit $[x^{(i)}]_k \neq 0$ konvergiert die Iteration

$$x^{(i)} = \frac{Ax^{(i-1)}}{\|Ax^{(i-1)}\|}, \quad \lambda^{(i)} = \frac{[Ax^{(i)}]_k}{[x^{(i)}]_k}$$

gegen λ_n mit

$$|\lambda^{(i)} - \lambda_n| = \mathcal{O}\left(\left|\frac{\lambda_{n-1}}{\lambda_n}\right|^i\right)$$

Hier wird lediglich eine kurze Beweis-Skizze gegeben. Der vollständige Beweis kann bei Interesse in (Hanke-Bourgeois 2008, S. 220–221) nachgelesen werden.

Beweis zu Satz 2.4.1
Es gilt $\gamma_i \in \mathcal{O}\big(\left|\frac{\lambda_{n-1}}{\lambda_n}\right|^i\big)$. Damit folgt

$$\begin{aligned}\lambda^{(i)} = \frac{[Ax^{(i)}]_k}{[x^{(i)}]_k} &\overset{(2)}{=} \frac{[A^{i+1}x^{(0)}]_k}{\|A^i x^{(0)}\|} \cdot \frac{\|A^i x^{(0)}\|}{[A^i x^{(0)}]_k} \overset{(3)}{=} \frac{\alpha_n \lambda_n^{i+1}\left([w_n]_k + [\gamma_{i+1}]_k\right)}{\alpha_n \lambda_n^i \left([w_n]_k + [\gamma_i]_k\right)} \\ &= \lambda_n + \mathcal{O}\left(\left|\frac{\lambda_{n-1}}{\lambda_n}\right|^i\right).\end{aligned}$$

□

Bemerkung: Die Potenzmethode ist ein sehr einfaches Verfahren, das numerisch robust ist. Allerdings ist nur die Approximation eines einzigen Eigenwertes, hier λ_n, möglich. Es können aber auch andere Eigenwerte bestimmt werden. Der in diesem Zusammenhang einfachste Fall ist die Bestimmung des betragsmäßig kleinsten Eigenwertes $|\lambda_1|$. Dies führt einen auf die inverse Iteration nach Wieland.

2.4.3. Inverse Iteration nach Wieland

Die Idee beruht darauf, dass durch λ einer regulären Matrix $A \in \mathbb{R}^{n\times n}$ mit λ^{-1} ein Eigenwert der Matrix A^{-1} gegeben ist. D. h.

$$Aw = \lambda w \quad \Leftrightarrow \quad A^{-1}w = \lambda^{-1}w$$

Dieses Resultat ist eine Folgerung der Potenzmethode, indem diese auf $A^{-1}w$ angewendet wird. Die Potenzmethode liefert den betragsmäßig größten Eigenwerte A^{-1}. D. h. der Kehrwert ist dann $\lambda_{\min}$, sprich der betragsmäßig kleinste Eigenwert von A.

Dieses Prinzip kann man weiter verallgemeinern und einen sogenannten Shift $\sigma \in \mathbb{R}$ einführen, mit dem prinzipiell bei geschickter Wahl beliebige Eigenwerte λ_j, $j = 1, \ldots, n$, bestimmt werden können.

Es sei λ ein Eigenwert mit zugehörigem Eigenvektor $w \in \mathbb{R}^n$ zu A und $\sigma \in \mathbb{R} \setminus \sigma(A)$. Aus dem Eigenwertproblem erhält man

$$(A - \sigma I)w = (\lambda - \sigma)w \quad \Leftrightarrow \quad = (A - \sigma I)^{-1}w = (\lambda - \sigma)^{-1}w.$$

Die Äquivalenzumformung ist gültig, da mit $\sigma \notin \sigma(A)$ auch $\lambda - \sigma \neq 0$ gilt. Wenn die Potenzmethode auf $(A - \sigma I)^{-1}$ angewendet wird, erhält man eine Approximation von $(\lambda_j - \sigma)^{-1}$, wobei λ_j der Eigenwert von A ist, der am nächsten an σ liegt.

Satz 2.4.2 Inverse Iteration mit Shift

Es seien $A \in \mathbb{R}^{n\times n}$ regulär und $\sigma \in \mathbb{R}$. Für die Eigenwerte λ_i von A gelte

$$0 < |\lambda_1 - \sigma| < |\lambda_2 - \sigma| \leq |\lambda_3 - \sigma| \leq \ldots \leq |\lambda_n - \sigma|.$$

Sei $x^{(0)} \in \mathbb{R}^n$ ein Startvektor, z. B. $x^{(0)} = (1 \cdots 1)^T$. Dann konvergiert die Iteration

$$(A - \sigma I)\,\tilde{x}^{(i)} = x^{(i-1)}, \qquad x^{(i)} = \frac{\tilde{x}^{(i)}}{\|\tilde{x}^{(i)}\|},$$

$$\mu^{(i)} = \frac{\tilde{x}_k^{(i)}}{x_k^{(i-1)}}, \qquad \lambda^{(i)} = \sigma + \left(\mu^{(i)}\right)^{-1}$$

gegen den Eigenwert λ_1 mit

$$|\lambda^{(i)} - \lambda_1| = \mathcal{O}\left(\left|\frac{\lambda_1 - \sigma}{\lambda_2 - \sigma}\right|^i\right).$$

Beweis zu Satz 2.4.2
Folgt unmittelbar aus Satz 2.4.1. □

Bemerkung: Bei der inversen Iteration muss in jedem Schritt ein linear Gleichungssystem $(A - \sigma I)\,\tilde{x}^{(i)} = x^{(i-1)}$ gelöst werden. Im Allgemeinen ist die Nummerierung der λ_i unterschiedlich zu der aus der Potenzmethode. Für $\sigma = 0$ ist die Nummerierung allerdings gleich und man approximiert den betragsmäßig kleinsten Eigenwert von A.

2.5. QR-Verfahren

Wie zuvor schon bemerkt, können bei der Potenzmethode nur einzelne Eigenwerte bestimmt werden. Manchmal ist man daran interessiert, alle Eigenwerte simultan zu bestimmen. An dieser Stelle kommt das gängigste Zerlegungsverfahren ins Spiel: Das QR-Verfahren[2] (siehe (Richter und Thomas Wick 2017)). Dieses Verfahren wird auch zur Lösung von linearen Gleichungssystemen der Form $Ax = b$ genutzt und stellt daher eine Alternative zur wohl-bekannten LR-Zerlegung dar.

Ein Nachteil der LR-Zerlegung ist die schlechte Konditionierung der Matrix R. Es gilt

$$A = LR \quad \Leftrightarrow \quad L^{-1}A = R = FA$$

mit der Frobenius-Matrix F. Daraus folgt dann

$$\operatorname{cond}(R) = \operatorname{cond}(FA) \leq \operatorname{cond}(F) \cdot \operatorname{cond}(A) \leq n^2 \cdot \operatorname{cond}(A).$$

Eine Motivation des QR-Verfahrens ist die bessere Konditionierung mit einer Matrix Q, für die $\operatorname{cond}(Q) = 1$ gilt. Damit erhält man nämlich

$$\operatorname{cond}(R) \leq \underbrace{\operatorname{cond}(Q^{-1})}_{=1} \cdot \operatorname{cond}(A) = \operatorname{cond}(A).$$

2.5.1. Grundlagen

Definition 2.5.1 Orthogonale Matrix

Eine Matrix $Q \in \mathbb{R}^{n \times n}$ heißt orthogonal, falls die Zeilen- und Spaltenvektoren eine Orthonormalbasis des $\mathbb{R}^n$ bilden. Es gilt $Q^{-1} = Q^T$, d. h. $Q^T Q = I$.

Wir definieren weiter:

[2] Das Q steht für eine orthogonale Matrix und das R für eine rechte obere Dreiecksmatrix.

Definition 2.5.2

1. Symmetrische Matrix: $Q = Q^T$
2. Hermitsche Matrix: $Q = \overline{Q}^T$
3. Normale Matrix: $Q^T Q = QQ^T$ bzw. $\overline{Q^T}Q = Q\overline{Q^T}$
4. Orthogonale Matrix: $Q^{-1} = Q^T$ mit $Q^T Q = QQ^T = I$
5. Unitäre Matrix: $Q^{-1} = \overline{Q^T}$ mit $\overline{Q^T}Q = Q\overline{Q^T} = I$

Für orthogonale Matrizen Q gilt

$$\|w\|_2^2 = \langle w, w\rangle = \langle Q^T Q w, w\rangle = \langle Qw, Qw\rangle = \langle \lambda w, \lambda w\rangle = |\lambda|^2 \cdot \langle w, w\rangle = |\lambda|^2 \cdot \|w\|_2^2$$

und somit ist $|\lambda| = 1$, woraus unmittelbar $\mathrm{cond}_2(Q) = 1$ folgt.

Satz 2.5.3 Orthogonale Matrix

Es sei $Q \in \mathbb{R}^{n\times n}$ eine orthogonale Matrix. Dann ist Q regulär und es gelten

$$Q^{-1} = Q^T, \quad Q^T Q = I, \quad \|Q\|_2 = 1, \quad \mathrm{cond}_2(Q) = 1.$$

Daraus erhält man dann folgende Eigenschaften:

(i) Es gilt $\det(Q) = \pm 1$.

(ii) Für zwei orthogonale Matrizen Q_1, Q_2 ist auch $Q_1 \cdot Q_2$ eine orthogonale Matrix.

(iii) Für beliebige $x, y \in \mathbb{R}^n$ gelten

$$\|Qx\|_2 = \|x\|_2 \quad \text{und} \quad \langle Qx, Qy\rangle_2 = \langle x, y\rangle_2.$$

Definition 2.5.4 QR-Zerlegung

Die Zerlegung von $A \in \mathbb{R}^{n\times n}$ in eine orthogonale Matrix Q und eine rechte obere Dreiecksmatrix R mit $A = QR$ heißt QR-Zerlegung.

Satz 2.5.5 Existenz einer QR-Zerlegung

Zu einer regulären Matrix $A \in \mathbb{R}^{n\times n}$ existiert eine QR-Zerlegung $A = QR$. Diese ist bis auf Vorzeichen eindeutig. Für die Eindeutigkeit fordert man $r_{ii} > 0$, $i = 1, \ldots, n$.

2.5.2. Konstruktion der Matrix Q

Es sei $\{a_1, \ldots, a_n\}$ eine Basis des $\mathbb{R}^n$. Laut Basistransformationssatz kann $\{a_1, \ldots, a_n\}$ in eine orthogonale bzw. orthonormale Basis überführt werden. Dies geschieht z. B. mit Hilfe des Gram-Schmidt-Verfahrens. Numerisch ist das Gram-Schmidt-Verfahren allerdings nicht geeignet, da dieses auf orthogonalen Projektionen basiert und hier durch Auslöschungseffekte leicht die Orthogonalitätsbedingungen verletzt werden. Eine erste Verbesserung bietet das modifizierte Gram-Schmidt-Verfahren, worin der Orthogonalisierungsschritt durch eine iterative Vorschrift ersetzt wird (siehe (Richter und Thomas Wick 2017)[S. 98]). Weitere entscheidende Verbesserungen in der numerischen Stabilität werden aber durch zwei alternative Verfahren erreicht:

(i) Spiegelungen (Householder-Verfahren)

(ii) Rotationen (Givens-Verfahren)

2.5.3. Householder-Verfahren zur Erstellung einer QR-Zerlegung

Hier sei nochmal an ein Resultat der linearen Algebra erinnert: Gilt für eine Matrix $\det(Q) = 1$, so handelt es sich um eine Drehung, für $\det(Q) = -1$ ist Q eine Spiegelung.

Definition 2.5.6 Householder-Transformation

Für einen Vektor $v \in \mathbb{R}^n$ mit $\|v\|_2 = 1$ ist durch $vv^T \in \mathbb{R}^{n\times n}$ das dyadische Produkt

$$\begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \cdot (v_1 \cdots v_n) = \begin{pmatrix} v_1v_1 & v_1v_2 & \cdots & v_1v_n \\ \vdots & & & \vdots \\ v_nv_1 & \cdots & \cdots & v_nv_n \end{pmatrix} \in \mathbb{R}^{n\times n}$$

definiert. Die Matrix $S := I - 2vv^T \in \mathbb{R}^{n\times n}$ heißt Householder-Transformation.

Satz 2.5.7 Eigenschaften von Householder-Transformationen

Für $S = I - 2vv^T$ gelten folgende Aussagen:

(i) S ist symmetrisch ($S = S^T$).

(ii) S ist orthogonal ($S^{-1} = S^T$).

(iii) $S_1 S_2$ ist orthogonal, wenn S_1, S_2 Householder-Transformationen darstellen.

Bemerkung: Sei $v \in \mathbb{R}^n$ mit $\|v\|_2 = 1$. Weiter sei $x \in \mathbb{R}^n$ mit

$$x = \alpha v + w^\perp, \quad \alpha \in \mathbb{R},$$

wobei $w^\perp \in \mathbb{R}^n$ ist mit $v^T w^\perp = 0$. Dann ist

$$S(\alpha v + w^\perp) = (I - 2vv^T)(\alpha v + w^\perp) = \alpha(v - 2v \underbrace{v^T v}_{=1}) + w^\perp - 2v \underbrace{v^T w^\perp}_{=0} = -\alpha v + w^\perp.$$

Anhand des negativen Vorzeichens sieht man, dass die Householder-Transformation an dieser Stelle eine Spiegelung entlang des Orthogonalraums zu v darstellt.

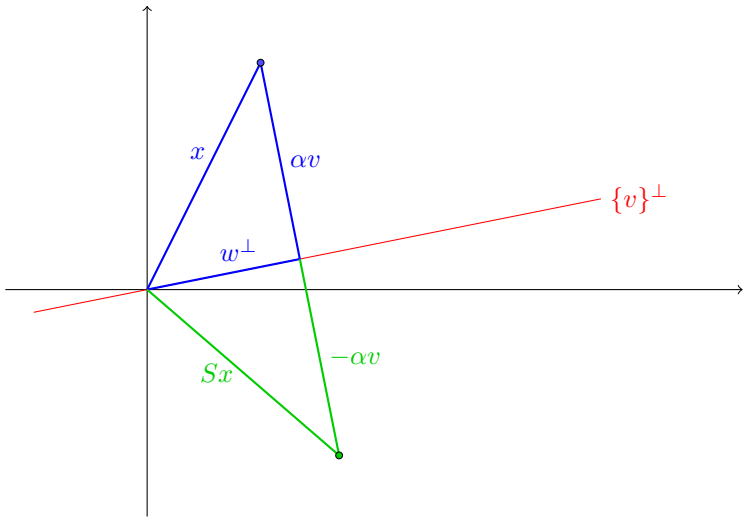


Abbildung 2.3.: Spiegelung von x am Orthogonalraum $\{v\}^\perp$ als Folge der Householder-Transformation.

Konstruktion

Das Ziel ist es, mit Hilfe von S eine Matrix $A \in \mathbb{R}^{n\times n}$ sukzessive in eine obere Dreiecksmatrix zu transformieren. Konkret heißt das

$$A^{(0)} := A, \quad A^{(i)} = S^{(i)}A^{(i-1)}, \quad S^{(i)} = I - 2v^{(i)}(v^{(i)})^T.$$

Am Ende erhält man $R := A^{(n-1)}$.

1. Schritt: Hier versucht man, den ersten Spaltenvektor in ein Vielfaches des ersten Einheitsvektors umzuformen, weil man damit eine orthogonale Matrix erstellt. Man möchte also

$$A^{(1)} = \left(a_{ij}^{(1)}\right)_j = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(1)} & & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{pmatrix}$$

erhalten mittels $a_j^{(1)} = S^{(1)}a_j^{(0)}$. Hier stellt sich die Frage, wie $S^{(1)}$ gewählt werden muss, sodass die Bedingung

$$a_1^{(1)} = S^{(1)}a_1^{(0)} \in \operatorname{span}\{e_1\},$$

die an $A^{(1)}$ gestellt wird, erfüllt ist. Hierbei bezeichnet e_1 den ersten Einheitsvektor im $\mathbb{R}^n$. Dafür sei an Abbildung 2.3 verwiesen: Abhängig vom Vektor $a_1^{(0)}$ wird die Householder-Transformation als eine Spiegelung konstruiert, die den Vektor auf $\operatorname{span}\{e_1\}$ transformiert. Beispielsweise kann man mit $\operatorname{span}\left(a_1^{(0)} \pm \|a_1^{(0)}\|\, e_1\right)$ entsprechende Spiegelgeraden konstruieren.

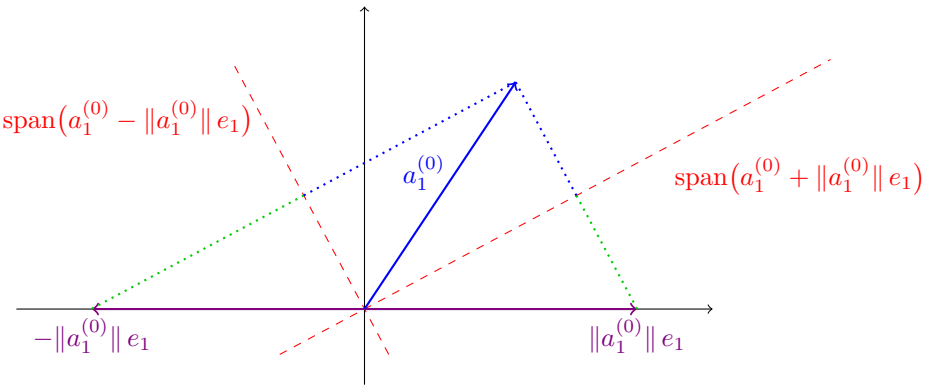


Abbildung 2.4.: Spiegelung von $a_1^{(0)}$ auf $\operatorname{span}\{e_1\}$.

Dies führt zu $v^{(1)} = a_1^{(0)} \pm \|a_1^{(0)}\| \, e_1$. Aus numerischen Stabilitätsgründen nimmt man

$$v^{(1)} = \frac{a_1^{(0)} + \operatorname{sign}(a_{11}^{(0)}) \, \|a_1^{(0)}\| \, e_1}{\left\| a_1^{(0)} + \operatorname{sign}(a_{11}^{(0)}) \, \|a_1^{(0)}\| \, e_1 \right\|}$$

mit

$$\operatorname{sign}(x) = \begin{cases} -1, & x < 0, \\ 0, & x = 0, \\ 1, & x > 0. \end{cases}$$

Die Wahl von $\operatorname{sign}(x)$ vermindert Auslöschungseffekte. Damit folgt dann

$$\begin{aligned} a_1^{(1)} = S^{(1)} a_1^{(0)} &= -\operatorname{sign}(a_{11}^{(0)}) \, \|a_1^{(0)}\| \, e_1 \\ a_j^{(1)} = S^{(1)} a_j^{(0)} &= a_j^{(0)} - 2\langle v^{(1)}, a_j^{(0)} \rangle \, v^{(1)}, \quad j = 2, \ldots, n \end{aligned}$$

$$\Rightarrow A^{(1)} = S^{(1)} A^{(0)} = \left(\begin{array}{c|ccc} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ \hline 0 & a_{22}^{(1)} & & a_{2n}^{(1)} \\ \vdots & & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{array} \right),$$

wobei im folgenden der untere rechte Block die Matrix $\tilde{A}^{(1)} \in \mathbb{R}^{(n-1)\times(n-1)}$ darstellt.

2. Schritt: Man konstruiert nun $\tilde{S}^{(2)} \in \mathbb{R}^{(n-1)\times(n-1)}$, indem man den obigen Algorithmus für $a_2^{(2)}$ und $a_i^{(2)}$, $i = 3, \ldots, n$, anwendet. D. h.

$$\tilde{a}_2^{(1)} = (a_{22}^{(1)}, a_{32}^{(1)}, \ldots, a_{n2}^{(1)})^T \in \mathbb{R}^{n-1}.$$

Damit folgt dann

$$\tilde{v}^{(2)} := \frac{\tilde{a}_2^{(1)} + \operatorname{sign}(a_{22}^{(1)}) \, \|\tilde{a}_2^{(1)}\| \, \tilde{e}_2}{\left\| \tilde{a}_2^{(1)} + \operatorname{sign}(a_{22}^{(1)}) \, \|\tilde{a}_2^{(1)}\| \, \tilde{e}_2 \right\|}$$

mit $\tilde{e}_2$ als ersten[3] Einheitsvektor im $\mathbb{R}^{n-1}$. Daraus erhält man

$$S^{(2)} = \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \tilde{I} - 2\tilde{v}^{(2)}(\tilde{v}^{(2)})^T & \\ 0 & & & \end{array}\right) \in \mathbb{R}^{n\times n}$$

Dabei sei der rechte untere Block als $\tilde{S}^{(2)} \in \mathbb{R}^{(n-1)\times(n-1)}$ bezeichnet. Dann ist $A^{(2)} = S^{(2)}A^{(1)}$.

i-ter Schritt: Man erhält mit dem gleichen Vorgehen

$$S^{(i)} = \left(\begin{array}{ccc|c} 1 & & 0 & \\ & \ddots & & 0 \\ 0 & & 1 & \\ \hline & 0 & & \tilde{I} - 2\tilde{v}^{(2)}(\tilde{v}^{(2)})^T \end{array}\right) \in \mathbb{R}^{n\times n}$$

mit $\tilde{S}^{(i)} \in \mathbb{R}^{(n-i+1)\times(n-i+1)}$ als rechter unterer Block.

Nach $n-1$ Schritten erhält man dann

$$R := \underbrace{S^{(n-1)}S^{(n-2)}\cdots S^{(1)}}_{=:Q^T}A$$

und somit $QR = A$.

Satz 2.5.8 Aufwand der QR-Zerlegung

Sei $A \in \mathbb{R}^{n\times n}$ regulär. Dann lässt sich die QR-Zerlegung mit Hilfe der Householder-Transformation in

$$\frac{2n^3}{3} + \mathcal{O}(n^2)$$

arithmetischen Operationen numerisch stabil durchführen.

[3] Im $\mathbb{R}^n$ wäre dies der zweite Einheitsvektor, darum der Index 2. Durch "Abschneiden" der ersten Zeile und Spalte ist dies dann der erste Einheitsvektor in einer Dimension geringer.

Beweis zu Satz 2.5.8
Für die Berechnung von $\tilde{v}^{(i)}$ betrachte man

$$c_1^{(i)} := \sum_{k=2}^{i-1} \big[\tilde{a}_i^{(i-1)}\big]_k^2, \quad c_2^{(i)} := \operatorname{sign}\big(a_{ii}^{(i-1)}\big) \underbrace{\sqrt{\big[a_i^{(i-1)}\big]_1^2 + c_1^{(i)}}}_{= \|\tilde{a}_i^{(i-1)}\|}.$$

Hierin berechnet sich $c_1^{(i)}$ in $n-i-1$ Additionen, das Quadrieren ist jeweils ein zusätzlicher konstanter Aufwand $\mathcal{O}(1)$. Für $c_2^{(i)}$ hat man einen zusätzlichen Summanden, also $n-i$ Additionen. Quadrieren, Wurzelziehen und Multiplizieren ist jeweils konstanter Aufwand, also lässt sich $c_2^{(i)}$ in $n-i+\mathcal{O}(1)$ Schritten berechnen. Jetzt sei

$$w^{(i)} := \tilde{a}_i^{(i-1)} + c_2^{(i)}\, \tilde{e}_i \quad \Rightarrow \quad \tilde{v}^{(i)} = \frac{w^{(i)}}{\|w^{(i)}\|}.$$

Das heißt $w^{(i)}$ entsteht durch Addition von dem bereits berechneten $c_2^{(i)}$ in die erste Komponente von $\tilde{a}_i^{(i-1)}$, ist also konstanter Aufwand. Für $\|w^{(i)}\|$ kann man auf vorherige Ergebnisse zurückgreifen:

$$\|w^{(i)}\| = \sqrt{c_1^{(i)} + \big(\,\big[\tilde{a}_i^{(i-1)}\big]_1 + c_2^{(i)}\big)^2}$$

Also ist auch hier nur konstanter Aufwand notwendig. In $\tilde{v}^{(i)}$ müssen nun noch $n-i$ Divisionen ausgeführt werden, wodurch sich insgesamt ein Aufwand von $2(n-i)+\mathcal{O}(1)$ für die Berechnung von $\tilde{v}^{(i)}$ ergibt. Für das QR-Verfahren kommt zusätzlich noch

$$a_j^{(i)} = \underbrace{a_j^{(i-1)}}_{n-i \text{ Op.}} - \underbrace{2\big\langle v^{(i)}, a_j^{(i-1)}\big\rangle}_{n-i+\mathcal{O}(1) \text{ Op.}} v^{(i)}$$

für $j = 1, \ldots, n$, was also $2(n-i)^2 + \mathcal{O}(n)$ Aufwand bedeutet. Alles zusammen ergibt also einen Aufwand von $2(n-i)^2 + 2(n-i) + \mathcal{O}(n)$ für $i = 1, \ldots, n-1$. Das resultiert in

$$2\sum_{i=1}^{n-1}\big((n-i)^2 + (n-i) + \mathcal{O}(n)\big) = \frac{2n^3}{3} + \mathcal{O}(n^2).$$

Die numerische Stabilität folgt aus $\operatorname{cond}_2\big(S^{(i)}\big) = 1$. □

Bemerkung: Sowohl das LR- wie auch das QR-Verfahren benötigen zum Erstellen der Zerlegung $\mathcal{O}(n^3)$ Operationen. D. h. die Effizienz ist asymptotisch gesehen gleich. Allerdings ist die Konditionierung der Matrix R bei der QR-Zerlegung besser, als mit der LR-Zerlegung. Andererseits ist die Herleitung des QR-Verfahrens komplizierter, sowie dessen Implementierung aufwendiger.

BEISPIEL Es sei $n = 3$, d. h. $A \in \mathbb{R}^{3\times 3}$.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \quad a_1 = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix}, \quad a_2 = \begin{pmatrix} a_{12} \\ a_{22} \\ a_{32} \end{pmatrix}, \quad a_3 = \begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix}$$

Erster Schritt: Man bildet

$$v_1^{(1)} = \frac{a_1 + \operatorname{sign}(a_{11}) \, \|a_1\| \, e_1}{\left\| a_1 + \operatorname{sign}(a_{11}) \, \|a_1\| \, e_1 \right\|}.$$

Damit erhält man

$$a_1^{(1)} = \; - \operatorname{sign}(a_{11}) \, \|a_1\| \, e_1,$$

$$a_i^{(1)} = a_i - 2\langle v^{(1)}, a_i \rangle \, v^{(1)}, \quad i = 2, 3.$$

Daraus erhält man dann die Matrix

$$A^{(1)} = S^{(1)} A^{(0)} = \left(\begin{array}{c|cc} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ \hline 0 & a_{22}^{(1)} & a_{32}^{(1)} \\ 0 & a_{23}^{(1)} & a_{33}^{(1)} \end{array} \right),$$

woraus man

$$\tilde{A}^{(1)} = \begin{pmatrix} a_{22}^{(1)} & a_{32}^{(1)} \\ a_{23}^{(1)} & a_{33}^{(1)} \end{pmatrix} \in \mathbb{R}^{2\times 2}$$

abliest. Für den zweiten Schritt benutzt man nun

$$\tilde{a}_2^{(1)} = \begin{pmatrix} a_{22}^{(1)} \\ a_{23}^{(1)} \end{pmatrix}, \quad \tilde{a}_3^{(1)} = \begin{pmatrix} a_{32}^{(1)} \\ a_{33}^{(1)} \end{pmatrix}$$

und bildet damit

$$\tilde{v}^{(2)} = \frac{\tilde{a}_2^{(1)} + \operatorname{sign}(a_{22}^{(1)}) \, \|\tilde{a}_2^{(1)}\| \, \tilde{e}_2}{\left\| \tilde{a}_2^{(1)} + \operatorname{sign}(a_{22}^{(1)}) \, \|\tilde{a}_2^{(1)}\| \, \tilde{e}_2 \right\|}$$

mit $\tilde{e}_2 \in \mathbb{R}^2$ als ersten Einheitsvektor. Damit erhält man

$$\tilde{a}_2^{(2)} = \; - \operatorname{sign}(a_{22}^{(1)}) \, \|\tilde{a}_2^{(1)}\| \, \tilde{e}_2,$$

$$\tilde{a}_i^{(2)} = \tilde{a}_i^{(2)} - 2\langle \tilde{v}^{(2)}, \tilde{a}_i^{(2)} \rangle \, \tilde{v}^{(2)}, \quad i = 3.$$

Daraus bekommt man die kompakte Schreibweise

$$\tilde{S}^{(2)} = I - 2\tilde{v}^{(2)}(\tilde{v}^{(2)})^T.$$

Nun sind die $n-1$ Schritte beendet, also hat man

$$R = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & \tilde{a}_{22}^{(2)} & \tilde{a}_{23}^{(2)} \\ 0 & 0 & \tilde{a}_{33}^{(2)} \end{pmatrix}.$$

D. h. $R = S^{(2)}A^{(1)} = S^{(2)}S^{(1)}A^{(0)}$ und somit ist

$$Q = (S^{(2)}S^{(1)})^T = (S^{(1)})^T(S^{(2)})^T$$

mit

$$S^{(1)} = I - 2v^{(1)}(v^{(1)})^T$$

$$S^{(2)} = \left(\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & \multicolumn{2}{c}{\multirow{2}{*}{$\tilde{S}^{(2)}$}} \\ 0 & & \end{array}\right) = \left(\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & \multicolumn{2}{c}{\multirow{2}{*}{$\tilde{I} - 2\tilde{v}^{(2)}(\tilde{v}^{(2)})^T$}} \\ 0 & & \end{array}\right)$$

Beispiel 2: Für die Matrix

$$\begin{pmatrix} 1 & 4 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 1 \end{pmatrix}$$

erhält man somit nach längerer Rechnung (diese wird dem interessierten Leser überlassen) die Matrizen

$$Q = \begin{pmatrix} 0.408 & 0.816 & 0.408 \\ 0.802 & -0.535 & 0.267 \\ 0.436 & 0.218 & -0.873 \end{pmatrix}, \quad R = \begin{pmatrix} 2.449 & 2.449 & 2.041 \\ 0 & 3.742 & 1.336 \\ 0 & 0 & 0.218 \end{pmatrix}.$$

Was hier schwer zu sehen ist: Die Matrix Q ist tatsächlich orthogonal.

2.5.4. QR-Verfahren zur Eigenwert-Berechnung

Grundlegend beobachtet man

$$A = QR = QR(QQ^T) = Q(RQ)Q^T,$$

also dass A orthogonal ähnlich zur Matrix RQ ist. Das heißt insbesondere, dass A und RQ dieselben Eigenwerte haben. Das führt zu folgendem Algorithmus zur Berechnung von Eigenwerten einer regulären Matrix $A \in \mathbb{R}^{n\times n}$:

1. `Setze:`

   $A_0 := A$

2. `Für` $k = 0, 1, 2, \ldots$

   `Erstelle QR-Zerlegung` $A_k = Q_k R_k$

   `Berechne` $A_{k+1} = R_k Q_k$

Eine Basis-Implementierung dieses Algorithmus findet der interessierte Leser im Program 6.4 auf Seite 207 in (Quarteroni, Saleri und P.Gervasio 2014). Wie zu sehen ist, erhält man mit dem QR-Verfahren zur Eigenwertberechnung eine Folge von Matrizen, die ähnlich zu A sind. Alle A_k haben demnach dieselben Eigenwerte. Was noch zu zeigen ist, ist dass die A_k gegen eine obere Dreiecksmatrix konvergieren, deren Diagonaleinträge die Eigenwerte von A hat. Das besagt der folgende Satz:

Satz 2.5.9 QR-Verfahren und Eigenwertberechnung

Es sei $A \in \mathbb{R}^{n\times n}$ mit separierten Eigenwerten, d. h.

$$|\lambda_1| > |\lambda_2| > |\lambda_3| > \ldots > |\lambda_{n-1}| > |\lambda_n|.$$

Dann gilt für die Diagonalelemente $a_{ii}^{(k)}$ für $i = 1, \ldots, n$ der Matrix A_k

$$\left\{a_{11}^{(k)}, a_{22}^{(k)}, \ldots, a_{nn}^{(k)}\right\} \overset{k\to\infty}{\longrightarrow} \{\lambda_1, \lambda_2, \ldots, \lambda_n\}.$$

Die Matrix A_k hat die Gestalt

$$A_{k\to\infty} = \begin{pmatrix} \lambda_1 & & * \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}.$$

Falls $A = A^T$ gilt, so sind zusätzlich $a_{ij}^{(k\to\infty)} = 0$ für $j > i$.

Für den Beweis dieses Satzes wird zuvor noch ein anderes Resultat benötigt, dessen Beweis widerum dem Leser zur Übung überlassen wird.

Lemma 2.5.10

Sei $(M_k) \subset \mathbb{R}^{n\times n}$ eine Folge regulärer Matrizen mit QR-Zerlegungen $M_k = Q_k R_k$, wobei $r_{ii}^{(k)} > 0$ für alle $i = 1, \ldots, n$. Ferner gelte $\lim_{k\to\infty} M_k = I$, dann gilt auch

$$\lim_{k\to\infty} Q_k = I = \lim_{k\to\infty} R_k.$$

Beweis zu Satz 2.5.9

Sei $\Lambda := \operatorname{diag}(\lambda_i)$, dann gilt laut Verfahrensvorschrift über die Orthogonalität von Q_k

$$A_{k+1} = R_k Q_k = \underbrace{Q_k^T \overbrace{Q_k R_k}^{=A_k}}_{=I} Q_k = Q_k^T A_k Q_k.$$

Wendet man das iterativ auf sich selbst an, so folgt daraus

$$A_{k+1} = Q_k^T \cdots Q_0^T A_0 \underbrace{Q_0 \cdots Q_k}_{=:P_k} = P_k^T A P_k. \tag{1}$$

Zusätzlich definiere man $S_k := R_k \cdots R_0$. Dann erhält man

$$P_k S_k = P_{k-1} Q_k R_k S_{k-1} = P_{k-1} A_k S_{k-1} \overset{(1)}{=} P_{k-1} P_{k-1}^T A P_{k-1} S_{k-1}.$$

Als Produkt orthogonaler Matrizen ist P_{k-1} ebenfalls orthogonal, womit sich weiterhin

$$P_k S_k = A P_{k-1} S_{k-1} = A^2 P_{k-2} S_{k-2} = A^k P_0 S_0 = A^{k+1} \tag{2}$$

ergibt. $P_k S_k$ ist demnach eine QR-Zerlegung von A^{k+1}. Aufgrund der separierten Eigenwerte ist A diagonalisierbar. Es gibt also eine reguläre Matrix W, sodass $A = W\Lambda W^{-1}$ gilt. Seien

(i) $L_W S_W = P_W W^{-1}$ eine LR-Zerlegung und

(ii) $Q_W R_W = W P_W^{-1}$ eine QR-Zerlegung.

P_W beschreibt hierbei einer Permutationsmatrix, durch die die Existenz einer LR-Zerlegung gewährleistet ist. Weil Λ^k eine Diagonalmatrix ist, sind Zeilen- und Spaltenoperationen auf ihr identisch. D. h. sämtliche Permutationen in den Zeilen von P_W werden durch Permutationen in den Spalten von P_W^{-1} rückgängig gemacht, womit

$$P_W \Lambda^k P_W^{-1} = \Lambda^k \tag{3}$$

gilt. Es folgt

$$A^k = (W\Lambda W^{-1})^k = W\Lambda^k W^{-1} \overset{\text{(i)}}{=} W\Lambda^k P_W^{-1} L_W S_W \overset{\text{(ii)}}{=} Q_W R_W P_W \Lambda^k P_W^{-1} L_W S_W$$

$$\overset{(3)}{=} Q_W R_W \Lambda^k L_W S_W = Q_W R_W \big[\Lambda^k L_W \underbrace{\Lambda^{-k}}_{=I}\big] \Lambda^k S_W$$

$$= Q_W R_W \begin{pmatrix} 1 & & & 0 \\ & \ddots & & \\ & & \ddots & \\ \ell_{ij}\left(\frac{\lambda_i}{\lambda_j}\right)^k & & & 1 \end{pmatrix} \Lambda^k S_W = Q_W R_W (I + N_k) \Lambda^k S_W$$

$$= Q_W (R_W + R_W N_k) \underbrace{R_W^{-1} R_W}_{=I} \Lambda^k S_W = Q_W (I + R_W N_k R_W^{-1}) R_W \Lambda^k S_W.$$

Nach Voraussetzung gilt $\left|\frac{\lambda_i}{\lambda_j}\right| < 1$ für alle $i > j$, und somit ist $N_k \to 0$, also $I + R_W N_k R_W^{-1} \to I$. Sei $\tilde{Q}_k \tilde{R}_k$ eine QR-Zerlegung von $I + R_W N_k R_W^{-1}$ mit $\tilde{r}_{ii}^{(k)} > 0$. Aus Lemma 2.5.10 ergibt sich $\tilde{Q}_k,\ \tilde{R}_k \to I$ für $k \to \infty$. Somit ist

$$A^{k+1} = \underbrace{Q_W \tilde{Q}_{k+1}}_{\text{orthogonal}} \underbrace{\tilde{R}_{k+1} R_W \Lambda^{k+1} S_W}_{\text{obere Dreiecksmatrix}}$$

eine weitere QR-Zerlegung von A^{k+1}. Eine QR-Zerlegung ist für $r_{ii} > 0$ eindeutig, also gibt es eine Matrix $D_k = \operatorname{diag}(\pm 1)$, sodass $P_k = Q_W \tilde{Q}_{k+1} D_k$ gilt. Nun gilt auch

$$A = W \Lambda W^{-1} \overset{\text{(ii)}}{=} (Q_W R_W P_W) \Lambda (P_W^{-1} R_W^{-1} Q_W^{-1}) \overset{(3)}{=} Q_W R_W \Lambda R_W^{-1} Q_W^{-1}$$

$$\Leftrightarrow \quad Q_W^T A Q_W = R_W \Lambda R_W^{-1}.$$

Zusammen mit $P_k = Q_W \tilde{Q}_{k+1} D_k$ schließt man hieraus

$$A_{k+1} \overset{(1)}{=} P_k^T A P_k = D_k^T \tilde{Q}_{k+1}^T Q_W^T A Q_W \tilde{Q}_{k+1} D_k$$

$$= \underbrace{D_k^T \tilde{Q}_{k+1}^T}_{\to} \quad \underbrace{R_W \Lambda R_W^{-1}}_{=} \quad \underbrace{\tilde{Q}_{k+1} D_k}_{\to}$$

$$\overset{k\to\infty}{\longrightarrow} \begin{pmatrix} * & & 0 \\ & \ddots & \\ 0 & & * \end{pmatrix} \begin{pmatrix} * & \cdots & * \\ & \ddots & \vdots \\ 0 & & * \end{pmatrix} \begin{pmatrix} * & & 0 \\ & \ddots & \\ 0 & & * \end{pmatrix} = \begin{pmatrix} * & \cdots & * \\ & \ddots & \vdots \\ 0 & & * \end{pmatrix}.$$

Die Eigenwerte der Matrix A verändern sich während der Iteration nicht, also hat die Ergebnismatrix ebenfalls dieselben Eigenwerte. Die obere Dreiecksmatrix A_k für $k \to \infty$ hat demnach die Eigenwerte von A auf der Hauptdiagonalen stehen. □

Bemerkung: Als Abbruchskriterium der QR-Zerlegung wird

$$\max_{i>j} |a_{ij}^{(k)}| \leq \text{TOL} \quad (\text{z. B. } 10^{-8}).$$

genutzt, insbesondere auch für $A = A^T$. Im Allgemeinen konvergieren die Diagonalelemente von A_k mindestens linear gegen die gesuchten Eigenwerte. In speziellen Fällen (und mit speziellen Tricks) lässt sich sogar kubische Konvergenz erzielen. Die Konvergenzgeschwindigkeit hängt wiederum von der Separation der Eigenwerte ab. D. h. je besser die Eigenwerte separiert sind, desto schneller die Konvergenz.

2.5.5. QR-Verfahren basierend auf vorheriger Reduktion

Das QR-Verfahren ist oftmals von langsamer Konvergenz geplagt. Die Zahl der Schritte übersteigt nicht unüblich 100, und in jedem Schritt liegt ein Aufwand von $\mathcal{O}(n^3)$ vor. Das ist also ein sehr teures Unterfangen, weshalb man sich dafür interessiert, ob man den Aufwand pro Schritt reduzieren kann? Hierfür betrachte man die Eigenwert-Aufgabe für eine Matrix $A \in \mathbb{R}^{n\times n}$. Ziel wäre es, A zu einer Matrix $\tilde{A}$ mit "einfacherer" Struktur zu reduzieren, wie etwa einer Dreiecksmatrix, einer Bandmatrix, o. Ä. Bei den meisten Reduktionsmethoden müssen allerdings bereits Eigenwerte bekannt sein, welche aber mit dem Verfahren erst gelöst werden sollen. Das wird also die Hauptproblematik, die genauer zu untersuchen ist. Sobald man eine reduzierte Matrix $\tilde{A}$ vorliegen kann, kann man das QR-Verfahren darauf anwenden, und durch die spezielle Struktur einen Aufwand von lediglich $\mathcal{O}(n^2)$ pro Schritt erhalten. Natürlich birgt eine Reduktion auch Kosten, aber diese treten bloß einmalig bei der initialen Reduktion auf, nicht aber bei jedem Iterationsschritt des Verfahrens.

Grundlage der Reduktion sind, wie so häufig, Ähnlichkeitstransformationen

$$A = A^{(0)} \quad \longrightarrow \quad A^{(1)} = S_1^{-1} A^{(0)} S_1 \quad \longrightarrow \quad \cdots \quad \longrightarrow \quad A^{(k)} = S_k^{-1} A^{(k-1)} S_k,$$

die die Matrix A schrittweise in eine ähnliche Matrix überführen, die eine möglichst einfache Struktur hat.

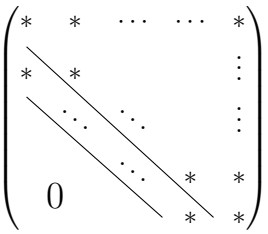
Definition 2.5.11 Schursche Normalform

Sei $A \in \mathbb{C}^{n\times n}$ mit Eigenwerten $\lambda_1, \ldots, \lambda_n$. Dann existiert eine unitäre Matrix $U \in \mathbb{C}^{n\times n}$ mit

$$\overline{U}^T A U = \begin{pmatrix} \lambda_1 & & * \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}.$$

Aus numerischer Sicht ist die Schursche Normalform leider nicht optimal, weil diese zu hohen Aufwand hat, um berechnet zu werden. Obwohl der Aufwand nur einmalig auftritt, überwiegt

dieser in Regel den Einsparungen in den folgenden Iterationsschritten. Darum wird eine Form benötigt, die noch "nah genug" an diese Form herankommt, aber leichter zu erhalten ist. Das führt zu der Hessenberg-Normalform:



(4)

Diese ist keine obere Dreiecksmatrix, sondern trägt noch eine zusätzliche Nebendiagonale. Berechnet wird diese im Folgenden über Householder-Transformationen.

Satz 2.5.12 Hessenberg-Normalform

Zu jeder Matrix $A \in \mathbb{R}^{n\times n}$ existiert eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$, sodass $Q^T AQ$ die Form aus (4) besitzt. Falls $A = A^T$, so ist $Q^T AQ$ eine Tridiagonalmatrix.

Beweis zu Satz 2.5.12

Der Beweis wird via Konstruktion über Householder-Transformationen durchgeführt. Im Unterschied zur QR-Zerlegung müssen die orthogonalen Householder-Matrizen $S_{(i)}$ allerdings beidseitig, nicht nur von links, an A multipliziert werden. Entsprechend wird zur Gewährleistung der Ähnlichkeitsstruktur eine von der QR-Zerlegung unterschiedliche Wahl der Vektoren $v^{(i)}$ in $S_{(i)} = I - 2v^{(i)}\left(v^{(i)}\right)^T$ stattfinden, was letztendlich zu der entstehenden Nebendiagonale führt.

Schritt 1: Seien $A = (a_1, \ldots, a_n)$ die Spaltenvektoren von A. Gesucht ist ein Vektor $v^{(1)} = \left(0, v_2^{(1)}, \ldots, v_n^{(1)}\right)^T$, sodass $S_{(1)}a_1 \in \operatorname{span}(e_1, e_2)$. Dazu passt die Wahl

$$v^{(1)} = \frac{\tilde{a}_1 + \|\tilde{a}_1\| e_2}{\left\|\tilde{a}_1 + \|\tilde{a}_1\| e_2\right\|}$$

mit $\tilde{a}_1 = (0, a_{21}, \ldots, a_{n1})^T$. Hier wird im Gegensatz zur QR-Verteilung kein sign benötigt, weil durch $S_{(1)}^{-1} A S_{(1)}$ die Vorzeichen immer positiv sind. Zudem benutzt man hier bereits in der ersten Iteration den zweiten Einheitsvektor e_2 und setzt in a_1 die erste Komponente auf 0. Für diese

Wahl gilt

$$A^{(1)} = \underbrace{\left(\begin{array}{c|ccc} a_{11} & a_{12} & \cdots & a_{1n} \\ \hline * & * & \cdots & * \\ 0 & \vdots & & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \cdots & * \end{array}\right)}_{S_{(1)}A^{(0)}} \underbrace{\left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \cdots & * \end{array}\right)}_{S_{(1)}^T} = \left(\begin{array}{c|ccc} a_{11} & a_{12} & \cdots & a_{1n} \\ \hline * & * & \cdots & * \\ 0 & \vdots & & \vdots \\ \vdots & \vdots & \tilde{A}^{(1)} & \vdots \\ 0 & * & \cdots & * \end{array}\right)$$

In Schritt 2 wird Householder auf $\tilde{A}^{(1)}$ angewandt. Nach $n-2$ Schritten erhält man $A^{(n-2)}$ mit der Hessenberg-Gestalt, sowie

$$\begin{aligned} A^{(1)} &= S_{(1)} A^{(0)} S_{(1)}^T \\ A^{(2)} &= S_{(2)} S_{(1)} A^{(0)} S_{(1)}^T S_{(2)}^T \\ &\vdots \\ A^{(n-2)} &= \underbrace{S_{(n-2)} \cdots S_{(1)}}_{=Q} A \underbrace{S_{(1)}^T \cdots S_{(n-2)}^T}_{=Q^T}. \end{aligned}$$

Also ist $A^{(n-2)} = QAQ^T$ und für $A = A^T$ gilt weiterhin

$$\left(QAQ^T\right)^T = QA^TQ^T = QAQ^T,$$

d. h. $A^{(n-2)}$ ist symmetrisch. Per Konstruktion ist $A^{(n-2)}$ eine Hessenberg-Matrix, und wenn sie zeitgleich symmetrisch ist, so ist sie eine Tridiagonalmatrix. □

Bemerkung: Für $A \in \mathbb{R}^{n\times n}$ hat diese Konstruktion einer Hessenberg-Normalform einen Aufwand von $\frac{5}{3}\, n^3 + \mathcal{O}(n^2) = \mathcal{O}(n^3)$.

Satz 2.5.13 QR-Zerlegung von Hessenberg-Matrizen

Sei $A \in \mathbb{R}^n$ eine Hessenberg-Matrix. Dann kann die QR-Zerlegung $A = QR$ (und daher auch das QR-Verfahren zur Eigenwert-Bestimmung) mittels Householder-Transformationen in $2n^2 + \mathcal{O}(n)$, also in $\mathcal{O}(n^2)$ arithmetischen Operationen durchgeführt werden. Die Matrix RQ hat wieder Hessenberg-Gestalt.

Beweis zu Satz 2.5.13

(i) Für eine Matrix A in Hessenberg-Normalform gilt $a_{ij} = 0$ für $i > j + 1$, denn

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n-1,n} & a_{nn} \end{pmatrix}.$$

Per Induktion wird gezeigt, dass das für $A^{(k)}$ nach einem Schritt der QR-Zerlegung weiterhin gilt. Dafür reicht es bereits aus, nur den Schritt auf $A^{(1)}$ zu prüfen, weil dieser die genutzten Voraussetzung auf den nächsten Schritt vererbt. Man wähle $v^{(1)} = a_1 + \|a_1\| e_1$ resp. $\operatorname{sign}(a_{11})$. Wegen $a_1 = (a_{11}, a_{21}, 0, \cdots, 0)^T$ gilt $v_k^{(1)} = 0$ für $k > 2$. Für die Spaltenvektoren rechnet man dann

$$a_i^{(1)} = a_i - \left\langle a_i, v^{(1)} \right\rangle v^{(1)}, \quad i = 2, \ldots, n$$

Über die Wahl von $v^{(1)}$ folgert sich $\left(a_i^{(1)}\right)_k = 0$ für $k > i + 1$. Demzufolge hat $A^{(1)}$ Hessenberg-Gestalt, was sich für $i = 2, \ldots, n - 1$ fortsetzt. Damit ist RQ in Hessenberg-Normalform.

(ii) $v^{(i)}$ hat in jedem Schritt zwei von 0 verschiedene Einträge und benötigt daher bloß konstanten Aufwand zur Berechnung. Für $a_i^{(1)}$ benötigt man 4 arithmetische Operationen, und das für $n - i$ Spaltenvektoren, daher ergibt sich ein Gesamtaufwand von

$$\sum_{i=1}^{n-1} \big(2 + 4(n - i)\big) = 2n^2 + \mathcal{O}(n) = \mathcal{O}(n^2). \qquad \square$$

2.6. SVD und Berechnung singulärer Werte

Sei $A \in \mathbb{C}^{m \times n}$ und o. B. d. A. $m \geq n$. Im Folgenden betrachtet man singuläre Werte $\sigma_1, \ldots, \sigma_n$, welche die wesentliche Dynamik eines zugrundeliegenden physikalischen Problems ausdrückt, das durch die Matrix A beschrieben wird. In der Hinsicht sind singuläre Werte also vergleichbar mit klassischen Eigenwerten. Faktisch sind singuläre Werte nichtnegative Wurzeln von Eigenwerten, allerdings von A^*A oder AA^*, wobei $A^* := \overline{A}^T$. Das heißt:

$$\sigma_i = \sqrt{\lambda_i}, \quad \lambda_i \text{ ist EW von } A^*A, \quad i = 1, \ldots, n.$$

Definition 2.6.1 Singular Value Decomposition

Seien $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ die singulären Werte einer Matrix A. Als SVD bezeichnet man die Zerlegung $A = U\Sigma V^*$ mit den folgenden Anforderungen an die Matrizen U, V, Σ:

1. $U \in \mathbb{C}^{m\times m}$ und $V \in \mathbb{C}^{n\times n}$ sind unitär, d. h. $UU^* = U^*U = I$, bzw. $VV^* = V^*V = I$
2. $\Sigma \in \mathbb{R}^{m\times n}$ hat die Gestalt

$$\Sigma = \begin{array}{l} \left.\begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_n \end{pmatrix}\right] n \\ \left.\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}\right] m-n \end{array}$$

Tatsächlich tragen die führenden $\sigma_1, \ldots, \sigma_\gamma$ für $\gamma < n$ bereits den Großteil der Hauptinformationen, wodurch Σ entsprechend reduziert werden kann. Das folgt dann ebenfalls für U und V^*, wodurch man eine Reduktion $\tilde{A} = \tilde{U}\tilde{\Sigma}\tilde{V}^*$ mit $\tilde{A} \in \mathbb{C}^{\tilde{m}\times\tilde{n}}$ und $\tilde{m} \ll m$, $\tilde{n} \ll n$ erhält.

Als Beispiel dazu diene Bildkompression (siehe beispielsweise (Richter, Wahl und Thomas Wick 2024)). Die Pixel eines Bildes werden in einer Matrix A gespeichert. Die führenden $\sigma_1, \ldots, \sigma_\gamma$ der SVD berechnet man mittels

$$\varepsilon(\gamma) = \frac{\sum_{i=1}^{\gamma} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2}.$$

Man bestimmt γ dann derart, dass $\varepsilon(\gamma) > 0.99$, was bedeutet, dass die ersten γ singulären Werte 99% der Gesamtinformationen von A enthalten. Daraus folgt dann eine reduzierte Matrix $\tilde{A}$, welche ohne relevanten Datenverlust komprimierte Bilddaten enthält. Der interessierte Leser kann sich dazu die Polarstern-Bildkompression in (ebd.) anschauen.

Wie zu Beginn kurz erwähnt, sind die singulären Werte $\sigma_1, \ldots, \sigma_n$ von den Eigenwerten von $AA^* \in \mathbb{C}^{m\times m}$ abgeleitet. Der triviale Ansatz wäre also, AA^* zu berechnen, diese Matrix auf Hessenberg-Normalform zu reduzieren und darauf das QR-Verfahren anzuwenden, um die Eigenwerte und darüber dann die singulären Werte zu erhalten. Allerdings ist

$$\operatorname{cond}(AA^*) \leq \operatorname{cond}(A) \cdot \operatorname{cond}(A^*)$$

sehr groß, was zu Rundungsfehlern und Auslöschung führt. Aus numerischer Perspektive ist das also keine ideale Lösung. Es wird also ein anderes Verfahren benötigt, um die singulären Werte zu bestimmen. Das führt zu der Idee, A erst in eine bestimmte Form zu zerlegen, um dann die Eigenwertberechnung auf einer reduzierten Matrix durchzuführen, welche sich wesentlich kostengünstiger durchführen lässt. Das Vorgehen ist ähnlich zu Abschnitt 2.5.5.

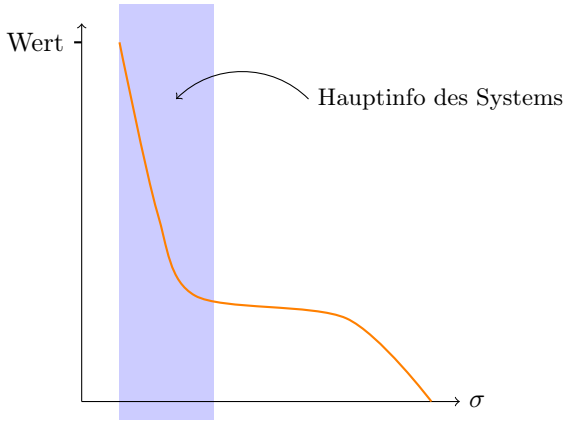


Abbildung 2.5.: Oft beobachteter Abfall der singulären Werte.

2.6.1. Reduktion auf Bidiagonalmatrix

In diesem Abschnitt folgen wir weitestgehend einer berühmten Arbeit (G. Golub und Kahan 1965). Es sei angemerkt, dass weitere Verbesserungen im Algorithmus in (G. H. Golub und Reinsch 1970) vorgeschlagen und letzteres heute noch als Lösungsschema der SVD verwendet wird.

Satz 2.6.2 Golub-Kahan Bidiagonalisierung

Sei $A \in \mathbb{C}^{m\times n}$ eine beliebige Matrix. Dann kan A in $A = PJQ^*$ zerlegt werden, wobei P und Q unitär sind und $J \in \mathbb{C}^{m\times n}$ die Form

$$J = \begin{pmatrix} \alpha_1 & \beta_1 & & 0 \\ 0 & \alpha_2 & \ddots & \\ \vdots & & \ddots & \beta_{n-1} \\ 0 & \cdots & 0 & \alpha_n \\ \hline 0 & & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & 0 \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right] n \\ \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right] m-n \end{matrix}$$

Beweis zu Satz 2.6.2

Sei $A^{(1)} := A$ und seien

$$A^{(3/2)},\ A^{(2)},\ A^{(5/2)},\ldots,A^{(n)},\ A^{(n+\frac{1}{2})}$$

für $k = 1, \ldots, n$ definiert als

$$A^{(k+\frac{1}{2})} = P^{(k)} A^{(k)}, \qquad A^{(k+1)} = A^{(k+\frac{1}{2})} Q^{(k)}.$$

Die Matrizen $P^{(k)}$ und $Q^{(k)}$ werden dabei über Householder-Transformationen berechnet:

$$P^{(k)} = I - 2x^{(k)} {x^{(k)}}^*, \quad {x^{(k)}}^* x^{(k)} = 1 \tag{1}$$

$$Q^{(k)} = I - 2y^{(k)} {y^{(k)}}^*, \quad {y^{(k)}}^* y^{(k)} = 1 \tag{2}$$

Man beachte, dass für die Berechnung der Matrizen das dyadische Produkt der Vektoren, für die Nebenbedingung das Skalarprodukt, genutzt wird. Nun soll $P^{(k)}$ so bestimmt werden, dass $a_{i,k}^{(k+\frac{1}{2})} = 0$ für $i = k+1, \ldots, m$. Für $Q^{(k)}$ soll $a_{k,j}^{(k+1)} = 0$ für $j = 2, \ldots, n$ gelten. D. h. $A^{(k)}$ soll in zwei Schritten in Bidiagonalform transformiert werden:

$$P^{(k)} A^{(k)} = \begin{pmatrix} * \ (k{=}1) & & & & \\ 0 & * \ (k{=}2) & & & \\ \vdots & 0 & \ddots & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & & \ddots \end{pmatrix} \qquad A^{(k+\frac{1}{2})} Q^{(k)} = \begin{pmatrix} * & * & 0 & \cdots & \cdots & 0 \\ 0 & * & * & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & \ddots & & \\ \vdots & \vdots & & \ddots & \ddots & \\ 0 & 0 & & & \ddots & \ddots \end{pmatrix}$$

Die Zielmatrix $A^{(k+1)}$ soll also diese Form haben:

$$\left(\begin{array}{ccccc|ccc} \alpha_1 & \beta_1 & & & & & & \\ & \alpha_1 & \ddots & & & & 0 & \\ & & \ddots & \ddots & & & & \\ & & & \ddots & \beta_{k-1} & & & \\ & & & & \alpha_k & \beta_k & & \\ & & & & & * & \cdots & * \\ & 0 & & & & \vdots & \ddots & \vdots \\ & & & & & * & \cdots & * \\ \hline 0 & \cdots & \cdots & \cdots & 0 & * & \cdots & * \\ \vdots & & & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & * & \cdots & * \end{array}\right)$$

Weil der Fall für $Q^{(k)}$ analog folgt, beschränkt man sich auf die Konstruktion von $P^{(k)}$. Setze

$$x_i^{(k)} = 0, \quad i = 1, \ldots, k-1, \tag{3}$$

denn in $A^{(k+1)}$ sind die dazugehörigen Spalten bereits behandelt worden. Da $P^{(k)}$ unitär ist,

bleibt die Länge erhalten, und daher

$$|\alpha_k|^2 = \sum_{i=k}^{m} \left|a_{i,k}^{(k)}\right|^2. \tag{4}$$

Da $P^{(k)}$ hermitsch ist, gilt

$$A^{(k+\frac{1}{2})} = P^{(k)} A^{(k)} \quad \Leftrightarrow \quad P^{(k)} A^{(k+\frac{1}{2})} = A^{(k)}$$

Wendet man darauf (1) an, so erhält man komponentenweise

- $\left(1 - 2\left|x_k^{(k)}\right|^2\right)\alpha_k = a_{k,k}^{(k)}$ (Diagonale)
- $-2x_i^{(k)}\overline{x}_k^{(k)}\alpha_k = a_{i,k}^{(k)}, \quad i = k+1, \ldots, m$

Äquivalent umgeformt bekommt man:

$$\left|x_k^{(k)}\right|^2 = \frac{1}{2}\left(1 - \frac{a_{k,k}^{(k)}}{\alpha_k}\right) \qquad x_i^{(k)} = \frac{-a_{i,k}^{(k)}}{2\alpha_k \overline{x}_k^{(k)}}$$

Nun kann der Vektor $x^{(k)}$ konstruiert werden. Die ersten Komponenten sind durch (3) bereits bekannt. Aufgrund numerischer Stabilität wählt man $\operatorname{sign}(\alpha_k)$ so, dass $x_k^{(k)}$ möglichst große Werte enthält. In (4) erhält man somit

$$\alpha_k = -\frac{a_{k,k}^{(k)}}{\left|a_{k,k}^{(k)}\right|}\left(\sum_{i=k}^{m}\left|a_{i,k}^{(k)}\right|^2\right)^{\frac{1}{2}}.$$

Zusammenfassend erhält man hieraus

$$A^{(k+\frac{1}{2})} = P^{(k)} A^{(k)} = \left(I - 2x^{(k)}x^{(k)^*}\right)A^{(k)} = A^{(k)} - 2x^{(k)}\left(x^{(k)^*}A^{(k)}\right).$$

Setzt man jetzt

$$s_k := \left(\sum_{i=k}^{m}\left|a_{i,k}^{(k)}\right|^2\right)^{\frac{1}{2}} \qquad c_k := \left(2s_k \frac{a_{k,k}^{(k)}}{\left|a_{k,k}^{(k)}\right|}\, x_k^{(k)}\right)^{-1}$$

dann lässt sich $x^{(k)}$ wie folgt berechnen:

$$x_i^{(k)} = \begin{cases} 0 & : \quad i < k \\ \left[\frac{1}{2}\left(1 + \frac{a_{k,k}^{(k)}}{s_k}\right)\right]^{\frac{1}{2}} & : \quad i = k \\ c_k a_{i,k}^{(k)} & : \quad i > k \end{cases}$$

Auf analoge Weise löst man $A^{(k+1)} = A^{(k+\frac{1}{2})} - 2\big(A^{(k+\frac{1}{2})} y^{(k)}\big) y^{(k)^*}$. Bei Interesse an einen detaillierteren Beweis sei an (G. Golub und Kahan 1965) verwiesen. □

2.6.2. Berechnung der singulären Werte

Das ursprüngliche Problem war $A = U\Sigma V^*$. Jetzt gefunden wurde eine Zerlegung $A = PJQ^*$, es fehlt also noch eine Zerlegung $J = X\Sigma Y^*$, also eine Transformation von einer Bidiagonal- auf eine Diagonalmatrix. Dafür kann etwa das QR-Verfahren genutzt werden, wobei vorher J auf quadratische Form $\tilde{J} \in \mathbb{C}^{n \times n}$ gebracht wird, wo die letzten $n - m$ Zeilen gelöscht werden. In alternativen Methoden kann $\tilde{J} = J^* J$ (Tridiagonalmatrix) im QR-Verfahren genutzt werden. Eine nochmal andere Variante ist

$$\tilde{J} = \begin{pmatrix} 0 & J \\ J^* & 0 \end{pmatrix},$$

was aber erst in Tridiagonalform gebracht werden muss. Dieses alternative Verfahren wird hier aber nicht weiter thematisiert. Zuletzt sei bemerkt, dass aus der eigentlichen Bidiagonalisierung zwar die Eigenwerte bereits auf der Diagonalen abzulesen sind, diese aber noch vorzeichenbehaftet sind. Daher sind die in diesem Abschnitt besprochenen Modifikationen notwendig.

3. Krylow-Unterraum-Verfahren

In diesem Kapitel beschäftigen wir uns mit der iterative Lösung von linearen Gleichungssystemen und Eigenwertproblemen. Erstere sind zwar schon in Numerik I behandelt worden, allerdings aus Modul- und Zeitgründen nicht alle Verfahren. Letztere (EW-Probleme) sind sowieso erst in Numerik II gemacht worden (Kapitel 2). Es gibt eine Klasse von Verfahren mit denen lineare Gleichungssysteme und Eigenwertprobleme beide gelöst werden. Können aus diesem Grund ist es nun sinnvoll ein eigenes Kapitel zu beginnen. Die grundlegende Literatur ist das Lehrbuch von (Saad 2003).

Gegeben seien $A \in \mathbb{R}^{n\times n}$ und $b \in \mathbb{R}^n$. Man möchte nun das Gleichungssystem $Ax = b$ lösen, d. h. $x = A^{-1}b$. Grundlegend möchte man das Problem zunächst auf eine geringere Dimension reduzieren, sucht also eine Lösung in einem Untervektorraum, was über geeignete Bedingungen ein iterativer Prozess wird, der in jedem Schritt den Untervektorraum um eine Dimension erhöht. Heißt, man löst $Ax = b$ mit einer approximierten Lösung x_m aus einem affinen Raum $x_0 + K_m$, worin $\dim(x_0 + K_m) = m$ gilt. Konkret nutzt man

$$Ax = b \quad \Leftrightarrow \quad 0 = b - Ax \quad \Leftrightarrow \quad x = x - Ax + b,$$

woraus sich eine Fixpunkt-Iteration herleiten lässt:

$$x_{m+1} = x_m - Ax_m + b = (I - A)x_m + b$$

Führt man die Iteration nun mit Startwert x_0 durch und setzt $r_0 := b - Ax_0$, so folgt:

$$\begin{aligned} m = 0: \quad & x_1 = (I - A)x_0 + x_0 && = x_0 + r_0 \\ m = 1: \quad & x_2 = (I - A)(x_0 + r_0) + x_0 && = x_0 + 2r_0 - Ar_0 \\ & \vdots \end{aligned}$$

Die Näherungen x_m zur Lösung des Gleichungssystems $x = A^{-1}b$ sind Linearkombinationen der Vektoren r_0, Ar_0, A^2r_0, …, $A^{k-1}r_0$ um x_0 geshiftet. Den Vektorraum, welcher diese Elemente als Basisvektoren enthält, nennt man Krylow-Raum:

$$K_m(A, r_0) := \{\, r_0,\, Ar_0,\, A^2r_0,\, \ldots,\, A^{m-1}r_0 \,\}, \quad \dim(K_m) = m.$$

Damit ist also $x_m \in x_0 + K_m(A, r_0)$. Obwohl der Ursprung der Krylow-Räume, und insbesondere

des Krylow-Verfahrens, in der Berechnung der Koeffizienten charakteristischer Polynome liegt, werden Krylow-Raum-Operationen eher zur Lösung von $Ax = b$ genutzt und stellt ein Black-Box-Verfahren dar, das sich durch einfache Implementierung und Robustheit auszeichnet. Die Fixpunktiteration zur Herleitung ist natürlich nicht die einzige Möglichkeit, um eine bessere Näherungslösung $x_m \in x_0 + K_m$ zu finden. Allgemein nutzt man dafür orthogonale Projektionen, die als Petrov-Galerkin-Bedingungen bekannt sind:

$$\underbrace{b - Ax_m}_{\text{Defekt/Residuum}} \perp L_m$$

Dabei bezeichnet L_m einen weiteren m-dimensionalen Unterraum. Nutzt man nun ein bestimmtes Verfahren, um eine Näherungslösung per orthogonaler Projektion zu erhalten, so können Ergebnisse trotzdem durch Wahl von L_m variieren. Ein gutes Beispiel dafür sind FOM und GMRES, die in den Kapiteln 3.2 und 3.6 behandelt werden. Es gibt zwei Klassen zur Wahl von L_m: einmal $L_m = K_m$ bzw. $L_m = AK_m$. Die zweite Klasse ist $L_m = K_m(A^T, r_0)$, wo mit der transponierten Matrix A^T gearbeitet wird. Letztere betrachten wir in diesem Kurs nicht und verweisen stattdessen auf Saad 2003[Kapitel 7].

3.1. Arnoldi-Verfahren

Das Arnoldi-Verfahren von 1951 ist neben dem Lanczos-Verfahren eines der ersten Krylow-Raum-Verfahren, welches iterativ zur Bestimmung von $Ax = b$ und von Eigenwerten und dazugehörigen Eigenvektoren dient. Arnoldi selbst ist noch nicht das Lösungsschema sondern zunächst wird in diesem Verfahren zu einer gegebenen Matrix $A \in \mathbb{C}^{n\times n}$ und einem Startvektor $q \in \mathbb{C}^n$ eine orthonormale Basis des dazugehörigen Krylow-Raums bestimmt. Der Algorithmus liest sich:

Algorithmus 8 (Arnoldi)**.**

1. *Wähle Startvektor* $v_1 \in \mathbb{C}^n$ *mit* $\|v_1\|_2 = 1$
2. *Für* $j = 1, \ldots, m$:
3. *Für* $i = 1, \ldots, j$:
4. $h_{ij} = \langle Av_j, v_i \rangle$
5. $w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
6. $h_{j+1,j} = \|w_j\|_2$
7. *Falls* $h_{j+1,j} = 0 \rightarrow$ *Stopp*
8. $v_{j+1} = \frac{w_j}{h_{j+1,j}}$

In den Schritten (3) und (4) werden Einträge einer Hessenberg-Matrix (siehe auch Abschnitt 2.5.5) berechnet. Schritt (5) ist der Gram-Schmidt-Schritt, über den nach Schritt (6) ein Orthonormalbasisvektor vorliegen sollte. Durch dieses Verfahren erhält man also eine Faktorisierung der ursprünglichen Matrix A durch $AV = VH$ mit der orthogonalen Matrix $V = (v_1, \ldots, v_n)$ und der Hessenberg-Matrix $H = (h_{ij})_{j=1}^n$. D. h.

$$AV = (Av_1, \ldots, Av_n) = (v_1, \ldots, v_n) \begin{pmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & \cdots & h_{2n} \\ & h_{32} & \ddots & & \\ & & \ddots & \ddots & \\ & & & h_{n,n-1} & h_{nn} \end{pmatrix}$$

Satz 3.1.1

Angenommen, es wird bis m orthonormalisiert, d. h. die Abbruchbedingung in Schritt (7) tritt nicht ein, dann bilden die Vektoren $v_1, \ldots, v_n$ aus dem Algorithmus eine Orthonormalbasis des Krylow-Raums

$$K_m = \text{span}\{\, v_1,\, Av_1,\, \ldots,\, A^{m-1}v_1 \,\}.$$

Beweis zu Satz 3.1.1

Per Konstruktion sind die v_j orthonormal für $j = 1, \ldots, m$. Es soll gezeigt werden, dass jeder Vektor v_j von der Form $q_{j-1}(A)v_1$ ist, wobei q_{j-1} ein Polynom vom Grad $j-1$ darstellt, also

$$q_{j-1}(A) = a_0 + a_1 A + \cdots + a_{j-1}A^{j-1}, \quad a_i \in \mathbb{R}, \quad i = 1, \ldots, j-1.$$

Daraus würde dann folgen, dass der Krylow-Raum K_m durch die v_j aufgespannt wird, weil diese linear unabhängig sind (folgt aus Orthogonalität) und Linearkombinationen der $A^{j-1}v_j$ wären. Per Austauschlemma von Steinitz folgte dann das Ergebnis. Dass die v_j die gewünschte Form haben, wird induktiv gezeigt. Mit $q_0(A) = 1$ hat man den Induktionsanfang $v_1 = q_0(A)v_1$. Den Induktionsschritt schließt man aus

$$\begin{aligned} h_{j+1,j}v_{j+1} = h_{j+1,j}\frac{w_j}{h_{j+1,j}} &= Av_j - \sum_{i=1}^{j} h_{ij}v_i = Aq_{j-1}(A)v_1 - \sum_{i=1}^{j} h_{ij}q_{i-1}(A)v_1 \\ &= \underbrace{\left(Aq_{j-1}(A) - \sum_{i=1}^{j} h_{ij}q_{i-1}(A)\right)}_{=:\, h_{j+1,j}\, q_j(A)} v_1. \end{aligned}$$

Somit kann v_{j+1} als $q_j(A)v_1$ dargestellt werden, womit sich der Beweis schließt. □

Satz 3.1.2

Sei $V_m \in \mathbb{R}^{n\times m}$ mit den durch den Arnoldi-Algorithmus erzeugten Spaltenvektoren $v_1, \ldots, v_m$ mit $v_i \in \mathbb{R}^n, i = 1, \ldots, m$. Des Weiteren sei $\overline{H}_m \in \mathbb{R}^{(m+1)\times m}$ die Hessenbergmatrix mit den Koeffizienten h_{ij} und es sei $H_m \in \mathbb{R}^{m\times m}$ die Matrix aus $\overline{H}_m$, wo die letzte Zeile gelöscht wird. Zuletzt sei w_m aus Schritt 5 des vorherigen Arnoldi-Algorithmus. Dann gelten die folgenden Relationen:

$$AV_m = V_m H_m + w_m e_m^T \tag{3.1}$$
$$= V_{m+1}\overline{H}_m \tag{3.2}$$
$$V_m^T A V_m = H_m \tag{3.3}$$

Beweis zu Satz 3.1.2

(3.2) folgt direkt aus dem Arnoldi-Algorithmus:

$$Av_j = \sum_{i=1}^{j+1} h_{ij} v_i, \quad j = 1, \ldots, m$$

(3.1) ergibt sich aus Aufspalten von $V_{m+1}\overline{H}_m$ und (3.3) ist rückblickend zu der Konstruktion einer Hessenberg-Normalform in Golub-Kahan (Abschnitt 2.6) klar. □

Algorithmus 9 (Arnoldi MGS[1])**.**

1. *Wähle Startvektor* $v_1 \in \mathbb{R}^n$ *mit* $\|v_1\|_2 = 1$
2. *Für* $j = 1, \ldots, m$:
3. $\quad w_j = Av_j$
4. $\quad$ *Für* $i = 1, \ldots, j$:
5. $\quad\quad h_{ij} = \langle w_j, v_i \rangle$
6. $\quad\quad w_j = w_j - h_{ij} v_i$
7. $\quad h_{j+1,j} = \|w_j\|_2$
8. $\quad$ *Falls* $h_{j+1,j} = 0 \rightarrow$ *Stopp*
9. $\quad v_{j+1} = \dfrac{w_j}{h_{j+1,j}}$

[1] Modified Gram-Schmidt; Indem bestimmte sich wiederholende Prozesse aus den Schleifen herausgezogen und die Reihenfolge einiger Operationen geändert werden, erhält man wesentlich höhere numerische Stabilität. Arnoldi-Householder wäre nochmal stabiler als Arnoldi MGS.

3.2. FOM

Letztlich ist FOM (Full Orthogonalization Method) das Lösen von $Ax = b$ mittels Arnoldi, wofür $L_m = K_m(A, v_0)$ benutzt wird. Das heißt die Galerkin-Bedingung ist

$$b - Ax_m \perp K_m$$

Man setzt für einen Startwert $x_0 \in \mathbb{R}^n$:

$$r_0 = b - Ax_0, \quad v_1 = \frac{r_0}{\|r_0\|_2}, \quad \beta = \|r_0\|_2$$

Aus Satz 3.1.2 ist $V_m^T A V_m = H_m$ bekannt und außerdem gilt $V_m^T r_0 = V_m^T(\beta v_1) = \beta e_1$, da V_m^T orthogonal (hermitsch) ist. Aus der dritten Bedingung in Satz 3.1.2, i.e., $V_m^T A V_m = H_m$ folgt somit

$$x_m = x_0 + V_m H_m^{-1} V_m^T r_0 = x_0 + V_m H_m^{-1}(\beta e_1). \tag{3.4}$$

Daraus erhält man die approximierte Lösung $x_m \in x_0 + K_m$ anhand eines entkoppelten Schemas:

$$x_m = x_0 + V_m y_m, \quad y_m = H_m^{-1}(\beta e_1).$$

Es sei bemerkt, dass die Inverse H_m^{-1} kostengünstig berechnet werden kann, da H_m eine Hessenberg-Matrix ist.

Algorithmus 10 (FOM)**.**

1. $r_0 = b - Ax_0, \quad \beta = \|r_0\|_2, \quad v_1 = \dfrac{r_0}{\beta}$
2. $H_m = \{\, h_{ij} \,\}_{i,j=1,\dots,m} \in \mathbb{R}^{m \times m}$ *(auf Null initialisiert)*
3. **Für** $j = 1, \dots, m$:
4. $\quad w_j = A v_j$
5. $\quad$ **Für** $i = 1, \dots, j$:
6. $\quad\quad h_{ij} = \langle w_j, v_i \rangle$
7. $\quad\quad w_j = w_j - h_{ij} v_i$
8. $\quad h_{j+1,j} = \|w_j\|_2$
9. $\quad$ **Falls** $h_{j+1,j} = 0 \;\rightarrow\;$ **Setze** $j = m \;\rightarrow\;$ **goto** 11.
10. $\quad v_{j+1} = \dfrac{w_j}{h_{j+1,j}}$
11. $y_m = H_m^{-1}(\beta e_1), \quad x_m = x_0 + V_m y_m$

Jetzt verbleibt die Frage, wie gut m ist? Soll heißen, ab wann gilt $\|b - Ax_m\| < \text{TOL}$, also

$b - Ax \approx b - Ax_m$? Der nachfolgende Satz gibt uns Relationen an die Hand, womit die aktuelle Approximationsgüte zum Schritt m bestimmt werden kann.

Satz 3.2.1

Für $x_m \in \mathbb{R}^n$ berechnet mit der FOM gilt

(i) $b - Ax_m = -h_{m+1,m}\, e_m^T\, y_m\, v_{m+1}$

(ii) $\|b - Ax_m\|_2 = h_{m+1,m}\, |e_m^T\, y_m|$

Beweis zu Satz 3.2.1
Es gilt

$$b - Ax_m = b - A(x_0 + V_m y_m) = b - Ax_0 - AV_m y_m = r_0 - AV_m y_m.$$

Nach Satz 3.1.2 folgt hieraus

$$= r_0 - (V_m H_m + w_m\, e_m^T)\, y_m \overset{\text{FOM}}{=} \beta v_1 - V_m H_m y_m - v_{m+1}\, h_{m+1,m}\, e_m^T\, y_m. \qquad (1)$$

Ebenfalls nach dem FOM-Algorithmus gilt

$$y_m = H_m^{-1}(\beta e_1) \quad \Leftrightarrow \quad H_m y_m = \beta e_1$$

und somit ergibt sich

$$\beta v_1 - V_m H_m y_m = \beta v_1 - V_m \beta e_1 = \beta v_1 - \beta v_1 = 0,$$

woraus in (1) eingesetzt bereits (i) folgt:

$$b - Ax_m = -h_{m+1,m}\, e_m^T\, y_m\, v_{m+1}$$

Weil v_{m+1} orthonormal ist, gilt $\|v_{m+1}\|_2 = 1$ und per Konstruktion ist $h_{m+1,m} > 0$, also folgt (ii) wie folgt:

$$\|b - Ax_m\|_2 = \|\underbrace{-h_{m+1,m}\, e_m^T\, y_m}_{\in\mathbb{R}}\, v_{m+1}\|_2 = |\underbrace{-h_{m+1,m}}_{>0}\, e_m^T\, y_m| \cdot \underbrace{\|v_{m+1}\|_2}_{=1} = h_{m+1,m}\, |e_m^T\, y_m| \qquad \square$$

Durch diesen Satz ist die rechte Seite mittels der FOM vollständig berechenbar, womit Schranken für $b - Ax_m$ bzw. dessen Norm angegeben werden können.

Satz 3.2.2 Aufwand der FOM

Die FOM zur Erstellung der H_m und damit Lösung von $Ax = b$ benötigt $\mathcal{O}(m^2 n)$ arithmetische Operationen.

Beweis zu Satz 3.2.2

Betrachte den FOM-Algorithmus. Für die äußere Schleife werden zur Berechnung der w_j insgesamt $m \cdot n$ arithmetische Operationen benötigt. In der inneren Schleife werden i Operationen für h_{ij} und $i \cdot n$ Operationen für die Anpassung von w_j benötigt. Danach werden in der äußeren Schleife noch $h_{j+1,j}$ und v_{j+1} mit je n Operationen bestimmt. Daraus ergibt sich

$$\sum_{\substack{j=1 \\ \uparrow \\ \text{äußere Schleife}}}^{m} \Bigg(\sum_{\substack{i=1 \\ \uparrow \\ \text{innere Schleife}}}^{j} (i + in) + m \cdot n + n + n \Bigg) \approx \sum_{j=1}^{m} (m + mn) + mn + 2jn$$

$$\approx m^2 + m^2 n + m^2 n + 2mn \approx 2m^2 n = \mathcal{O}(m^2 n). \qquad \square$$

Für $m = n$, sprich wenn FOM auf das gesamte System der Dimension n angewendet wird, hat man wie bei der LR- und QR-Zerlegung kubischen Aufwand $\mathcal{O}(n^3)$. In der Praxis wird daher $m \ll n$ gewählt, wobei gleichzeitig eine hinreichende Güte sichergestellt werden sollte, vgl. Satz 3.2.1.

3.3. (Symmetrischer) Lanczos-Algorithmus

Der Lanczos-Algorithmus ist eine Anwendung des Arnoldi-Algorithmus für symmetrische Matrizen. Er dient zur Erstellung der Hessenberg-Matrizen und unitären Transformationen im Zuge der Orthonormalisierung des Krylow-Raums K_m.

Satz 3.3.1 Tridiagonalität von H_m

Für den Arnoldi-Alogirthmus mit $A = A^T$ sind die Koeffizienten h_{ij} derart, dass H_m tridiagonal und insbesondere symmetrisch ist.

Beweis zu Satz 3.3.1

Es gilt $H_m = V_m^T A V_m$ und somit ist H_m symmetrisch, weil $A = A^T$ symmetrisch und V_m unitär ist. Per Konstruktion ist H_m eine Hessenberg-Matrix, und aufgrund der Symmetrie muss H_m bereits tridiagonal sein. $\square$

Für das Lanczos-Verfahren ist es üblich, die Tridiagonalmatrix wie folgt zu bezeichnen:

$$H_m =: T_m = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \beta_2 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_m \\ 0 & \cdots & 0 & \beta_m & \alpha_m \end{pmatrix}$$

Algorithmus 11 (Lanczos)**.**

1. *Wähle* v_1 *mit* $\|v_1\|_2 = 1$
2. $\beta_1 = v_0 = 0$
3. *Für* $j = 1, 2, \ldots, m$:
4. $\quad w_j = Av_j - \beta_j v_{j-1}$
5. $\quad \alpha_j = \langle w_j, v_j \rangle$
6. $\quad w_j = w_j - \alpha_j v_j$ *//Gram-Schmidt Orthogonalisierung*
7. $\quad \beta_{j+1} = \|w_j\|$
8. $\quad$ *Falls* $\beta_{j+1} = 0 \rightarrow$ *Stopp*
9. $\quad v_{j+1} = \dfrac{w_j}{\beta_{j+1}}$

Der Lanczos-Algorithmus ist sehr effizient, da die 3-stufige Rekursion in den Schritten (4)–(6) gut genutzt werden kann. In Schritt (9) gilt nämlich

$$\beta_{j+1} v_{j+1} = w_j = Av_j - \beta_j v_{j-1} - \alpha_j v_j.$$

Lanczos führt nun auf zwei Schemata. Einmal zur Eigenwertberechnung und zum CG-Verfahren (Kapitel 3.5), das zur Lösung von $Ax = b$ genutzt wird.

3.4. Lanczos zur Eigenwertberechnung

Konzept

1. Gegeben sie $A \in \mathbb{R}^{n \times n}$.
2. Konstruiere Krylov-Raum K_m mit Dimension $m \leq n$.
3. Orthogonalisiere K_m mittels Arnoldi oder für symmetrische Matrizen mit Lanczos.
4. Nutze orthogonale Ähnlichkeitstransformation $H_m = V_m^T A V_m$ mit $V_m^T V_m = I$. Damit besitzen H_m und A dieselben Eigenwerte.
5. Wende QR-Verfahren auf H_m an.

Wie bereits bekannt, wird in der Praxis zumeist $m \ll n$ gewählt, weil häufig nur die Hauptinformationen des Systems relevant sind. Entsprechend wichtig ist zu wissen, welche Eigenwerte mit einer Iteration bis $m \ll n$ berechnet werden.

Satz 3.4.1

Das QR-Verfahren angewendet auf H_m liefert die Approximation der ersten m Eigenwerte der ursprünglichen Matrix A.

Der interessierte Leser kann den Beweis in (Parlett 1998), Chapter 12 “Krylov Subspaces” nachlesen. Fehlerabschätzungen und gute Approximationen der ersten m Eigenwerte sind im selben Buch in Chapter 12.4 “The Error Bounds of Kaniel and Saad” zu finden.

Satz 3.4.2

Es seien $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ die Eigenwerte der Matrix $A \in \mathbb{R}^{n\times n}$ mit $A = A^T$. Des Weiteren seien

$$\mu_1^{(m)} \geq \mu_2^{(m)} \geq \ldots \geq \mu_m^{(m)}$$

die Eigenwerte der Hessenberg-Matrix $H_m = V_m^T A V_m$. Dann gelten für $1 \leq j \leq m$ die beiden Ungleichungsketten

$$\text{(i)} \quad \lambda_{n-j+1} \leq \mu_{m+1-j+1}^{(m+1)} \leq \mu_{m-j+1}^{(m)}$$
$$\text{(ii)} \quad \mu_j^{(m)} \leq \mu_j^{(m+1)} \leq \lambda_j$$

D. h. der j-kleinste Eigenwert von H_m konvergiert monoton von oben gegen den j-kleinsten Eigenwert von A und der j-größte Eigenwert von H_m konvergiert monoton von unten gegen den j-größten Eigenwert von A.

Die Abschätzungen sind rein qualitativ und ohne Güte, demnach geben diese Ungleichungsketten keinen Aufschluss über die Konvergenzgeschwindigkeit. Hierzu dient dann der Satz von Parlett (1998), den sich der interessierte Leser in (Parlett 1998) anschauen kann. Für den Beweis der vorliegenden Aussage werden zuvor noch zwei Hilfmittel benötigt.

Satz 3.4.3 Courant-Fischer

Sei $A \in \mathbb{C}^{n\times n}$ hermitsch, $\{ z_1, \ldots, z_n \} \in \mathbb{C}^n$ eine Orthonormalbasis. Seien $\lambda_1 \geq \ldots \geq \lambda_n$ die Eigenwerte von A mit Eigenvektoren x_i für $i = 1, \ldots, n$. Dann gilt

$$\min_{0\neq x\in Z_m} \frac{x^* A x}{x^* x} \leq \lambda_m, \quad Z_m := \operatorname{span}\{ z_1, \ldots, z_m \}.$$

Für $Z_m = \operatorname{span}\{ x_1, \ldots, x_m \}$ gilt sogar Gleichheit.

Definition 3.4.4 Rayleigh-Quotient

Sei zu einer Matrix A die Eigenwertgleichung $Aw = \lambda w$ gegeben. Dann folgt

$$w^T Aw = w^T \lambda w \quad \Leftrightarrow \quad \langle Aw, w\rangle_2 = \lambda \|w\|_2^2.$$

Falls w bekannt ist, so heißt

$$\lambda = \frac{\langle Aw, w\rangle}{\|w\|_2^2}$$

der zu w assoziierte Rayleigh-Koeffizient. Die durch den Quotienten gegebene Funktion heißt Rayleigh-Quotient und bildet jeden Eigenvektor auf seinen dazugehörigen Eigenwert ab.

Man erinnere sich daran, dass durch $|\lambda| \leq \|A\|$ die gröbste Eigenwert-Abschätzung gegeben ist. Diese folgt auch aus dem Rayleigh-Koeffizienten:

$$|\lambda| \leq \sup_{w\neq 0} \frac{\langle Aw, w\rangle}{\|w\|_2^2} \leq \sup_{w\neq 0} \frac{\|A\| \cdot \|w\|_2^2}{\|w\|_2^2} = \|A\|.$$

Andererseits liefert der Satz von Courant-Fischer (3.4.3) einen Rayleigh-Koeffizienten als untere Schranke an λ_m. Das führt zu dem sogenannten Min-Max-Prinzip: Die Aufgabe, den größten bzw. kleinsten Eigenwert von A zu finden ist gleichbedeutend damit, den größten bzw. kleinsten Rayleigh-Quotienten zu finden. Damit ist jetzt alles beisammen, um Satz 3.4.2 zu beweisen:

Beweis zu Satz 3.4.2
Sei $\{ y_i \}$ eine Orthonormalbasis aus Eigenvektoren zu H_m mit Eigenwerten $\mu_i^{(m)}$, $i = 1, \ldots, m$. Nun ist $\{ z_i \}$ für $z_i = V_m y_i$ ebenfalls eine Orthonormalbasis, weil V_m orthogonal (hermitsch) ist. Der Beweis wird in drei Schritten geschehen: Zuerst wird gezeigt, dass die Folge $\mu_j^{(m)}$ durch λ_j beschränkt ist. Danach wird die Monotonie der Folge gezeigt und zuletzt, dass das Ergebnis für $\mu_{m+1-j+1}^{(m)}$ umgekehrt ist.

(i) Man setze $\mu_j^{(m)}$ auf seine Darstellung als Rayleigh-Koeffizient, d. h.

$$\mu_j^{(m)} = \min_{y\in Y_j} \frac{y^T H_m y}{y^T y} \quad \text{mit} \quad Y_j = \operatorname{span}\{ y_1, \ldots, y_j \}.$$

Das Minimum kommt daher, dass $\mu_j^{(m)}$ in Y_j der kleinste Eigenwert ist, also der Rayleigh-Quotient sein Minimum in diesem Eigenwert annimmt. Nun nutze man $z = V_m y$. Daraus ergibt sich

$$z^T z = y^T \underbrace{V_m^T V_m}_{=I} y = y^T y$$

Damit erhält man

$$\mu_j^{(m)} = \min_{y \in Y_j} \frac{y^T H_m y}{y^T y} = \min_{y \in Y_j} \frac{y^T V_m^T A V_m y}{y^T y} = \min_{z \in Z_j} \frac{z^T A z}{z^T z}$$

und aus Satz 3.4.3 folgt somit $\mu_j^{(m)} \leq \lambda_j$.

(ii) Für diesen Schritt wird y in der Darstellung von $\mu_j^{(m)}$ "gehoben", also in seiner Dimension um 1 erhöht. Konkret passiert das durch $\hat{y} = \binom{y}{0} \in \mathbb{R}^{m+1}$. Damit gilt

$$\begin{array}{l} V_m\, y = V_{m+1}\, \hat{y}, \quad \hat{y}^T \hat{y} = y^T y. \\ \uparrow \\ \in \mathbb{R}^{n \times m} \end{array}$$

Setzt man das nun in die Darstellung von $\mu_j^{(m)}$ ein, so resultiert

$$y_j^{(m)} = \min_{y \in Y_j} \frac{y^T H_m y}{y^T y} = \min_{y \in Y_j} \frac{y^T V_m^T A V_m y}{y^T y} = \min_{\hat{y} \in \hat{Y}_j} \frac{\hat{y}^T V_{m+1}^T A V_{m+1} \hat{y}}{\hat{y}^T \hat{y}} \leq \mu_j^{(m+1)}.$$

(iii) Man betrachte die Matrix $-A$. Für diese gilt $-\lambda_n \geq \cdots \geq -\lambda_1$ und somit

$$V_m^T(_A) V_m : \quad -\mu_m^{(m)} \geq \cdots \geq -\mu_1^{(m)}.$$

Gemäß der Ergebnisse aus den Schritten (i) und (ii) gilt somit für $-A$:

$$-\mu_{m-j+1}^{(m)} \leq \cdots \leq -\mu_{m+1-j+1}^{(m+1)} \leq -\lambda_{n-j+1}$$

Daraus folgt die Behauptung. □

Wie anhand des Satzes von Courant-Fischer bemerkbar ist, gilt das Resultat aus Satz 3.4.2 auch auf $\mathbb{C}^{n \times n}$ mit hermitschen Matrizen, nicht nur für symmetrische reelle Matrizen.

3.5. CG-Verfahren

Das CG-Verfahren (Conjugate Gradients) ist eine Anwendung von FOM für symmetrische Matrizen, d. h. ein lineares Gleichungssystem $Ax = b$ wird über Arnoldi gelöst. Nach Satz 3.3.1 ist die Hessenbergmatrix H_m aus dem Arnoldi-Verfahren für symmetrische Matrizen eine Tridiagonalmatrix T_m. Gemäß FOM löst man nach der approximierten Lösung

$$x_m = x_0 + V_m y_m, \quad y_m = T_m^{-1}(\beta e_1).$$

Hierbei ist T_m^{-1} selbst mit Gauß-Verfahren relativ effizient berechenbar. Man erhält daraus den

Algorithmus 12 (Lanczos für lineare Systeme)**.**

1. $r_0 = b - Ax_0, \quad \beta = \|r_0\|_2, \quad v_1 = \dfrac{r_0}{\beta}$
2. **`Für`** $j = 1, 2, \ldots, m$:
3. $\quad w_j = Av_j - \beta_j v_{j-1}$ $\qquad$ $//Für\ j = 1{:}\ \beta_1 v_0 = 0$
4. $\quad \alpha_j = \langle w_j, v_j \rangle$
5. $\quad w_j = w_j - \alpha_j v_j$
6. $\quad \beta_{j+1} = \|w_j\|_2$
7. $\quad$ **`Falls`** $\beta_{j+1} = 0 \ \rightarrow\ m := j,$ **`goto`** 9
8. $\quad v_{j+1} = \dfrac{w_j}{\beta_{j+1}}$
9. $T_m = \text{tridiag}(\beta_i, \alpha_i, \beta_{i+1}), \quad V_m = (v_1, \ldots, v_m)$
10. **`Löse`** $y_m = T_m^{-1}(\beta e_1), \quad x_m = x_0 + V_m y_m$

Natürlich sollte m "gut" gewählt werden. Häufig wird mit Größenordnungen von $n = 10^3, \ldots, 10^8$ gearbeitet. Erfahrungsgemäß wählt man hier $m = 10^2, \ldots, 10^3$, wobei diese Wahl ab $n = 10^5$ vorkonditionierte Systeme $P^{-1}Ax = P^{-1}b$ mit $P^{-1} \approx A^{-1}$ benötigt.

3.5.1. Zweite Herleitung CG-Verfahren

In der Einleitung zu Krylow-Unterraum-Verfahren wurde erwähnt, dass häufig orthogonale Projektionen genutzt werden, um bessere Näherungslösungen zu berechnen. Grundlage ist also die Petrov-Galerkin-Bedingung $b - Ax_m \perp L_m$ bzw.

$$\langle b - Ax_m, d_j \rangle = 0 \quad \forall j = 0, \ldots, m-1. \qquad \text{(Galerkin-Gleichung)}$$

Legt man $r_0 = d_0 = b - Ax_0$ fest, erhält man den Krylow-Raum

$$K_m(A, r_0) = \text{span}\{\, r_0, Ar_0, \ldots, A^{m-1}r_0 \,\}.$$

Die orthogonale Projektion soll jetzt auf dem Krylow-Raum selbst gebildet werden, also setzt man $L_m = K_m$. Die zweite Herleitung des CG-Verfahrens basiert nun auf dem aus Numerik I bekannten Gradientenverfahren

$$x_{j+1} = x_j + \underset{\substack{\uparrow \\ \text{Suchlänge}}}{\alpha_j} \underset{\substack{\uparrow \\ \text{Suchrichtung}}}{d_j}.$$

Die neue Suchrichtung soll dabei nicht im Erzeugnis $\text{span}\{\, d_0, \ldots, d_{m-1} \,\}$ der bisherigen Suchrichtungen liegen.

Lemma 3.5.1

Es gelte $A^m d_0 \in K_m$. Dann liegt $x_m \in \mathbb{R}^n$ von $Ax_m = b$ im m-ten Krylow-Raum K_m.

Beweis zu Lemma 3.5.1

Sei K_m aufgebaut und $x_m \in x_0 + K_m$ diejenige (beste) Approximation, die die Orthogonalitätsbedingung erfüllt. Dann ist

$$r_m = b - Ax_m = \underbrace{b - Ax_0}_{=d_0} + A\underbrace{(x_0 - x_m)}_{\in K_m} \in d_0 + AK_m = K_{m+1}.$$

Wegen $A^m d_0 \in K_m$ gilt $K_{m+1} \subseteq K_m$ und somit $r_m \in K_m$. Per Konstruktion erfüllt x_m die Galerkin-Gleichung, also $r_m \perp K_m$. Weil aber $r_m \in K_m$ gilt, muss $r_m = 0$ sein, was $Ax_m = b$ impliziert. Damit liegt die Lösung x_m des linearen Gleichungssystems in K_m. □

Als nächstes müssen α_j und d_j berechnet werden. Dies geschieht letzlich mit dem Arnoldi-Verfahren, bei der die Orthogonalisierung durch (modified) Gram-Schmidt ausgeführt wird. Nochmal effizienter geschieht das mit Hilfe von Rekursionsformeln. Gemäß dem Gradientenverfahren lässt sich x_m als

$$x_m = x_0 + \sum_{i=0}^{m-1} \alpha_i d_i$$

darstellen. Über die Galerkin-Gleichung folgt hieraus

$$\langle b - Ax_m, d_j \rangle = \left\langle b - A\left(x_0 + \sum_{i=0}^{m-1} \alpha_i d_i \right), d_j \right\rangle \stackrel{!}{=} 0.$$

Wegen der Orthogonalität der d_j gilt $\langle d_i, d_j \rangle = 0$ und somit resultiert

$$\langle b - Ax_0, d_j \rangle = \alpha_j \langle Ad_j, d_j \rangle \quad \Rightarrow \quad \alpha_j = \frac{\langle b - Ax_0, d_j \rangle}{\langle Ad_j, d_j \rangle}.$$

Um die Berechnung der Skalarprodukte im Zuge der Orthogonalisierung effizienter zu gestalten, wird eine 2-stufe Rekursion genutzt.

Lemma 3.5.2

Sei A symmetrisch positiv definit, $x_0 \in \mathbb{R}^n$, $d_0 = b - Ax_0$. Dann wird durch die Iteration

$$r_m = b - Ax_m, \quad \beta_{m-1} := -\frac{\langle r_m, Ad_{m-1}\rangle}{\langle d_{m-1}, Ad_{m-1}\rangle}, \quad d_m := r_m + \beta_{m-1} d_{m-1}$$

für $m = 1, 2, \ldots$ eine A-orthogonale Basis mit $\langle Ad_r, d_s\rangle = 0$ für $r \neq s$ erzeugt.

Beweis zu Lemma 3.5.2

Sei $\{ d_0, \ldots, d_{m-1} \}$ eine A-orthogonale Basis des K_m. Ferner seien $x_m \in x_0 + K_m$ und $r_m = b - Ax_m \in K_{m+1}$ mit $r_m \neq 0$. Der Fall $r_m = 0$ liefert nach dem vorherigen Lemma ein Abbruchskriterium der Iteration. Die Richtungen d_m werden mit dem Ansatz

$$d_m = r_m + \sum_{j=0}^{m-1} \beta_j^{m-1} d_j$$

berechnet. Die Orthogonalität besagt nun für $i = 0, \ldots, m-1$:

$$0 = \langle d_m, Ad_i\rangle = \langle r_m, Ad_i\rangle + \sum_{j=1}^{m-1} \beta_j^{m-1} \langle d_j, Ad_i\rangle = \langle r_m, Ad_i\rangle + \beta_i^{m-1} \langle d_i, Ad_i\rangle$$

Wegen $Ar_m \perp K_{m-1}$ gilt damit also

$$\langle r_m, Ad_i\rangle = \langle b - Ax_m, Ad_i\rangle = 0$$

für $i = 0, \ldots, m-2$, und somit insbesondere auch $\beta_i^{m-1} = 0$. Für $i = m-1$ erhält man hingegen

$$\beta_{m-1}^{m-1} = -\frac{\langle r_m, Ad_{m-1}\rangle}{\langle d_{m-1}, Ad_{m-1}\rangle}$$

Setzt man das nun in den Ansatz für d_m ein, so ergibt sich

$$d_m = r_m + \beta_{m-1} d_{m-1}.$$ □

Damit ist jetzt fast alles gefunden, um das CG-Verfahren zu beschreiben. Man wählt einen Startwert $x_0 \in \mathbb{R}^n$, setzt $d_0 = r_0 = b - Ax_0$ als Start-Residuum, dann nach Lemma 3.5.2

$$\beta_{m-1} = -\frac{\langle r_m, Ad_{m-1}\rangle}{\langle d_{m-1}, Ad_{m-1}\rangle}, \quad d_m = r_m + \beta_{m-1} d_{m-1}$$

und schlussendlich sucht man die Schrittlänge α_m. Dafür starte man mit

$$r_m = b - Ax_m \quad \Rightarrow \quad \langle r_m, d_m\rangle = \langle b - Ax_0 + Ax_0 - Ax_m, d_m\rangle = \langle \underbrace{b - Ax_0}_{=d_0} + A\underbrace{(x_0 - x_m)}_{\in K_m}, d_m\rangle$$

und setzt darin $x_{m+1} = x_0 + \sum_{i=0}^{m} \alpha_i d_i$ ein, woraus

$$\langle b - Ax_0 + A(x_0 - x_m), d_m\rangle = \alpha_m \langle Ad_m, d_m\rangle$$

mit $\langle d_j, d_m\rangle = 0$ für $j \neq m$ folgt. Dann ist

$$\langle r_m, d_m\rangle = \alpha_m \langle Ad_m, d_m\rangle \quad \Rightarrow \quad \alpha_m = \frac{\langle r_m, d_m\rangle}{\langle Ad_m, d_m\rangle}.$$

Als neues Residuum erhält man also

$$r_{m+1} = b - Ax_{m+1} = b - Ax_m - \alpha_m Ad_m = r_m - \alpha_m Ad_m$$

und man erhält den

Algorithmus 13 (CG-Verfahren)**.**

1. $x_0 \in \mathbb{R}^n, \quad r_0 = d_0 = b - Ax_0$
2. ***Für*** $m = 0, 1, 2, \ldots :$
3. $\quad \alpha_m = \dfrac{\langle r_m, d_m\rangle}{\langle Ad_m, d_m\rangle}$
4. $\quad x_{m+1} = x_m + \alpha_m d_m$
5. $\quad r_{m+1} = r_m - \alpha_m Ad_m$
6. $\quad \beta_m = -\dfrac{\langle r_{m+1}, Ad_m\rangle}{\langle d_m, Ad_m\rangle}$
7. $\quad d_{m+1} = r_{m+1} + \beta_m d_m$

Korollar 3.5.3

Das CG-Verfahren mit gegebener symmetrisch positiv definiter $A \in \mathbb{R}^{n\times n}$ bricht nach spätestens n Schritten ab. Das heißt prinzipiell (bei exakter Arithmetik) ist das CG-Verfahren ein direktes Verfahren.

Das folgt unmittelbar aus Lemma 3.5.1.

Satz 3.5.4

Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit. Dann gilt die Fehlerabschätzung

$$\|x_m - x\|_A \leq 2\underbrace{\left(\frac{1-\frac{1}{\sqrt{\kappa}}}{1+\frac{1}{\sqrt{\kappa}}}\right)}_{=:\varrho}{}^m \|x_0 - x\|_A, \quad m = 1, 2, 3, \ldots$$

mit $\kappa = \mathrm{cond}_2(A) = \frac{|\lambda_{\max}|}{|\lambda_{\min}|}$ und

$$\langle x, x\rangle_A := \langle Ax, x\rangle_2, \quad \|x\|_A = \sqrt{\langle Ax, x\rangle_2}.$$

Sind also $|\lambda_{\max}| \approx |\lambda_{\min}|$, dann ist $\kappa \approx 1$ und somit $\varrho \approx 0$, was schnelle Konvergenz impliziert. Ist hingegen $|\lambda_{\max}| \gg |\lambda_{\min}|$, dann ist $\kappa \to \infty$ und demnach $\varrho \approx 1$, womit die Fehlerabschätzung sehr schlecht ist. In dem Fall ist eine gute Vorkonditionierung notwendig!

3.6. GMRES-Verfahren

Das GMRES-Verfahren (Generalized Minimal RESidual) arbeitet, anders als das CG-Verfahren, nicht notwendigerweise mit symmetrischen Matrizen und nicht notwendigerweise positif definit. Daher ist diese Verfahren eine Verallgemeinerung. Deshalb anders sind die Wahl von $L_m = AK_m$ und die Nutzung der $\|\cdot\|_2$-Norm, statt der $\|\cdot\|_A$-Norm aus dem CG-Verfahren. Die Idee ist hierbei die Minimierung des Residuums in $\|\cdot\|_2$, was einige Schwierigkeiten erzeugt. Hierfür orientiert man sich am Ansatz aus der FOM in Kapitel 3.2, worin ein $x \in x_0 + K_m$ über $x = x_0 + V_m y$ erzeugt wurde. Definiere

$$J(y) := \|b - Ax\|_2 = \|b - A(x_0 + V_m y)\|_2.$$

Idee ist hier, das Minimum von $J(y)$ zu finden, denn für $\|b - Ax\|_2 \approx 0$ gilt $Ax \approx b$. Mit den Bezeichnungen

$$r_0 = b - Ax_0, \quad v_1 = \frac{r_0}{\|r_0\|_2}, \quad \beta = \|r_0\|_2$$

aus der FOM und durch Satz 3.1.2 (vgl. Seite 135) folgt

$$b - Ax = b - A(x_0 + V_m y) = r_0 - AV_m y = \beta v_1 - V_{m+1}\overline{H}_m y = V_{m+1}(\beta e_1 - \overline{H}_m y).$$

Aufgrund der Konstruktion mit Arnoldi sind die Spaltenvektoren in V_{m+1} orthonormal, d. h.

$$J(y) = \|b - Ax\|_2 = \|\beta e_1 - \overline{H}_m y\|_2, \quad \|V_{m+1}\| = 1.$$

Man erhält also insgesamt die Iteration

$$x_m = x_0 + V_m y_m, \quad y_m = \underset{y}{\operatorname{argmin}} \|\beta e_1 - \overline{H}_m y\|_2.$$

Die Hauptarbeit liegt dabei in der Lösung von y_m. Da $\overline{H}_m$ rechteckig mit $(m+1) \times m$ ist, liegt der Lösungsansatz bei einem Least-Squares-Problem. Eine erste Möglichkeit zur Lösung ist eine Normalengleichung. Für $B \in \mathbb{R}^{n \times m}$ mit $Bx = b$, $x \in \mathbb{R}^m$ und $b \in \mathbb{R}^n$ hat man $B^T Bx = B^T b$ als Normalengleichung. Für relativ große m, n gilt jedoch

$$\operatorname{cond}(B^T B) \leq \operatorname{cond}(B^T) \cdot \operatorname{cond}(B) = \operatorname{cond}(B)^2,$$

die Normalengleichung ist also sehr schlecht konditioniert. Hier liegt das Problem $\overline{H}_m y = \beta e_1$ vor. Die dazugehörige Normalengleichung wäre dann

$$\overline{H}_m^T \overline{H}_m y = \overline{H}_m^T \beta e_1.$$

Die Matrix $\overline{H}_m^T \overline{H}_m$ ist symmetrisch positiv definit, kann also numerisch stabil Cholesky-zerlegt werden. In der Praxis möchte man aber eher, dass alle Vorgänge robust und möglichst effizient ablaufen, daher gibt es einen anderen Lösungsansatz, nämlich die QR-Zerlegung für rechteckige Matrizen.

Satz 3.6.1 QR-Zerlegung rechteckiger Matrizen

Sei $B \in \mathbb{R}^{n \times m}$ mit $n > m$, $\operatorname{rang}(B) = m$. Dann existiert eine orthogonale Matrix $V \in \mathbb{R}^{n \times n}$ sowie eine rechteckige obere Dreiecksmatrix $\tilde{R} \in \mathbb{R}^{n \times m}$, sodass $B = V\tilde{R}$ mit

$$\tilde{R} = \begin{pmatrix} R \\ \hline 0 \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} R \\ \hline 0 \end{matrix}}\right\} m \\ \left.\vphantom{\begin{matrix} R \\ \hline 0 \end{matrix}}\right\} n-m \end{matrix}, \quad R = \begin{pmatrix} * & & & * \\ 0 & * & & \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & 0 & * \end{pmatrix} \in \mathbb{R}^{m \times m}$$

Die Matrix R entspricht hierbei der rechten oberen Dreiecksmatrix aus der klassischen QR-Zerlegung. Für einen Beweis dieses Satzes sei an (Richter und Thomas Wick 2017), Seite 118, verwiesen. Man erhält insgesamt den

Algorithmus 14 (GMRES-Verfahren)**.**

1. $x_0 \in \mathbb{R}^n, \quad r_0 = b - Ax_0, \quad \beta = \|r_0\|_2, \quad v_1 = \dfrac{r_0}{\beta}$
2. ***Für*** $j = 1, 2, \ldots, m$:

3. $w_j = Av_j$
4. **Für** $i = 1, \ldots, j$:
5. $h_{ij} = \langle w_j, v_j \rangle$
6. $w_j = w_j - h_{ij} v_i$
7. $h_{j+1,j} = \|w_j\|_2$
8. **Falls** $h_{j+1,j} = 0 \rightarrow m := j$, **goto** 10
9. $v_{j+1} = \dfrac{w_j}{h_{j+1,j}}$
10. **Baue** $(m+1) \times m$ **Hessenbergmatrix** $\overline{H}_m$
11. **Löse** $\underset{y}{\operatorname{argmin}} \|\beta e_1 - \overline{H}_m y\|_2$
12. **Löse** $x_m = x_0 + V_m y_m$

A. Kurzfragen

A.1. Kurzfragen Kapitel 1

1. Was ist eine lineare DGL? Man gebe auch ein Beispiel an.
2. Was ist eine nicht-lineare DGL? Man gebe ebenfalls ein Beispiel an.
3. Was ist ein gekoppeltes DGL-System?
4. Was beschreibt die Ordnung einer DGL?
5. Welche (praxisrelevanten; alltäglichen) Phänomene können mit gewöhnlichen DGL beschrieben werden?
6. Man gebe die analytische Lösung zur Modell-DGL $u'(t) = a\,u(t)$, $u(0) = 1$ an.
7. Welche numerischen Methoden zur Lösung von DGL haben wir im Zuge der Vorlesung kennengelernt?
8. Warum muss man überhaupt DGL mit Hilfe von numerischen Methoden lösen?
9. Was ist ein Galerkin-Ansatz zur Lösung von DGL?
10. Was ist ein Differenzenansatz zur Lösung von DGL?
11. Man beschreibe die wesentlichen Unterschiede von Galerkin-Verfahren zu Differenzenverfahren zur Lösung von DGL.
12. Was ist das implizite Eulerverfahren? Welche charakteristischen Eigenschaften weist dieses Verfahren auf?
13. Was sind Mehrschrittmethoden? Wieso sind Mehrschrittmethoden im Vergleich zu Einschrittmethoden überhaupt interessant?
14. Wozu dient das Gronwallsche Lemma?
15. Was besagt der Abschneidefehler?
16. Was ist eine Stabilitätsabschätzung?

17. Was folgt aus Konsistenz und Stabilität für die in der Vorlesung untersuchten Einschrittmethoden?

18. Was sind Runge-Kutta-Verfahren?

19. Man beschreibe A-Stabilität.

20. Wozu braucht man A-Stabilität?

21. Wozu dient adaptive Schrittweitensteuerung?

22. Wie kann numerische Stabilität untersucht werden? Was sind charakteristische Eigenschaften, wenn ein Verfahren nicht numerisch stabil ist?

23. Wozu dienen Eigenwerte?

24. Wie können Eigenwerte numerisch berechnet werden?

25. Man gebe das Stabilitätsintervall des expliziten Eulerverfahrens an.

26. Man gebe das Stabilitätsintervall der Trapezregel an.

27. Wie können implizite Verfahren gelöst werden?

28. Was sind Prädiktor-Korrektor-Verfahren?

29. Wozu werden Fixpunktverfahren bzw. Newtonverfahren im Zuge der Lösung von Differentialgleichungen benötigt?

30. Gegeben sei $u'(t) = a\,u^2(t)$, $u(0) = 10$. Man löse dieses Problem mit der Trapezregel und stelle hierzu das zugehörige Newtonverfahren auf.

31. Gegeben sie $u'(t) = a\,u^2(t) + b\,u^5(t),\ u(t_0) = u_0$. Löse mit Trapezregel und stelle entsprechendes Newtonschema auf.

32. Was ist eine differentiell-algebraische Gleichung (DAE)?

33. Was beschreibt der Index einer DAE?

34. Wozu dienen die Gerschgorin-Kreise?

35. Man beschreibe die Kernidee der Potenzmethode

36. Was ist das QR-Verfahren?

37. Richtig oder falsch: Das implizite Eulerverfahren hat die Ordnung 2.

38. Was bedeutet eigentlich, dass ein numerisches Verfahren die Ordnung α; bzw. man beschreibe alternativ, wie sich die Ordnungen 1 oder 2 in der Praxis äußern.

Lösungen:

1. Eine lineare DGL ist dann gegeben, wenn die gesuchte Funktion $u(t)$ lediglich in linearen Abhängigkeiten auftritt. Beispiele: $u'(t) = a\,u(t)$, $u'(t) = at\,u(t)$, $u'(t) = at^2\,u(t)$.

2. Hier treten die Ableitung $u'(t)$ bzw. $u(t)$ in nicht-linearen Zusammenhängen auf. Beispiele: $u'(t) = a\,u^2(t)$, $u'(t) = a\,u^2(t) + bu^7(t)$, $u(t)\,u'(t) = a\,u(t)$, $u(t)\,u'(t) = a\,u^3(t)$.

3. Ein DGL-System besteht aus mindestens zwei DGLen, deren Lösungen sich gegenseitig beeinflussen.

4. Die Ordnung ist durch die höchste Ableitungsstufe gegeben.

5. Ein paar Beispiele:
   a) Ausbreitung von Krankheiten
   b) Bevölkerungswachstum
   c) Finanzwissenschaften, Börsenkurse
   d) Zinses-Zins-Effekte
   e) Schwingungen (Federpendel)
   f) Entwicklung von Bakterienkulturen

6. $u(t) = u_0 \exp\big(a\,(t - t_0)\big)$. Mit $t_0 = 0$ und $u_0 = u(t_0) = 1$ erhält man $u(t) = \exp(at)$.

7. Differenzenverfahren (Einschritt, Mehrschritt), Galerkin-Verfahren, sukzessive Approximation

8. Methoden der Analysis (analytische Lösungen) sind oft nicht möglich, da DGL dafür zu kompliziert sein können, z. B. durch Nichtlinearitäten, DGL-Systemen oder gekoppelten DGL.

9. Multiplikation der DGL mit einer sogenannten Testfunktion aus einem geeigneten Funktionenraum und anschließender Integration.

10. Approximation der Ableitung der DGL durch einen Differenzenquotienten

11. Differenzenverfahren approximieren immer die Ableitung(en) der DGL, während Galerkin-Verfahren integrale Darstellungen mit Hilfe von sogenannten Testfunktionen aus entsprechenden Funktionenräumen nutzen.

12. Gegeben sei $u'(t) = f(t, u)$. Das implizite Euler-Verfahren schreibt sich dann

$$\frac{y_n - y_{n-1}}{h} = f(t_n, y_n).$$

    Eigenschaften: 1. Ordnung in h, A-stabil (gute Stabilitätseigenschaften)

13. Approximation der Lösung y_{n+1} durch mehrere vorangegangene Lösungen $y_n, y_{n-1}, \ldots$, wodurch eine höhere Ordnung des numerischen Verfahrens erzielt wird. Mehrschrittmethoden sind interessant, weil dabei nicht mit höheren Ableitungen der rechten Seite der DGL (Taylor, Runge-Kutta) gearbeitet werden muss.

14. Stabilitätsabschätzung

15. Das ist der lokale Diskretisierungsfehler, welchen man erhält, wenn die exakte Lösung u in das numerische Verfahren eingesetzt wird. Hierdurch erhält man die lokale Konsistenzordnung.

16. Eine Stabilitätsabschätzung schätzt den Fehler, z. B. $\|y_n - u(t_n)\|$, durch Konstanten und prinzipiell berechenbare Größen der rechten Seite ab. D. h. hierdurch erhält man eine Aussage, inwiefern sich der Fehler verhält, wenn sich gewisse Daten des Problems (DGL) wie in etwa Anfangswerte oder Koeffizienten verändern. Wünschenswert sind kleine Fehleränderungen, wenn sich die Daten nur leicht ändern.

17. Aus Konsistenz[1] und Stabilität[2] folgt die Konvergenz eines (linearen) numerischen Verfahrens zur Lösung einer DGL (siehe auch S. 39).

18. Runge-Kutta-Verfahren sind Einschrittmethoden höherer Ordnung.

19. Das Gebiet absoluter Stabilität ist definiert durch

$$SG = \big\{ z = ah \in \mathbb{C} \;\big|\; 0 \leq |\omega(z)| \leq 1 \big\},$$

wobei ω die Stabilitätsfunktion bezeichnet. A-Stabilität liegt vor, wenn

$$\big\{ z = ah \in \mathbb{C} \;\big|\; \mathfrak{Re}(z) \leq 0 \big\} \subset SG$$

gilt.

20. Wenn besonders gute numerische Stabilitätseigenschaften benötigt werden, z. B. bei DGL-Systemen mit sehr großen Koeffizienten (sogenannte steife Probleme).

21. Kleinere Schrittweiten in Bereichen, wo sich die Lösung $u(t)$ der DGL stark ändert, um diese damit genauer approximieren zu können.

22. Nehme Modellproblem, z. B. $u'(t) = a\,u(t)$, $u(t_0) = u_0$, und untersuche die exponentiell abklingenden Anteile, hier $a < 0$. Ein stabiles numerisches Verfahren sollte die abklingende kontinuierliche Lösung so gut wie möglich approximieren. D. h. gilt $\sup\limits_{t>t_0} \|u(t)\| < \infty$, dann

[1] auch lokaler Abschneidefehler; der Fehler, wenn die exakte Lösung in das numerische Verfahren eingesetzt wird.
[2] Fortpflanzung bisheriger Fehler auf zukünftige Fehler.

sollte $\max\limits_{n\geq 0} |y_n| < \infty$ gelten. In anderen Worten: Für die diskrete Lösung sollte gelten:

$$|y_{n+1}| \leq |y_n| \leq \ldots \leq |y_0|$$

Insbesondere muss dann bei expliziten Verfahren die Schrittweite $h = t_n - t_{n-1}$ geeignet gewählt werden, sodass Stabilität garantiert werden kann. Die kritischen Schrittweiten erhält man durch die Untersuchung des sogenannten Verstärkungsfaktors. Implizite Verfahren erfüllen automatisch die Bedingungen des Verstärkungsfaktors und bedingen (oft) keiner Schrittweitenbedingung. Die in diesem Sinne besten Verfahren sind die sogenannten A-stabilen Verfahren. In numerischen Simulationen äußern sich Instabilitäten häufig durch unphysikalische Oszillationen in numerischen Lösungen (siehe auch S. 54/55).

23. Bestimmung von Eigenfrequenzen bei Schwingungen, Stabilitätsverhalten von DGLen für sehr große Zeiten

24. Direkte Methode: Bestimme Eigenwerte direkt aus dem charakteristischen Polynom.

    Iterative Verfahren: Potenzmethode, Iteration mittels QR-Verfahren, Lanczos-Verfahren

25. $SI = \left\{ z \in \mathbb{R} \;\middle|\; -2 \leq z \leq 0 \right\}$

26. $SI = \left\{ z \in \mathbb{R} \;\middle|\; -\infty < z \leq 0 \right\}$

27. Fixpunktverfahren oder Newton

28. Berechne mit Hilfe eines expliziten Verfahren gute Startwerte für ein darauffolgendes implizites Verfahren.

29. Bei impliziten Verfahren kann oft nicht explizit nach der aktuellen Lösung y_{n+1} aufgelöst werden. D. h. mittels Fixpunkt oder Newton wird dann die Lösung y_{n+1} iterativ bestimmt (siehe auch Abschnitt 1.6.6).

30. Aus der Trapezregel

    $$\frac{y_{n+1} - y_n}{h} = \frac{1}{2}\big[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\big]$$

    erhält man das Nullstellenproblem

    $$g(y_{n+1}) := y_{n+1} - y_n - \frac{h}{2}\big[a\, y_n^2 + a\, y_{n+1}^2\big] = 0.$$

    Durch $g'(y_{n+1}) = 1 - ha\, y_{n+1}$ erhält man den Defekt-Schritt

    $$(1 - ha\, y_{n+1}^{(k)})w = -\Big(y_{n+1}^{(k)} - y_n - \frac{h}{2}\big[a\, y_n^2 + a(y_{n+1}^{(k)})^2\big]\Big)$$

und den entsprechenden Korrektur-Schritt

$$y_{n+1}^{(k+1)} = y_{n+1}^{(k)} + w.$$

31. Schritt 1:

$$\frac{y_n - y_{n-1}}{h} = \frac{1}{2}\left[a\, y_n^2 + b\, y_n^5 + a\, y_{n-1}^2 + b\, y_{n-1}^5\right]$$

Schritt 2:

$$g(y_n) = y_n - y_{n-1} - \frac{h}{2}\left[a\, y_n^2 + b\, y_n^5 + a\, y_{n-1}^2 + b\, y_{n-1}^5\right]$$

Schritt 3:

$$g'(y_n) = 1 - \frac{h}{2}\left[2a\, y_n + 5b\, y : n^4\right]$$

Schritt 4: Newton-Defekt-Korrektur: Für $k = 1, 2, \ldots$

$$g'(y_n^{(k)})w = \; - g(y_n^{(k)})$$

$$y_n^{(k+1)} = y_n^{(k)} + w$$

32. Eine DAE besteht aus mindestens zwei Gleichungen, von welchen mindestens eine Gleichung lediglich in algebraischer Form gegeben ist, sprich ohne Ableitung.

33. Anzahl der Umformungen, um die Ableitung der algebraischen Variablen explizit darstellen zu können.

34. Kriterium zur Eingrenzung der Eigenwerte einer Matrix A, beispielsweise können hieraus Startwerte für die iterativen Verfahren zur Eigenwertbestimmung gewonnen werden.

35. Approximiere größten Eigenwert einer Matrix mittels einer Matrix-Vektor-Multiplikation und entsprechenden Normierungen des neuen iterierten Vektors.

36. Die QR-Zerlegung ist ein direktes Verfahren zur Lösung linearer Gleichungssysteme $Ax = b$. Insbesondere ist Q eine orthogonale Matrix mit $Q^T Q = I$.

    Alternative 1: Zur Lösung von Eigenwertproblemen wird die QR-Zerlegung als innerer Löser im Zuge eines iterativen Verfahrens genutzt.

    Alternative 2: Das QR-Verfahren bestimmt $A_k = Q_k R_k$ (QR-Zerlegung) und berechnet dann $A_{k+1} = R_k Q_k$. Für $k \to \infty$ stehen dann die gesuchten Eigenwerte auf der Diagonalen von A_{k+1}.

37. Falsch: Die Ordnung ist 1.

38. Die Ordnung wird oft durch die Landau-Symbole dargestellt, z. B. $\|y_n - u(t_n)\| = \mathcal{O}(n^2)$. Wenn die Schrittweite h halbiert wird, dann viertelt sich der Fehler (quadratische Konvergenz). Andererseits hat man bei linearer Konvergenz $\mathcal{O}(h)$, d. h. bei Halbierung der Schrittweite wird der Fehler ebenfalls halbiert. In der Praxis sieht man die Konvergenzordnung entweder durch genaues hinschauen (siehe Tabelle auf Seite 53) oder man berechnet die Konvergenzordnung durch die Formel

$$\alpha = \frac{1}{|\log(2)|} \cdot \log\left(\left|\frac{y_h - y_{\frac{h}{2}}}{y_{\frac{h}{2}} - y_{\frac{h}{4}}}\right|\right).$$

A.2. Kurzfragen Kapitel 2 und 3

1. Man formuliere das Eigenwertproblem. Man gebe Beispiele weshalb EW wichtig sind.
2. Was sind geometrische Lokalisierungen von EW? (EW = Eigenwerte)
3. Man leite die gröbst mögliche Lokalisierung her. Was ist der Rayleigh-Koeffizient?
4. Was sind die Gerschgorin-Kreise?
5. Was besagt der Satz von Bauer-Fike?
6. Ist das EW-Problem gut oder schlecht konditioniert?
7. Was für grundsätzlich-mögliche Methoden zur EW-Berechnung gibt es?
8. Erhalten wir mit diesen Methoden dann auch die EV (EV = Eigenvektoren)?
9. Was tut die Potenzmethode nach Richard von Mises?
10. Man erläutere die grundlegende Idee der Potenzmethode nach Richard von Mises.
11. Man gebe einen Vorteil und einen Nachteil der Potenzmethode nach Richard von Mises.
12. Was tut die inverse Iteration nach Wieland?
13. Was ist ein Shift?
14. Zu welcher Klasse von Verfahren gehoert das QR-Verfahren?
15. Was ist eine orthogonale Matrix?
16. Was ist eine unitäre Matrix?
17. Was ist eine hermitsche Matrix?
18. Was ist eine symmetrische Matrix?
19. Ja oder nein: EW zu symmetrischen Matrizen sind stets reell.

20. Ja oder nein: EV in symmetrischen Matrizen sind stets orthogonal.
21. Ja oder nein: EW zu hermitschen Matrizen sind stets reell.
22. Ja oder nein: EV in hermitschen Matrizen sind stets orthogonal.
23. Sei Q orth. Man zeige $\|Qx\|_2 = \|x\|_2$
24. Was ist die QR-Zerlegung?
25. Was ist der Unterschied zwischen QR-Zerlegung und LR-Zerlegung?
26. Wie können Basen in ONB (Orthonormalbasen) überführt werden?
27. Was tut das Householder-Verfahren?
28. Man definiere die Householder-Transformation
29. Man zeige $S = S^T$ und $S^{-1} = S^T$ anhand eines Beispiels.
30. Man erläutere grob die Idee des Householder-Verfahrens.
31. Was ist der Aufwand der QR-Zerlegung?
32. Ja oder nein: Die QR-Zerlegung kann sogar auf rechteckige Matrizen erweitert werden.
33. Warum gibt es verschiedene Verfahren innerhalb der QR-Zerlegung um die ONB zu bauen?
34. Man erläutere das QR-Verfahren zur EW-Berechnung
35. Was ist der grundlegende Algorithmus des QR-Verfahrens?
36. Ja oder nein: mit dem QR-Verfahren werden alle EW approximiert. Man begründe die Antwort kurz.
37. Was ist eine Ähnlichkeitstransformation?
38. Was ist die Idee einer vorherigen Reduktion, um dann das QR-Verfahren anzuwenden?
39. Warum ist diese Vorgehensweise interessant?
40. Welche Reduktionsmethoden gibt es?
41. Was ist die Hessenberg-Normalform?
42. Wie sieht die Matrix-Struktur der Hessenberg-Normalform aus fuer allgemeines A aus?
43. Wie sieht die Matrix-Struktur der Hessenberg-Normalform aus fuer symmetrisches A aus?
44. Wie können die singulären Werte einer rechteckigen Matrix berechnet werden?
45. Man erläutere Theorem 1 aus Golub/Kahan

46. Warum muss die Bidiagonalmatrix J nochmals transformiert werden, um die singulären Werte zu erhalten?
47. Wozu dienen Krylowraumverfahren?
48. Was sind die Unterschiede zu LR/QR, Richardson, Jacobi-Verfahren, Gradientenverfahren?
49. Was ist die grundlegende Iteration aller iterativen Verfahren?
50. Was ist die Idee Krylowräume zu konstruieren?
51. Man erläutere das Arnoldi-Verfahren und gebe die wichtigsten Schritte an.
52. Weshalb gibt es (wie bei der QR-Zerlegung) verschiedene Varianten von Arnoldi?
53. Richtig oder falsch: mit Krylowraumverfahren können nur EW-Probleme berechnet werden.
54. Was tut die FOM?
55. Was ist der Aufwand der FOM?
56. Was ist die Hauptschwierigkeit innerhalb der FOM, sodass oft nur m « n Schritte verwendet werden und bei größerem m das gesamte Verfahren einen "Re-Start"benötigt?
57. Was tut der Lanczos-Algorithmus?
58. Was ist die Idee des Lanczos-Verfahrens zur EW-Berechnung?
59. Was ist die Idee des CG-Verfahrens?
60. Was ist der Unterschied zum Gradientenverfahren?
61. Was ist die (Petrov-) Galerkin-Gleichung?
62. Was ist der Unterschied zwischen FOM, GMRES, CG?
63. Obwohl FOM, GMRES, CG prinzipiell direkte Verfahren sind, werden diese nahezu ausschließlich als iterative Verfahren eingesetzt. Warum?
64. Wann sind FOM, GMRES, CG direkte Verfahren?
65. Was sind Normalengleichungen?
66. Wann treten Normalengleichungen auf?
67. Was ist Kondition von Normalengleichungen?
68. Was ist eine symmetrisch, positiv, definitive Matrix?
69. Man definiere die Konditionszahl
70. Weshalb arbeitet man bei GMRES mit der euklidischen Norm und beim CG-Verfahren mit der A-Norm?

Literatur

Amstutz, S. und T. Wick (Sep. 2022). *Refresher course in maths and a project on numerical modeling done in twos*. Hannover : Institutionelles Repositorium der Leibniz Universität Hannover, DOI: https://doi.org/10.15488/11629. DOI: https://doi.org/10.15488/11629.

Arndt, Daniel, Wolfgang Bangerth, Bruno Blais u. a. (2021). „The deal.II Library, Version 9.3“. In: *Journal of Numerical Mathematics* 29.3, S. 171–186. DOI: 10.1515/jnma-2021-0081. URL: https://dealii.org/deal93-preprint.pdf.

Arndt, Daniel, Wolfgang Bangerth, Denis Davydov u. a. (2021). „The deal.II finite element library: Design, features, and insights“. In: *Computers & Mathematics with Applications* 81, S. 407–422. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2020.02.022. URL: https://arxiv.org/abs/1910.13247.

Bangerth, W., R. Hartmann und G. Kanschat (2007). „deal.II – a General Purpose Object Oriented Finite Element Library“. In: *ACM Trans. Math. Softw.* 33.4, S. 24/1–24/27.

Braun, Martin (1983). *Differential Equations and Their Applications*. 4. Aufl. Springer-Verlag New York. ISBN: 9780387978949.

Ciarlet, Philippe G. (21. Nov. 2013). *Linear and Nonlinear Functional Analysis with Applications*. Society for Industrial and Applied Mathematics. ISBN: 1611972582.

Dahlquist, Germund G. (1963). In: *BIT Numerical Mathematics* 3: *A special stability problem for linear multistep methods*, S. 27–43. ISSN: 0006-3835.

Golub, G. und W. Kahan (1965). „Calculating the Singular Values and Pseudo-Inverse of a Matrix“. In: *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis* 2.2, S. 205–224. ISSN: 0887459X. URL: http://www.jstor.org/stable/2949777 (besucht am 05. 09. 2024).

Golub, G. H. und C. Reinsch (1970). „Singular value decomposition and least squares solutions“. In: *Numerische Mathematik* 14.5. ISSN: 0029-599X.

Hanke-Bourgeois, Martin (11. Dez. 2008). *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. 3. Aufl. Vieweg+Teubner Verlag. ISBN: 9783834807083.

Parlett, Beresford N. (1998). Society for Industrial and Applied Mathematics. ISBN: 9781611971163.

Quarteroni, A. und P.Gervasio (2020). *A Primer on Mathematical Modelling*. UNITEXT. Springer.

Quarteroni, A., F. Saleri und P.Gervasio (2014). *Scientific computing with MATLAB and Octave*. Texts in computational science and engineering. Springer.

Rannacher, Rolf (2. Juni 2017). *Numerik gewöhnlicher Differentialgleichungen*. Heidelberg University Publishing. ISBN: 9783946054320.

Richter, Thomas, Henry von Wahl und Thomas Wick (5. Nov. 2024). *Begriffe, Konzepte und zahlreiche Anwendungsbeispiele*. 2. Aufl. Springer Spektrum Berlin, Heidelberg. ISBN: 9783662695814.

Richter, Thomas und Thomas Wick (2017). *Einführung in die Numerische Mathematik. Begriffe, Konzepte und zahlreiche Anwendungsbeispiele*. Springer Spektrum. ISBN: 9783662541784.

Saad, Yousef (2003). *Iterative methods for sparse linear systems*. SIAM, Society for Industrial und Applied Mathematics; ISBN: 9780898715347.

Strang, G. (2010). *Wissenschaftliches Rechnen*. Springer.

Wick, T. (Jan. 2022). *Numerical methods for partial differential equations*. Hannover : Institutionelles Repositorium der Leibniz Universität Hannover, DOI: https://doi.org/10.15488/11709. DOI: https://doi.org/10.15488/11709.

Index