S E M I N A R A R B E I T

Nichtlineare Eigenwertprobleme

Die Konturintegral-Methode

ausgeführt am

Institut für
Analysis und Scientific Computing
TU Wien

unter der Anleitung von

Lothar Nannen

durch

Richard Weiss und Florian Schager

Wien, am 22. Januar 2021

Inhaltsverzeichnis

1 Einleitung

Wir beschäftigen uns in dieser Arbeit mit der Lösung nichtlinearer Eigenwertprobleme. Die herkömmliche (lineare) Formulierung des Eigenwert-Problems dürfte bekannt sein: Sei $A \in \mathbb{C}^{N\times N}$ eine Matrix. Suche $\lambda \in \mathbb{C}$ und $v \in \mathbb{C}^N \setminus \{0\}$, sodass $Av = \lambda v$, oder äquivalent

$$B(\lambda)v = 0, \quad B(\lambda) := A - \lambda I_N. \tag{1.1}$$

Dabei bezeichnet $\ker B(\lambda)$ den Eigenraum von λ. In manchen Anwendungen, beispielweise bei der Diskretisiserung nichtlinearer Differentialgleichungen, ist die Funktion B jedoch nicht mehr linear oder gar polynomiell. Herkömmliche Algorithmen zur Suche von Eigenwerten sind jedoch bloß für lineare B konzipiert.
Die wohl naheliegendste Variante einer nichtlinearen Eigenwert-Suche ist eine Nullstellen-Suche des nichtlinearen charakteristischen Polynoms $\lambda \mapsto \det B(\lambda)$. Bereits bei linearen Eigenwertproblemen kann man folgende Nachteile erkennen:

1. Die Berechnung der Determinante ist (insbesondere für große Matrizen) überaus kostspielig und numerisch instabil.
2. Wir bekommen bloß eine Nullstelle (Eigenwert).
3. Selbst wenn ein vergleichsweise effizienter Algorithmus wie das Newton-Verfahren zur Nullstellen-Suche verwendet wird, wird immer noch ein geeigneter Startwert benötigt.

Die, in [1] vorgeschlagene Konturintegral-Methode wird auf elegante Weise all diese Probleme umgehen. Die Grundidee des Algorithmus liegt darin, mittels Konturintegralen und dem Satz von Keldysh, das nichtlineare Eigenwertproblem auf ein lineares zu reduzieren. Dieses kann dann mit bekannten Methoden, wie dem QR-Verfahren, gelöst werden. Der Beweis dieses Satzes ist nicht Thema dieser Arbeit. Wir beschäftigen uns stattdessen ausführlicher mit der Herleitung und Analyse der Konturintegral-Methode. Dabei orientieren wir uns größtenteils an [2].
Abschließend werden wir noch eine Anwendung des Algorithmus auf das „Hadeler Problem“ aus [3] präsentieren.

2 Theoretische Grundlagen

In diesem Kapitel werden wir uns mit dem Satz von Keldysh auseinandersetzen. Dieser bildet nämlich die theoretische Basis für unser Verfahren. Bevor wir starten benötigen wir noch die folgende Begriffsbildung.

Definition 2.0.1. Seien $A \in \mathbb{C}^{N\times N}$, $\lambda \in \mathbb{C}$, und $v \in \mathbb{C}^N$ mit $Av = \lambda v$. Wir nennen (λ, v) ein *Rechts-Eigenpaar* von A, v einen *Rechts-Eigenvektor* von A zum Eigenwert λ, und $\ker(A - \lambda I_N)$ den *Rechts-Eigenraum* von λ.
Seien $B \in \mathbb{C}^{N\times N}$, $\mu \in \mathbb{C}$ und $w \in \mathbb{C}$, mit $w^*B = \mu w^*$. Wir nennen (μ, w) ein *Links-Eigenpaar* von B, w einen *Links-Eigenvektor* von B zum Eigenwert μ, $\ker(B^* - \overline{\mu} I_N)$ den *Links-Eigenraum* von μ.

Bemerkung 2.0.2. Wir schließen direkt an die Definition 2.0.1 an. Der Rechts-Eigenvektor v von A zum Eigenwert λ ist auch Links-Eigenvektor von A^* zum Eigenwert $\overline{\lambda}$, weil

$$v^*A^* = (Av)^* = (\lambda v)^* = \overline{\lambda} v^*.$$

Der Links-Eigenvektor w von B zum Eigenwert μ ist auch Rechts-Eigenvektor von B^* zum Eigenwert $\overline{\mu}$, weil

$$B^*w = (w^*B)^* = (\mu w^*)^* = \overline{\mu} w.$$

Daher ist die Definition von $\ker(B^* - \overline{\mu} I_N)$ als Links-Eigenraum von μ sinnvoll.
Sollte A hermitesch (selbstadjungiert) sein, so sind alle Eigenwerte reell, und damit alle Rechts-Eigenvektoren auch Links-Eigenvektoren zum selben Eigenwert.

2.1 Der Satz von Keldysh

Wir führen den Satz von Keldysh zunächst in seiner allgemeinsten Form aus.

Satz 2.1.1 (Keldysh, nicht-linear)**.** Sei $\Lambda \subset \mathbb{C}$ ein beschränktes Gebiet und $A \in H(\Lambda, \mathbb{C}^{N\times N})$ holomorph. Es existiere ein $\lambda \in \Lambda$, sodass $A(\lambda) \in \mathrm{GL}_N(\mathbb{C})$.
Weiter sei $\lambda_1 \in \Lambda$ ein halb-einfacher Eigenwert, d.h. es existiere eine L_1-dimensionale Orthonormalbasis aus (Rechts-)Eigenvektoren $v_{1,1}, \ldots, v_{1,L_1}$ von $\ker A(\lambda_1)$. Für diese gelte

$$\forall\, l = 1, \ldots, L_1 : A'(\lambda_1) v_{1,l} \notin \operatorname{ran} A(\lambda_1). \tag{2.1}$$

Dann existiert eine Basis aus Links-Eigenvektoren $w_{1,1}, \ldots, w_{1,L_1}$ von $\ker A^*(\lambda_1)$, sodass

$$\forall\, l,k = 1,\dots,L_1 : w_{1,l}^* A'(\lambda_1) v_{1,k} = \delta_{l,k}. \tag{2.2}$$

Weiterhin existiert eine Umgebung U_1 von λ_1 und $R_1 \in H(U_1, \mathbb{C}^{N\times N})$ holomorph, sodass

$$\forall\, \lambda \in U_1 \setminus \{\lambda_1\} : A(\lambda)^{-1} = \frac{1}{\lambda - \lambda_1} S_1 + R_1(\lambda), \quad \text{mit} \quad S_1 := \sum_{l=1}^{L_1} v_{1,l} w_{1,l}^*. \tag{2.3}$$

Der Beweis dieses Satzes ist sehr aufwändig und nicht zentraler Teil dieser Arbeit. Wir verweisen dazu also auf [1]. Wir wollen aber dennoch eine Intuition dafür vermitteln. Dazu behandeln wir dessen linearen Spezialfall mit der zusätzlichen Voraussetzung, dass die Matrix dabei hermitesch ist.

Bemerkung 2.1.2 (Keldysh, linear, hermitesch)**.** Sei $A \in \mathbb{C}^{N\times N}$ hermitesch (selbstadjungiert). Seien $\lambda_1 < \cdots < \lambda_k$ deren Eigenwerte, jeweils zu den Vielfachheiten $L_1,\dots,L_k$, und für alle $n = 1,\dots,k$ sei $V_n = (v_{n,1},\dots,v_{n,L_n})$ eine Orthonormalbasis von $\ker(A - \lambda_n I_N)$, sodass A unitär diagonalisierbar ist, d.h.

$$A = V^* D V, \quad \text{mit} \quad V := (V_1,\dots,V_k) \in \mathrm{U}_N(\mathbb{C}), \quad D := \operatorname{diag}(\lambda_1 I_{L_1},\dots,\lambda_k I_{L_k}).$$

In Analogie zu Satz 2.1.1, bilden nun $-v_{1,1},\dots,-v_{1,L_1}$ eine Basis von $\ker(A^* - \overline{\lambda}_1 I_N)$, sodass

$$\forall\, l,k = 1,\dots,L_1 : -v_{1,l}^*(-I_N) v_{1,k} = \delta_{l,k},$$

und es gilt

$$\begin{aligned}
\forall\, \lambda \in U_1 \setminus \{\lambda_1\} : (A - \lambda I_N)^{-1} &= (V^* D V - V^* \lambda V)^{-1} \\
&= (V^* \operatorname{diag}((\lambda_1 - \lambda) I_{L_1},\dots,(\lambda_k - \lambda) I_{L_k}) V)^{-1} \\
&= V^* \operatorname{diag}\left(\frac{1}{\lambda_1 - \lambda} I_{L_1},\dots,\frac{1}{\lambda_N - \lambda} I_{L_k}\right) V \\
&\overset{!}{=} \sum_{n=1}^{k} \frac{1}{\lambda_n - \lambda} \sum_{l=1}^{L_n} v_{n,l}^* v_{n,l} \\
&= \frac{1}{\lambda - \lambda_1} \sum_{l=1}^{L_1} v_{1,l}(-v_{1,l})^* + R_1(\lambda),
\end{aligned}$$

wobei $U_1 \subseteq \rho(A) \cup \{\lambda_1\}$ eine nichtleere Umgebung von λ_1 ist und

$$R_1 \in H(U_1, \mathbb{C}^{N\times N}), \quad \lambda \mapsto \sum_{n=2}^{k} \frac{1}{\lambda - \lambda_n} \sum_{l=1}^{L_n} v_{n,l}(-v_{n,l})^*.$$

Die Gleichheit mit dem „!“ erhält man durch eine elementare, aber sperrige Rechnung.

Die Eigenschaften aus Satz 2.1.1 sind Verallgemeinerungen bekannter Begriffe und Tatsachen aus dem linearen Setting. (2.1) fordert, dass es keine Hauptvektoren zweiter Stufe gibt und (2.2) beschreibt eine gewisse Orthogonalität zwischen Links- und Rechts-Eigenvektoren. Die Zusammenhänge zwischen dem linearen und nichtlinearen Fall wollen wir nun noch mit folgender Proposition verdeutlichen.

Proposition 2.1.3 (Keldysh, linear)**.** Sei λ_1 ein halb-einfacher Eigenwert einer Matrix $A \in \mathbb{C}^{N\times N}$, d.h. geometrische Vielfachheit L_1^{geo} und algebraische Vielfachheit L_1^{alg} stimmen überein, d.h.

$$L_1 := L_1^{\text{geo}} := \operatorname{def}(A - \lambda_1 I_N) = L_1^{\text{alg}} := \mu_1 = \max\{\mu \in \mathbb{N} : (\lambda - \lambda_1)^\mu \mid \chi_A(\lambda)\}.$$

Dann gilt Folgendes.

(i) Es gibt eine Orthonormalbasis $V_1 = (v_{1,1}, \dots, v_{1,L_1})$ von $\ker(A - \lambda_1 I_N)$.

Weiters gelten folgende 2 Analoga zu Satz 2.1.1.

(ii) Es gibt eine Basis $W_1 = (w_{1,1}, \dots, w_{1,L_1})$ von $\ker(A^* - \overline{\lambda}_1 I_N)$, sodass

$$\forall\, l, k = 1, \dots, L_1 : (v_{1,k}, w_{1,l})_2 = -\delta_{l,k}.$$

(iii) Es existiert eine Umgebung U_1 von λ_1 und $R_1 \in H(U_1, \mathbb{C}^{N\times N})$ holomorph, sodass $\forall\, \lambda \in U_1 \setminus \{\lambda_1\}$:

$$(A - \lambda I_N)^{-1} = \frac{1}{\lambda - \lambda_1} S_1 + R_1(\lambda), \quad S_1 := \sum_{l=1}^{L_1} v_{1,l} w_{1,l}^*.$$

Beweis (als Korollar). Wir wollen Satz 2.1.1 auf $\lambda \mapsto A - \lambda I_N$ anwenden. Wir überprüfen also die Voraussetzungen.

1. Unsere (lineare) Matrix-Funktion $\lambda \mapsto A - \lambda I_N$ ist offensichtlich (komponentenweise) holomorph.
2. Für alle $\lambda \in \rho(A)$, ist $A - \lambda I_N \in \mathrm{GL}_N(\mathbb{C})$.
3. Die Existenz einer Orthonormalbasis V_1 von $\ker(A - \lambda_1 I_N)$, wenn $\lambda_1 \in \sigma(A)$ halb-einfach ist, ist einfache Lineare Algebra.
4. Wir müssen uns also eigentlich bloß überlegen, dass

$$\forall\, l = 1, \dots, L_1 : \frac{\mathrm{d}}{\mathrm{d}\lambda}(A - \lambda I_N)\Big|_{\lambda=\lambda_1} v_{1,l} = -I_N v_{1,l} = -v_{1,l} \notin \operatorname{ran}(A - \lambda_1 I_N).$$

Weil λ_1 halb-einfach ist, sein Jordan-Block genau $\lambda_1 I_{L_1}$. Alle zugehörigen Jordan-Kästchen haben also die Größe 1×1. Es gibt somit bloß Hauptvektoren erster Stufe (Eigenvekotren), und keine zweiter oder höherer Stufe.

Sei $y \in \ker(A - \lambda_1 I_N) \cap \operatorname{ran}(A - \lambda_1 I_N)$, dann

$$\exists\, x \in \mathbb{C}^N : (A - \lambda_1 I_N)x = y, \quad (A - \lambda_1 I_N)^2 x = (A - \lambda_1 I_N)y = 0.$$

Angenommen, $y \neq 0$, dann wäre x ein Hauptvektor zweiter Stufe. Widerspruch! Damit gilt also

$$\ker(A - \lambda_1 I_N) \cap \operatorname{ran}(A - \lambda_1 I_N) = \{0\} \not\ni v_{1,1}, \ldots, v_{1,L_1} \begin{cases} \in \ker(A - \lambda I_N), \\ \notin \operatorname{ran}(A - \lambda I_N). \end{cases}$$

Wir können also Satz 2.1.1 anwenden.

(ii) Der Satz 2.1.1 gibt uns eine Basis $w_{1,1}, \ldots, w_{1,L_1}$ von $\ker(A - \lambda_1 I_N)^* = \ker(A^* - \overline{\lambda}_1 I_N)$, sodass

$$\begin{aligned} \forall\, l, k = 1, \ldots, L_1 : (v_{1,k}, w_{1,l})_2 = w_{1,l}^* v_{1,k} &= -w_{1,l}^*(-I_N)v_{1,k} \\ &= -w_{1,l}^* \frac{\mathrm{d}}{\mathrm{d}\lambda}(A - \lambda I_N)\Big|_{\lambda=\lambda_1} v_{1,k} = -\delta_{l,k}. \end{aligned}$$

(iii) Diese Tatsache kann eins zu eins aus Satz 2.1.1 übernommen werden.

□

Bemerkung 2.1.4 (Spektrale Projektion)**.** $P_1 := S_1 A'(\lambda_1)$ ist eine Projektion von $\mathbb{C}^N$ auf den (Rechts-)Eigenraum $\ker A(\lambda_1)$. Sei nämlich $x \in \mathbb{C}^N$, so gilt

$$P_1 x = \sum_{l=1}^{L_1} v_{1,l} \underbrace{w_{1,l}^* A'(\lambda_1) x}_{\in \mathbb{C}} \in \ker A(\lambda).$$

Außerdem ist P_1 idempotent, weil

$$\begin{aligned} \forall\, x \in \ker A(\lambda_1) : P_1 x &= \sum_{l=1}^{L_1} v_{1,l} w_{1,l}^* A'(\lambda_1) \sum_{k=1}^{L_1} (x, v_{1,k})_2 v_{1,k} \\ &= \sum_{l=1}^{L_1} \sum_{k=1}^{L_1} (x, v_{1,k})_2 v_{1,l} \underbrace{w_{1,l}^* A'(\lambda_1) v_{1,k}}_{=\delta_{l,k}} = \sum_{l=1}^{L_1} (x, v_{1,l})_2 v_{1,l} = x. \end{aligned}$$

Das folgende Resultat bringt eine Verallgemeinerung vom Satz von Keldysh auf endlich viele Eigenwerte. Es wird eine grundlegende Rolle in der Konstruktion des Algorithmus spielen.

Korollar 2.1.5. Sei $\Lambda \subset \mathbb{C}$ ein beschränktes Gebiet und $A \in H(\Lambda, \mathbb{C}^{N\times N})$ holomorph. Es existiere ein $\lambda \in \Lambda$, sodass $A(\lambda) \in \mathrm{GL}_N(\mathbb{C})$, d.h. invertierbar ist. Mögen $\lambda_1, \ldots, \lambda_k$, paarweise verschiedene Eigenwerte von A, die Voraussetzungen von Satz 2.1.1 erfüllen. Seien $S_1, \ldots, S_k$ die entsprechenden Summen aus (2.3).
Dann existiert ein $R \in H(\Lambda, \mathbb{C}^{N\times N})$, sodass

$$\forall\, \lambda \in \Lambda \setminus \{\lambda_1, \ldots, \lambda_k\} : A(\lambda)^{-1} = \sum_{n=1}^{k} \frac{1}{\lambda - \lambda_n} S_n + R(\lambda). \tag{2.4}$$

Beweis. Laut 2.1.1, existiert für $m = 1, \ldots, k$ eine Umgebung U_m von λ_m und $R_m \in H(U_m, \mathbb{C}^{N\times N})$ holomorph, sodass

$$\forall\, \lambda \in U_m \setminus \{\lambda_m\} : A(\lambda)^{-1} = \frac{1}{\lambda - \lambda_m} S_m + R_m(\lambda).$$

Die beiden Funktionen $R_\Lambda \in H(\Lambda \setminus \{\lambda_1, \ldots, \lambda_k\}, \mathbb{C}^{N\times N})$ und $R_{U_m} \in H(U_m, \mathbb{C}^{N\times N})$ stimmen auf $U_m \setminus \{\lambda_m\}$ überein, weil

$$\begin{aligned}\forall\, \lambda \in U_m \setminus \{\lambda_m\} : R_\Lambda(\lambda) &:= A(\lambda)^{-1} - \sum_{n=1}^{k} \frac{1}{\lambda - \lambda_n} S_n \\ &= \frac{1}{\lambda - \lambda_m} S_m + R_m(\lambda) - \sum_{n=1}^{k} \frac{1}{\lambda - \lambda_n} S_n = R_m(\lambda) - \sum_{\substack{n=1 \\ n\neq m}}^{k} \frac{1}{\lambda - \lambda_n} S_n =: R_{U_m}(\lambda).\end{aligned}$$

Damit können wir den Identitätssatz für holomorphe Funktionen anwenden. Die Funktion

$$R : \Lambda \to C^{N\times N} : \lambda \mapsto \begin{cases} R_\Lambda(\lambda), & \lambda \in \Lambda \setminus \{\lambda_1, \ldots, \lambda_k\}, \\ R_{U_m}(\lambda), & \exists\, m = 1, \ldots, k : \lambda \in U_m, \end{cases}$$

ist also wohldefiniert, holomorph auf Λ, und leistet das Gewünschte.

□

3 Herleitung des Verfahrens

3.1 Definitionen und Vorbereitungen

Es gelten die Bezeichnungen des vorherigen Kapitels. Sei $\Gamma \subset \Lambda$ zudem eine positiv orientierte Jordan-Kurve (d.h. ein geschlossener Weg). Γ umschließe alle Eigenwerte $\lambda_1, \ldots, \lambda_k \notin \Gamma$, und sonst keine weiteren. Eine Veranschaulichung davon ist in Abbildung 3.1 zu finden.

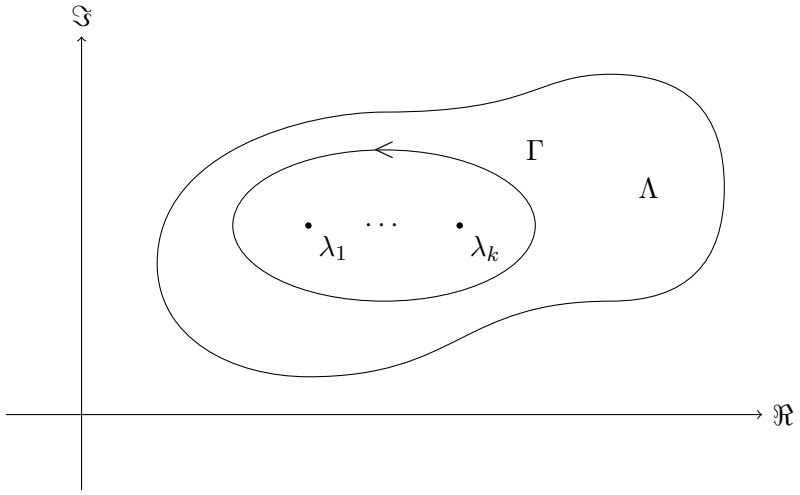

Abbildung 3.1: Λ Gebiet, Γ Jordan-Kurve, $\lambda_1, \ldots, \lambda_k$ Eigenwerte

Sei nun $f \in H(\mathbb{C}, \mathbb{C})$ holomorph. Für $n = 1, \ldots, k$, sei dazu r_n hinreichend klein, sodass $B_{r_n}(\lambda_n) \subset U_n$. Wir wenden Korollar 2.1.5 sowie die Cauchyche Integralformel und den Cauchychen Integralsatz an, und erhalten

$$\begin{aligned}
\frac{1}{2\pi i}\int_\Gamma f(\lambda)A(\lambda)^{-1}\,\mathrm{d}\lambda &\overset{2.1.5}{=} \frac{1}{2\pi i}\int_\Gamma f(\lambda)\left(\sum_{n=1}^{k}\frac{1}{\lambda-\lambda_n}S_n + R(\lambda)\right)\mathrm{d}\lambda \\
&= \sum_{n=1}^{k}\underbrace{\frac{1}{2\pi i}\int_{\partial B_{r_n}(\lambda_n)}\frac{f(\lambda)}{\lambda-\lambda_n}\,\mathrm{d}\lambda}_{=f(\lambda_n)}S_n + \frac{1}{2\pi i}\underbrace{\int_\Gamma f(\lambda)R(\lambda)\,\mathrm{d}\lambda}_{=0} \qquad (3.1)\\
&= \sum_{n=1}^{k} f(\lambda_n)\sum_{l=1}^{L_n} v_{n,l}w_{n,l}^*.
\end{aligned}$$

Diese Formel bildet die Grundlage zur Berechnung der Eigenwerte. Sei J die Summe aller Eigenraum-Dimensionen, d.h.

$$J := \sum_{n=1}^{k} L_n.$$

Wir vereinigen die Basen aller (Rechts)-Eigenräume zu

$$\begin{aligned} V := (V_1, \dots, V_k) &= (v_{1,1}, \dots, v_{1,L_1}, \dots, v_{k,1}, \dots, v_{k,L_k}) \\ &= \begin{pmatrix} v_{1,1,1} & \cdots & v_{1,L_1,1} & \cdots & v_{k,1,1} & \cdots & v_{k,L_k,1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ v_{1,1,N} & \cdots & v_{1,L_1,N} & \cdots & v_{k,1,N} & \cdots & v_{k,L_k,N} \end{pmatrix} \in \mathbb{C}^{N \times J}. \end{aligned}$$

Wir vereinigen die Basen aller Links-Eigenräume zu

$$\begin{aligned} W := (W_1, \dots, W_k) &= (w_{1,1}, \dots, w_{1,L_1}, \dots, w_{k,1}, \dots, w_{k,L_k}) \\ &= \begin{pmatrix} w_{1,1,1} & \cdots & w_{1,L_1,1} & \cdots & w_{k,1,1} & \cdots & w_{k,L_k,1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{1,1,N} & \cdots & w_{1,L_1,N} & \cdots & w_{k,1,N} & \cdots & w_{k,L_k,N} \end{pmatrix} \in \mathbb{C}^{N \times J}. \end{aligned}$$

Sei D die Diagonal-Matrix der Eigenwerte, ihrer Vielfachheit nach aufgeführt, d.h.

$$D := \operatorname{diag}(\underbrace{\lambda_1, \dots, \lambda_1}_{L_1\text{-viele}}, \dots, \underbrace{\lambda_k, \dots, \lambda_k}_{L_k\text{-viele}}) = \operatorname{diag}(\lambda_1 I_{L_1}, \dots, \lambda_k I_{L_k}) \in \mathbb{C}^{N \times N}. \tag{3.2}$$

Sei $\hat{V} \in \mathbb{C}^{N \times j}$ zunächst eine beliebige Zufallsmatrix mit $J \leq j \ll N$. Die folgenden Matrizen A_0 und A_1 sind der Grund für die Namensgebung der „Konturintegral-Methode“.

$$A_0 := \frac{1}{2\pi i} \int_\Gamma \lambda^0 A(\lambda)^{-1} \hat{V} \, \mathrm{d}\lambda \in \mathbb{C}^{N \times j}, \quad A_1 := \frac{1}{2\pi i} \int_\Gamma \lambda^1 A(\lambda)^{-1} \hat{V} \, \mathrm{d}\lambda \in \mathbb{C}^{N \times j} \tag{3.3}$$

Mittels (3.1) lassen sich nun A_0 und A_1 wie folgt umformen. Letztere Gleichheit ist dabei eine elementare, aber sperrige Rechnung.

$$A_i \overset{(3.3)}{=} \frac{1}{2\pi i} \int_\Gamma \lambda^i A(\lambda)^{-1} \hat{V} \, \mathrm{d}\lambda \overset{(3.1)}{=} \sum_{n=1}^{k} \lambda_n^i \sum_{l=1}^{L_n} v_{n,l} w_{n,l}^* \hat{V} = V D^i W^* \hat{V}, \quad i = 0, 1 \tag{3.4}$$

3.2 Überblick

Wenn wir in (3.4) explizit für $i = 0, 1$ einsetzen, erhalten wir

$$A_0 = VW^*\hat{V}, \quad A_1 = VDW^*\hat{V}. \tag{3.5}$$

Nun gilt es, D, die Diagonal-Matrix aller Eigenwerte aus den Matrizen A_0 und A_1 zu berechnen. Unsere Hoffnung besteht nämlich darin, D aus A_1 mittels A_0 zu „isolieren“.
Dazu müssen A_0 und A_1 zunächst „berechnet“, bzw. mit einer Quadraturformel approximiert werden. Das motivieren wir im ersten Schritt mit einer komplexen Variante der summierten Trapezregel. Nachdem dieser Schritt der weitaus teuerste ist, folgt eine genaue Konvergenz-Analyse im vierten Kapitel. Wie diese „Isolation“ von D vonstattengehen soll, erklären wir vollständig im zweiten Schritt.
Die Konturintegral-Methode ist zunächst oberflächlich im Algorithmus 1 zusammengefasst worden. Genauere Erklärungen zur ersten und zweiten Zeile folgen im ersten und zweiten Schritt. Die Auseinandersetzung mit der dritten Zeile (bzw. dem dritten Schritt) ist nicht Teil dieser Arbeit. Wir verwenden lediglich den Befehl `linalg.eigvals`, aus der `scipy`-Bibliothek.

Algorithm 1 Integral-Methode

Berechne (bzw. approximiere) $A_0, A_1 \in \mathbb{C}^{N\times j}$;
Berechne eine reduzierte Singulärwert-Zerlegung $A_0 = \tilde{V}\Sigma\tilde{W}^*$ auf J Singulärwerte;
Berechne Eigenwerte $\lambda_1, \ldots, \lambda_k$ der Matrix $\tilde{V}A_1\tilde{W}\Sigma^{-1} \sim D$;
return $\lambda_1, \ldots, \lambda_k$

3.3 Approximation der Konturintegrale

In der Praxis werden wir die Kurve Γ innerhalb welcher die gesuchten Eigenwerte liegen, zumeist als Kreis wählen. Betrachte daher die Parametrisierung von $\partial B_R(0)$ vermöge dem komplexen Weg

$$\gamma : [0,1) \to \partial B_R(0) : t \mapsto R\exp(2\pi i t), \quad \text{mit} \quad \gamma' : t \mapsto 2\pi i R\exp(2\pi i t).$$

Wir approximieren ihn durch die m-ten Einheitswurzeln. Diese wählen wir dann auch als Quadraturknoten mit uniformen Gewicht $1/m$. In Formeln geschrieben, lauten diese

$$\omega_m^0, \ldots, \omega_m^{m-1}, \quad \text{mit} \quad \omega_m := \exp\left(\frac{2\pi i}{m}\right), \quad m \in \mathbb{N}.$$

Sei $f \in H(U, \mathbb{C})$ holomorph. Wir verwenden nun tatsächlich eine klassische summierte Rechtecksregel und erhalten

$$
\begin{aligned}
Q(f) &:= \frac{1}{2\pi i}\int_{|\lambda|=R} f(\lambda)\ \mathrm{d}\lambda = \frac{1}{2\pi i}\int_{\gamma} f(\lambda)\ \mathrm{d}\lambda = \frac{1}{2\pi i}\int_0^1 \gamma'(t) f(\gamma(t))\ \mathrm{d}t \\
&= \int_0^1 R\exp(2\pi i t) f(R\exp(2\pi i t))\ \mathrm{d}t \\
&\approx \sum_{\nu=0}^{m-1} \frac{1}{m} R\exp\left(\frac{2\pi i\nu}{m}\right) f\left(R\exp\left(\frac{2\pi i\nu}{m}\right)\right) \\
&= \frac{R}{m}\sum_{\nu=0}^{m-1} \omega_m^\nu f(R\omega_m^\nu) =: Q_m(f).
\end{aligned}
$$

Für f werden wir in der Implementierung des Algorithmus $\lambda^i A(\lambda)^{-1}\hat{V}$ mit $i = 0, 1$ einsetzen. Die (überaus hohe) Genauigkeit der Quadraturformel Q_m wird im vierten Kapitel diskutiert.
Um $A(\lambda)^{-1}\hat{V}$ in den Integranden von A_0 und A_1 zu bestimmen, werden wir nicht die Inverse $A(\lambda)^{-1}$ direkt berechnen. Stattdessen bestimmen wir eine LU-Zerlegung von $A(\lambda)$. Dann führen wir für alle Spalten $\hat{v}_1, \ldots, \hat{v}_j$ der Zufallsmatrix $\hat{V} \in \mathbb{C}^{N\times j}$ jeweils eine Vorwärts- und Rückwärts-Substitution durch. Es gilt ja schließlich

$$
\forall\, i = 1, \ldots, j : A(\lambda)^{-1}\hat{v}_i = x_i \iff \hat{v}_i = A(\lambda)x_i = LUx_i = Ly_i, \quad y_i = Ux_i.
$$

Somit können dann die Spalten $x_1, \ldots, x_j$ von $A(\lambda)^{-1}\hat{V}$ effizient ermittelt werden, ohne $A(\lambda)^{-1}$ explizit auszurechnen. Um technische Details der LU-Zerlegung, z.B. das Arbeiten mit Permutationsmatrizen, Pivotsuche usw., machen wir uns hierbei aber keine Gedanken. Wir verwenden lediglich die Befehle `linalg.lu_factor` und `linalg.lu_solve` aus der `scipy`-Bibliothek.

3.4 Singulärwert-Zerlegung

Wir führen zunächst eine Singulärwert-Zerlegung von A_0 durch. Dies ist eine Zerlegung der Form

$$
\begin{gathered}
\tilde{V}\Sigma\tilde{W}^* = A_0 \in \mathbb{C}^{N\times j}, \\
\tilde{V} \in \mathrm{U}_N(\mathbb{C}), \quad \tilde{W} \in \mathrm{U}_j(\mathbb{C}), \quad \Sigma = \begin{pmatrix} \mathrm{diag}(\sigma_1, \ldots, \sigma_J, \ldots, \sigma_j) \\ 0_{(N-j)\times j} \end{pmatrix} \in \mathbb{C}^{N\times j}.
\end{gathered}
$$

Dabei gehen wir nicht auf die technischen Details zur Bestimmung der Singulärwert-Zerlegung ein. Wir verwenden lediglich den Befehl `linalg.svd` aus der `scipy`-Bibliothek.
Wir nehmen zunächst an, dass die Rechts- bzw. Links-Eigenraum-Basen $V, W \in \mathbb{C}^{N\times J}$ vollen Rang J haben, und die Zufallsmatrix $\hat{V} \in \mathbb{C}^{N\times j}$ vollen Rang j hat. Weiters gehen wir davon aus, dass $W^*\hat{V} \in C^{J\times j}$ vollen Rang $J \leq j$ hat. Das heißt, V und $W^*\hat{V}$ sind

injektiv, also auch deren Komposition $A_0 = VW^*\hat{V}$. Daher ist auch die Annahme, dass A_0 Rang J hat, sinnvoll.
Matrizen haben genau dann denselben Rang, wenn sie bzgl. $\equiv$ äquivalent sind (d.h. modulo Multiplikation mit zwei regulären Matrix von jeweils links bzw. rechts). $\tilde{V}$ und $\tilde{W}^*$ sind als unitäre Matrizen regulär. Für jede $\equiv$-Äquivalenzklasse finden wir einen Repräsentanten in einer Normalform als Blockmatrix einer Einheitsmatrix und sonst 0-en. Wir erhalten die Äquivalenz

$$\begin{pmatrix}\operatorname{diag}(\sigma_1,\dots,\sigma_J,\dots,\sigma_j)\\ 0_{(N-j)\times j}\end{pmatrix} = \Sigma \equiv \tilde{V}\Sigma\tilde{W}^* = A_0 \equiv \begin{pmatrix} I_J & 0_{J\times(j-J)}\\ 0_{(N-J)\times J} & 0_{(N-J)\times(j-J)}\end{pmatrix}.$$

Damit verschwinden die letzten Singulärwerte $\sigma_{J+1} = \dots = \sigma_j = 0$. Somit können wir statt der alten vollen Singulärwert-Zerlegung (jetzt indiziert mit „full“) eine reduzierte (indiziert mit „reduced“) verwenden. Dabei bestehen $\tilde{V}_{\text{reduced}}$, Σ_{reduced}, und $\tilde{W}_{\text{reduced}}$ aus den ersten J Spalten von $\tilde{V}_{\text{full}}$, Σ_{full}, bzw. $\tilde{W}_{\text{full}}$. Diese beiden Singulärwert-Zerlegungen sind tatsächlich gleichwertig, weil

$$\begin{aligned} A_0 &= \tilde{V}_{\text{full}}\Sigma_{\text{full}}\tilde{W}^*_{\text{full}} = \begin{pmatrix}\tilde{V}_{\text{reduced}} & *\end{pmatrix}\begin{pmatrix}\Sigma_{\text{reduced}} & 0_{J\times(j-J)}\\ 0_{(N-J)\times J} & 0_{(N-J)\times(j-J)}\end{pmatrix}\begin{pmatrix}\tilde{W}^*_{\text{reduced}}\\ *\end{pmatrix}\\ &= \tilde{V}_{\text{reduced}}\Sigma_{\text{reduced}}\tilde{W}^*_{\text{reduced}}.\end{aligned}$$

Die reduzierten Matrizen sind zwar nicht mehr unitär, aber es gilt

$$\begin{aligned} I_N &= \tilde{V}^*_{\text{full}}\tilde{V}_{\text{full}} = \begin{pmatrix}\tilde{V}^*_{\text{reduced}}\\ *\end{pmatrix}\begin{pmatrix}\tilde{V}_{\text{reduced}} & *\end{pmatrix} = \begin{pmatrix}\tilde{V}^*_{\text{reduced}}\tilde{V}_{\text{reduced}} & *\\ * & *\end{pmatrix},\\ I_j &= \tilde{W}^*_{\text{full}}\tilde{W}_{\text{full}} = \begin{pmatrix}\tilde{W}^*_{\text{reduced}}\\ *\end{pmatrix}\begin{pmatrix}\tilde{W}_{\text{reduced}} & *\end{pmatrix} = \begin{pmatrix}\tilde{W}^*_{\text{reduced}}\tilde{W}_{\text{reduced}} & *\\ * & *\end{pmatrix}.\end{aligned}$$

Wir vereinbaren, ab sofort nur noch die reduzierte Singulärwert-Zerlegung zu verwenden und den Index wegzulassen. Insgesamt erhalten wir also unsere reduzierte Singulärwert-Zerlegung

$$\begin{gathered} A_0 = \tilde{V}\Sigma\tilde{W}^*,\\ \tilde{V}\in\mathbb{C}^{N\times J},\quad \tilde{W}\in\mathbb{C}^{j\times J},\quad \Sigma = \operatorname{diag}(\sigma_1,\dots,\sigma_J)\in\mathbb{C}^{J\times J},\\ \tilde{V}^*\tilde{V} = \tilde{W}^*\tilde{W} = I_J \end{gathered} \tag{3.6}$$

Da wir an V vollen Rang vorausgesetzt haben, können wir auch annehmen, dass $S := \tilde{V}^*V \in \mathbb{C}^{J\times J}$ vollen Rang J hat, also $\in \mathrm{GL}_J(\mathbb{C})$, d.h. invertierbar ist. Nun nutzen wir die Singulärwert-Zerlegung von A_0 und berechnen

$$\Sigma\tilde{W}^* \overset{(3.6)}{=} \tilde{V}^*\tilde{V}\Sigma\tilde{W}^* \overset{(3.6)}{=} \tilde{V}^*A_0 \overset{(3.5)}{=} \tilde{V}^*VW^*\hat{V} = SW^*\hat{V}.$$

Setzen wir in dies in die Darstellung von A_1 über die Konturintegrale ein, erhalten wir

$$A_1 \overset{(3.5)}{=} VDW^*\hat{V} = VDS^{-1}\Sigma\tilde{W}^*,$$

was uns schließlich

$$\tilde{V}^*A_1\tilde{W}\Sigma^{-1} = \tilde{V}^*VDS^{-1}\Sigma\tilde{W}^*\tilde{W}\Sigma^{-1} = SDS^{-1} \sim D$$

liefert.
Somit ist die Diagonalmatrix D, welche ja die gesuchten Eigenwerte $\lambda_1, \ldots, \lambda_k$ von A enthält, ähnlich zur Matrix $\tilde{V}^*A_1\tilde{W}\Sigma^{-1}$. Ähnliche Matrizen haben dieselben Eigenwerte.

3.5 Voraussetzungen

Bei der Herleitung des Algorithmus' haben wir unter anderem auf die Matrizen V, W, D, und S zurückgegriffen. Man sollte sich aber bewusst sein, dass diese Matrizen bei der Implementierung des Algorithmus nicht verwendet werden, nachdem sie ohne Kenntnis der gesuchten Eigenpaare nicht berechenbar sind.
Die Tatsache, dass sie nicht im Algorithmus 1 vorkommen, heißt aber nicht, dass ihre Eigenschaften (z.B. $S \in \mathrm{GL}_J(\mathbb{C})$) unwichtig sind. Wählt man z.B. eine zu kleine Matrix $\hat{V}$, könnten wir unter Umständen die Regularität von S verlieren. Die numerischen Resultate werden dann in der Regel inkorrekt sein.
Wir fassen zusammen, was der Algorithmus 1 braucht, um korrekte Ergebnisse zu liefern.

1. $\lambda_1, \ldots, \lambda_k \in \Lambda$ seien allersamt halb-einfach und liegen im Inneren von Γ, d.h. insbesondere nicht darauf. Sonst könnten theoretisch Probleme bei der Approximationen von A_0 und A_1, vermöge einer der Quadraturformeln Q_m, $m \in \mathbb{N}$, entstehen. Sollte dies passieren, müsste man die Kurve Γ anpassen.

2. $V, W \in \mathbb{C}^{N\times J}$ haben vollen Rang J, d.h. deren Spalten seien linear unabhängig.

   V, W bestehen ja schließlich aus Rechts- bzw. Links-Eigenvektoren. Die Voraussetzung gilt zumindest im linearen Fall, daher ist es durchaus plausibel sie auch im nichtlinearen Fall anzunehmen.

3. $\hat{V}$ sei eine hinreichend große, gleichverteilt gewählte Zufallsmatrix, mit vollem Rang j.

   Zur Illustration, dass auch diese Annahme sinnvoll ist, sei dazu K ein endlicher Körper und $\hat{V} \in K^{j\times j}$. Durch Abzählen der möglichen Komponenten der Spalten der regulären Matrizen, kommt man auf die Wahrscheinlichkeit

$$\begin{aligned}\mathbf{P}(\hat{V} \in \mathrm{GL}_j(K)) &= \frac{|\,\mathrm{GL}_j(K)|}{|K^{j\times j}|} = \frac{1}{|K|^{j\cdot j}}\prod_{i=1}^{j}\left(|K|^j - |K|^{i-1}\right)\\ &= \prod_{i=1}^{j}\frac{|K|^j - |K|^{i-1}}{|K|^j} = \prod_{i=1}^{j}\left(1 - \frac{1}{|K|^{j+1-i}}\right) \xrightarrow{|K|\to|\mathbb{C}|} 1.\end{aligned}$$

4. $W^*\hat{V} \in C^{J\times j}$ habe vollen Rang $J \leq j$. Diese Annahme ist sinnvoll. Es gilt nämlich

$$W^*\hat{V} = (w_{1,1}, \ldots, w_{k,L_k})^*(\hat{v}_1, \ldots, \hat{v}_j)$$
$$= \begin{pmatrix} w_{1,1}^*\hat{v}_1 & \cdots & w_{1,1}^*\hat{v}_j \\ \vdots & \ddots & \vdots \\ w_{k,L_k}^*\hat{v}_1 & \cdots & w_{k,L_k}^*\hat{v}_j \end{pmatrix} = \begin{pmatrix} (\hat{v}_1, w_{1,1})_2 & \cdots & (\hat{v}_j, w_{1,1})_2 \\ \vdots & \ddots & \vdots \\ (\hat{v}_1, w_{k,L_k})_2 & \cdots & (\hat{v}_j, w_{k,L_k})_2 \end{pmatrix}.$$

Wir haben bereits vorausgesetzt, dass $\hat{V} \in \mathbb{C}^{N\times j}$ vollen Rang j hat, d.h. $\hat{v}_1, \ldots, \hat{v}_j$ linear unabhängig sind. Die Spalten von $W^*\hat{V}$ wären also genau dann linear abhängig, wenn eine davon 0 ist, d.h.

$$\exists\, i = 1, \ldots, j : (\hat{v}_i, w_{1,1})_2 = \cdots = (\hat{v}_i, w_{k,L_k})_2 = 0, \quad \text{d.h.} \quad \hat{v}_i \in (\operatorname{span} W)^{\perp_2}.$$

In der Praxis wird dieser Fall zum Glück kaum eintreten. Hier sieht man auch den Grund, warum wir $\hat{V}$ als Zufallsmatrix gewählt haben. Würde man beispielsweise stattdessen als Spalten in $\hat{V}$ Einheitsvektoren wählen, könnte diese Bedingung in der Praxis durchaus verletzt werden. Bei der Vektoriteration (Potenzmethode) geht man beispielweise auch davon aus, dass der Startvektor (Zufallsvektor) nicht orthogonal auf den Eigenraum des betrags-größten Eigenwerts steht.

5. $S := \tilde{V}^*V \in \mathbb{C}^{J\times J}$ habe vollen Rang J, also $S \in \mathrm{GL}_J(\mathbb{C})$. Diese Annahme ist, analog zum vorigen Punkt sinnvoll, da

$$S^* = V^*\tilde{V} = (v_1, \ldots, v_J)^*(\tilde{v}_1, \ldots, \tilde{v}_J)$$
$$= \begin{pmatrix} v_1^*\tilde{v}_1 & \cdots & v_1^*\tilde{v}_J \\ \vdots & \ddots & \vdots \\ v_J^*\tilde{v}_1 & \cdots & v_J^*\tilde{v}_J \end{pmatrix} = \begin{pmatrix} (\tilde{v}_1, v_1)_2 & \cdots & (\tilde{v}_J, v_1)_2 \\ \vdots & \ddots & \vdots \\ (\tilde{v}_1, v_J)_2 & \cdots & (\tilde{v}_J, v_J)_2 \end{pmatrix}.$$

Die Matrix $\tilde{V} \in C^{N\times J}$ hat, als Teilmatrix einer unitären Matrix, vollen Rang J, d.h. $\tilde{v}_1, \ldots, \tilde{v}_J$ sind linear unabhängig. Die Spalten von S^* wären also genau dann linear abhängig, wenn eine davon 0 ist, d.h.

$$\exists\, i = 1, \ldots, J : (\tilde{v}_i, v_1)_2 = \cdots = (\tilde{v}_i, v_J)_2 = 0, \quad \text{d.h.} \quad \tilde{v}_i \in (\operatorname{span} V)^{\perp_2}.$$

Wieder ist dieser Fall in der Praxis vernachlässigbar.

4 Analyse des Algorithmus

In diesem Kapitel untersuchen wir nun den teuersten Schritt des Verfahrens: Die Approximation von A_0 und A_1 durch eine geeigente Quadraturformel.

4.1 Konvergenz der Quadraturformel

In diesem Abschnitt werden wir die exponentielle Konvergenz der Quadraturformel auf einem Kreis für holomorphe Funktionen auf einem Gebiet rund um den Kreis nachweisen.

Lemma 4.1.1. Sei $R > 0$, $1 < a_-, 1 < a_+$ und U ein Ringgebiet.

$$U := \{z \in \mathbb{C} : R/a_- < |z| < R \cdot a_+\}$$

Weiter sei $f \in H(U, \mathbb{C}^N)$ eine holomorphe Funktion und

$$Q(f) := \frac{1}{2\pi i} \int_{|\lambda|=R} f(\lambda)\, \mathrm{d}\lambda.$$

Dann gilt für die summierte Rechtecksregel (hier gleichbedeutend mit einer summierten Trapezregel)

$$Q_m(f) := \frac{R}{m} \sum_{\nu=0}^{m-1} \omega_m^\nu f(R\omega_m^\nu), \quad \omega_m := \exp \frac{2\pi i}{m},$$

mit $m \in \mathbb{N}$ Quadraturknoten für alle $\rho_\pm \in (1, a_\pm)$ die Fehlerabschätzung

$$|Q_m(f) - Q(f)| \leq \max_{|\lambda|=R\rho_+} \|f(\lambda)\| \frac{\rho_+^{-m}}{1-\rho_+^{-m}} + \max_{|\lambda|=R\rho_-} \|f(\lambda)\| \frac{\rho_-^{-m}}{1-\rho_-^{-m}}.$$

Beweis. Wir können f in eine Laurent-Reihe entwickeln, d.h.

$$f(\lambda) = \sum_{\mu=-\infty}^{\infty} \lambda^\mu f_\mu, \quad \text{mit} \quad f_\mu := \frac{1}{2\pi i} \int_{|\lambda|=R} \lambda^{-\mu-1} f(\lambda)\, \mathrm{d}\lambda.$$

Aufgrund der Holomorphie von f konvergiert die Reihe für alle λ gleichmäßig in einer kompakten Teilmenge von D. Betrachte die Parametrisierung

$$\gamma : [0,1] \to \partial B_R^{\mathbb{C}}(0), \quad t \mapsto R\exp(2\pi i t).$$

Wir berechnen für das Integral

$$Q(\lambda \to \lambda^\mu) = \frac{1}{2\pi i}\int_{|\lambda|=R} \lambda^\mu \, \mathrm{d}\lambda = \frac{1}{2\pi i}\int_\gamma \lambda^\mu \, \mathrm{d}\lambda = \frac{1}{2\pi i}\int_0^1 \gamma'(t)\gamma(t)^\mu \, \mathrm{d}t$$

$$\begin{aligned}
&= R\int_0^1 \exp(2\pi i t)(R\exp(2\pi i t))^\mu \, \mathrm{d}t \\
&= R^{\mu+1}\int_0^1 \exp(2\pi i t(\mu+1)) \, \mathrm{d}t \\
&= \begin{cases} R^{\mu+1}, & \mu+1=0, \\ 0, & \text{sonst}, \end{cases}
\end{aligned}$$

sowie die Quadratur

$$\begin{aligned}
Q_m(\lambda \to \lambda^\mu) &= \frac{R}{m}\sum_{\nu=0}^{m-1} \omega_m^\nu (R\omega_m^\nu)^\mu = \frac{R^{\mu+1}}{m}\sum_{\nu=0}^{m-1} \exp\left(\frac{2\pi i}{m}(\mu+1)\right)^v \\
&= \begin{cases} R^{\mu+1}, & \mu+1 \in m\mathbb{Z} \\ \frac{R^{\mu+1}}{m}\frac{1-\exp(2\pi i(\mu+1))}{1-\exp\left(\frac{2\pi i}{m}(\mu+1)\right)} = 0, & \text{sonst.} \end{cases}
\end{aligned}$$

Zusammengesetzt ergibt das

$$Q_m(\lambda \to \lambda^\mu) - Q(\lambda \to \lambda^\mu) = \begin{cases} R^{\mu+1}, & \mu+1 \in m\mathbb{Z} \setminus \{0\}, \\ 0, & \text{sonst.} \end{cases}$$

Mit der Linearität und Stetigkeit des Integrals, sowie der Quadraturformel erhalten wir somit

$$Q_m(f) - Q(f) = \sum_{l=1}^{\infty}(f_{lm}R^{lm} + f_{-lm}R^{-lm}).$$

Zur Berechnung der Koeffizienten können wir, mit Hilfe des Cauchy'schen Integralsatzes, die Kurve über die integriert wird innerhalb von D beliebig verändern. Damit berechnen wir

$$
\begin{aligned}
|f_{lm}R^{lm}| &= \left| \frac{R^{lm}}{2\pi i} \int_{|\lambda|=\rho^+ R} \lambda^{-lm-1} f(\lambda)\ \mathrm{d}\lambda \right| \\
&= \left| R^{lm+1}\rho_+ \int_0^1 \exp(2\pi it)(R\rho_+ \exp(2\pi it))^{-lm-1} f(\exp(2\pi it))\ \mathrm{d}t \right| \\
&= \left| \rho_+^{-lm} \int_0^1 \exp(2\pi it)^{-lm} f(\exp(2\pi it))\ \mathrm{d}t \right| \leq \max_{|\lambda|=\rho^+ R} \|f(\lambda)\| \rho_+^{-lm}
\end{aligned}
$$

und analog

$$
|f_{-lm}R^{-lm}| \leq \max_{|\lambda|=\rho_- R} \|f(\lambda)\| \rho_-^{-lm}.
$$

Daraus erhalten wir zusammen

$$
\begin{aligned}
|Q_m(f) - Q(f)| &\leq \sum_{l=1}^{\infty} \left(\max_{|\lambda|=\rho^+ R} \|f(\lambda)\| \rho_+^{-lm} + \max_{|\lambda|=\rho_- R} \|f(\lambda)\| \rho_-^{-lm} \right) \\
&= \max_{|\lambda|=R\rho_+} \|f(\lambda)\| \frac{\rho_+^{-m}}{1-\rho_+^{-m}} + \max_{|\lambda|=R\rho_-} \|f(\lambda)\| \frac{\rho_-^{-m}}{1-\rho_-^{-m}}.
\end{aligned}
$$

□

Für weitergehende Konvergenzanalyse zitieren wir aus [2] noch, ohne Beweis, den folgenden, auf Lemma 4.1.1 aufbauenden Satz.

Satz 4.1.2. Werden die Kurven in der Definition (3.3) von A_0 und A_1 als Kreise der Form $z_0 + R\mathrm{e}^{it}$ mit $t \in [0, 2\pi)$ gewählt und durch eine Rechtecksregel mit m äquidistant verteilten Punkten diskretisiert, so erhalten wir für die angenäherten Matrizen $A_{0,1}^{(m)} \in \mathbb{C}^{N\times j}$ die Fehlerabschätzung

$$
\left\| A_{0,1}^{(m)} - A_{0,1} \right\| \leq C(\rho_-^{m-r+1} + \rho_+^{m-r+1}),
$$

wobei r die maximale Ordnung der Pole von $A(\lambda)^{-1}$ und

$$
\rho_- := \max_{\substack{\lambda\in\sigma(A) \\ |\lambda-z_0|<R}} \frac{|\lambda - z_0|}{R}, \quad \rho_+ := \max_{\substack{\lambda\in\sigma(A) \\ |\lambda-z_0|>R}} \frac{R}{|\lambda - z_0|}
$$

ist.

Als unmittelbare Folgerung dieser Resultate sehen wir, dass die Konvergenzrate unserer Quadraturformel wesentlich von der Distanz des nähesten Eigenwert zu unserer Kurve abhängt. Daher ist es wohl von Vorteil im Algorithmus die Kurve über die integriert wird mit gewissen Abstand zu den Eigenwerten zu wählen und gegebenfalls anzupassen.

4.2 Aufwand

Trotz der exponentiellen Konvergenzrate der Quadratur, ist die Berechnung der angenäherten Matrizen $A_{0,1}^{(m)}$ relativ aufwändig.
Für jeden Quadraturpunkt und jede Spalte von $\hat{V}$ muss ein lineares Gleichungssystem gelöst werden. Dies entspricht somit mj linearen Gleichungssystemen der Größe $N \times N$. Verwendet man eine LU-Zerlegung von $A(\lambda)$ und nehmen wir für diese den Aufwand $\mathcal{O}(N^3)$ an, so erhalten wir in etwa einen Gesamtaufwand zur Berechnung von $A_0^{(m)}$ und $A_1^{(m)}$ von $\mathcal{O}(m(N^3 + 2N^2 j))$.

5 Numerische Ergebnisse

In diesem Kapitel werden wir mit unserer persönlichen Implementierung das „Hadeler Problem“

$$T(z) = (e^z - 1)B_1 + z^2 B_2 - B_0$$
$$B_0 = b_0 I_n, \quad B_1 = (b_{jk}^{(1)}), \quad B_2 = (b_{jk}^{(2)})$$
$$b_{jk}^{(1)} = (n + 1 - \max(j,k))jk, \quad b_{jk}^{(2)} = n\delta_{jk} + 1/(j+k), \quad j,k = 1,\ldots,n,$$

aus [3] lösen, und die theoretischen Ergebnisse veranschaulichen.
Wir legen die Parameter mit $n = 200$, $b_0 = 100$ fest und wählen unsere Kurve als Kreis mit Mittelpunkt $\texttt{z} = -30$ und Radius $\texttt{R} = 11.5$. Als Toleranz zur Bestimmung der „korrekten“ Singulärwerte wählen wir $\texttt{tol} = 10^{-5}$.
Im ersten Testlauf plotten wir die berechneten Eigenwerte im Vergleich zu den zusätzlich aussortierten Werten. Zur besseren Veranschaulichung haben wir die Quadraturpunkte, über die wir summieren, auch miteingezeichnet. Wie man in Abbildung 5.1 sieht, befinden sich die, von unserem Algorithmus als „korrekt“ identifizierten, Eigenwerte alle innerhalb unserer Konturintegralkurve, während die zusätzlichen Werte außerhalb liegen.

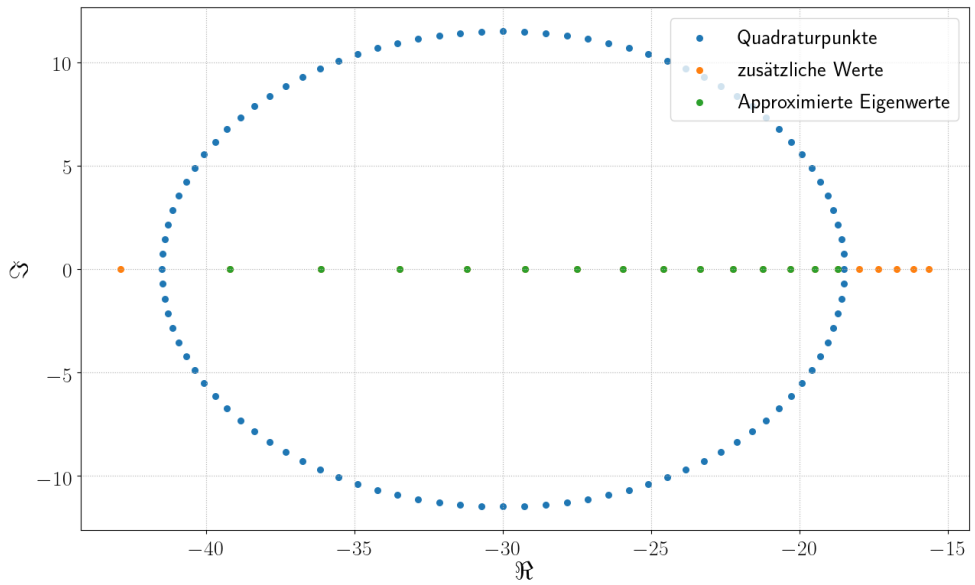

Abbildung 5.1: Approximierte Eigenwerte vs. zusätzliche Werte

5.1 Cut-Off

In Abbildung 5.2 sehen wir den Vergleich der Größenordnungen der eigentlichen Singulärwerte mit den zusätzlichen Singulärwerten. Nachdem hier ein deutlicher Abfall zu erkennen ist, ist es nicht schwer, die zusätzlichen Werte auszusortieren.

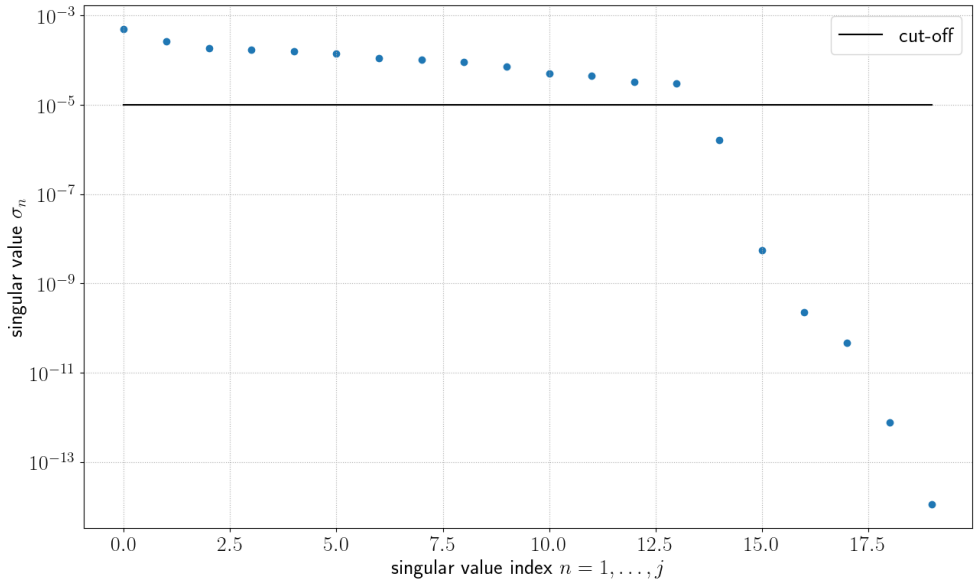

Abbildung 5.2: Echte vs. zusätzliche Singulärwerte

Eine Heuristik zur Überprüfung der Qualität eines approximierten Eigenwerts λ besteht darin, die Konditionszahl der jeweiligen Matrix $A(\lambda)$ zu betrachten. Da wir gegen eine singuläre Matrix konvergieren sollten, würden wir erwarten, dass die Konditionszahl gegen unendlich geht. In der Tat ist das Fall und man erkennt in Abbildung 5.3 sogar einen deutlichen Unterschied zwischen den echten und zusätzlichen Eigenwerten.

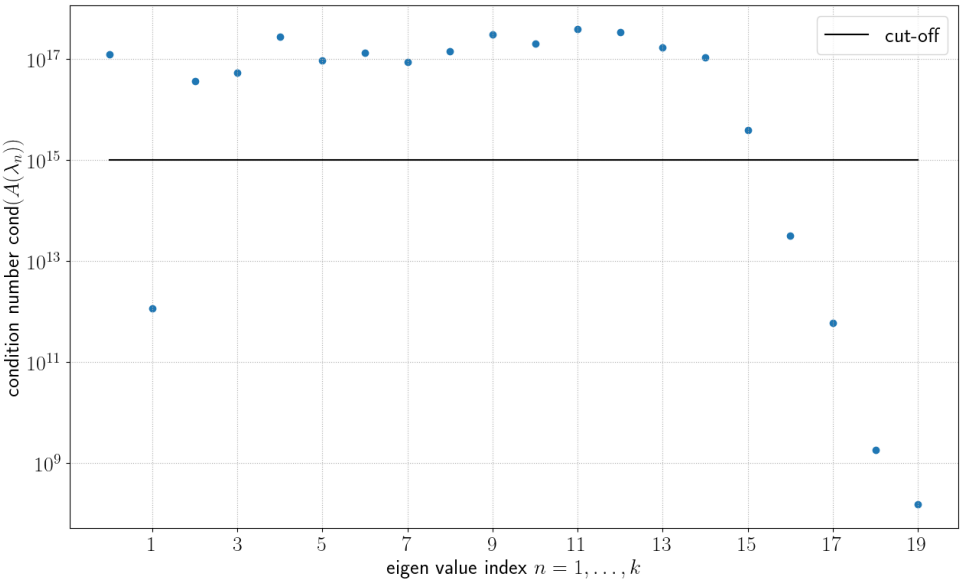

Abbildung 5.3: Konditionszahlen von $A(\lambda_n)$

5.2 Pitfalls

Schlussendlich wollen wir noch Situationen aufzeigen, an denen der Algorithmus scheitert. Die erste Möglichkeit dafür besteht darin, dass die Quadratur zu ungenau ist, und somit die „korrekten“ Singulärwerte nicht mehr von den zusätzlichen Werten zu unterscheiden sind. In Abbildung 5.4 sieht man, wie sich die approximierten Singulärwerte bei unterschiedlicher Anzahl von Quadraturknoten verhalten. Ab $\mathtt{m} \geq 20$ Quadraturknoten aufwärts ist bereits ein Sprung zwischen den echten und zusätzlichen Werten zu erkennen, der sich bei steigender Anzahl noch verdeutlicht.

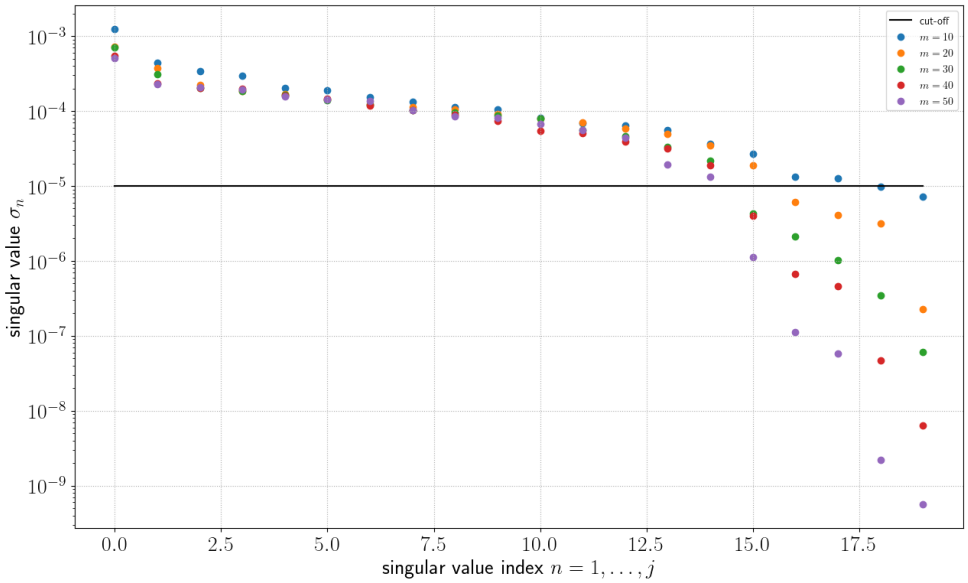

Abbildung 5.4: Singulärwerte in Abhängigkeit von der Anzahl der Quadraturknoten

Das zweite Hauptproblem ist, wenn die Zufallsmatrix zu klein gewählt wird. Sollte die Zufallsmatrix weniger Spalten haben, als wir Eigenwerte innerhalb unserer Kurve erwarten, könnte man ja noch hoffen, zumindest einen Teil der Eigenwerte korrekt approximieren zu können. Wie man in Abbildung 5.5 sieht, können wir in der Regel nicht einmal das erwarten.

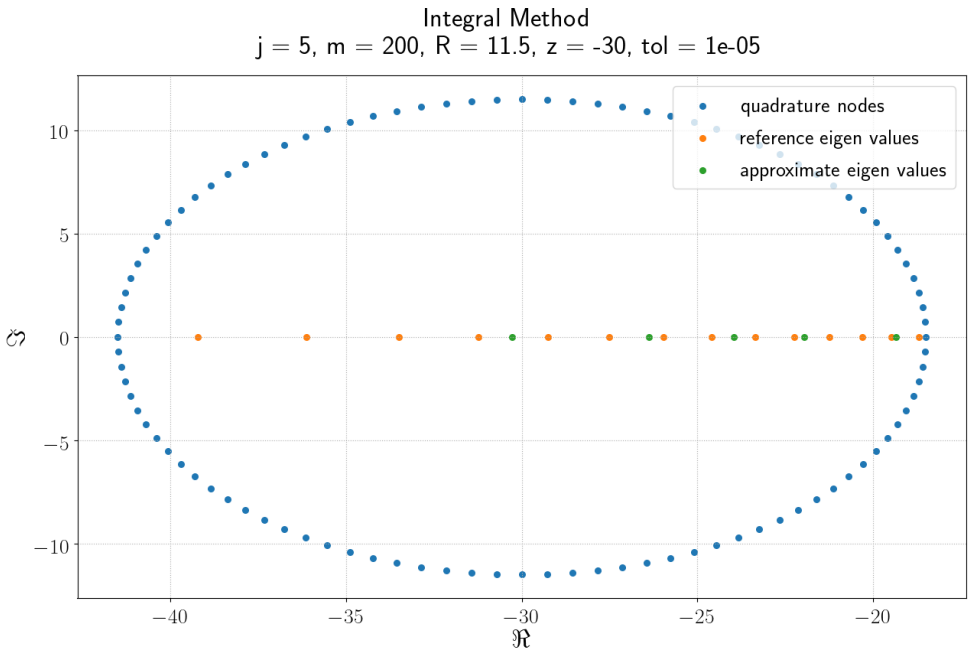

Abbildung 5.5: Zu kleine Zufallsmatrix

6 Fazit

Die Konturintegral-Methode ist ein Verfahren zur Lösung nichtlinearer Eigenwertprobleme, basierend auf dem „Satz von Keldysh“. Die zentrale Formel, die dem Algorithmus zu Grunde liegt ist

$$\frac{1}{2\pi i}\int_\Gamma f(\lambda)A(\lambda)^{-1}\,\mathrm{d}\lambda = \sum_{n=1}^{k} f(\lambda_n)\sum_{l=1}^{L_n} v_{n,l}w_{n,l}^*.$$

Um den Algorithmus durzuführen, müssen wir die Matrizen

$$\begin{aligned} A_0 &:= \frac{1}{2\pi i}\int_\Gamma \lambda^0 A(\lambda)^{-1}\hat{V}\,\mathrm{d}\lambda = VD^0W^*\hat{V} \in \mathbb{C}^{N\times j},\\ A_1 &:= \frac{1}{2\pi i}\int_\Gamma \lambda^1 A(\lambda)^{-1}\hat{V}\,\mathrm{d}\lambda = VD^1W^*\hat{V} \in \mathbb{C}^{N\times j} \end{aligned}$$

mit der Quadratur-Formel

$$\begin{aligned} Q(f) := \frac{1}{2\pi i}\int_{|\lambda|=R} f(\lambda)\,\mathrm{d}\lambda \approx Q_m(f) := \frac{R}{m}\sum_{\nu=0}^{m-1}\omega_m^\nu f(R\omega_m^\nu),\\ \omega_m := \exp\frac{2\pi i}{m},\quad R>0,\quad m\in\mathbb{N},\quad f\in H(\mathbb{C}), \end{aligned}$$

approximieren, die reduzierte Singulärwertzerlegung

$$A_0 = \tilde{V}\Sigma\tilde{W}^*$$

bestimmen und das lineare Eigenwertproblem der Matrix

$$\tilde{V}^*A_1\tilde{W}\Sigma^{-1} \sim D = \mathrm{diag}(\lambda_1,\ldots,\lambda_k)$$

(beispielsweise mit dem QR-Verfahren) lösen.
Die Integralmethode zur Berechnung der Eigenwerte nichtlinearer Eigenwertprobleme ist vergleichsweise neu und Gegenstand aktueller Forschung. Die Methode hat die Lösung von nichtlinearen Eigenwertproblemen deutlich vereinfacht, dennoch birgt sie auch wesentliche Probleme. Neben dem relativ hohen Aufwand wird durch die Verwendung von $A_{0,1}^{(m)}$ die Anzahl der nicht-verschwindenden Singulärwerte deutlich erhöht. Solange die „korrekten“

Singulärwerte um Größenordnungen über den zusätzlichen liegen, können letztere einfach aussortiert werden. Sind die Approximationen von A_0, A_1 nicht genau genug, kann es passieren, dass die „korrekten“ Singulärwerte kaum mehr von den zusätzlichen Singulärwerten zu unterscheiden sind, und der Algorithmus eventuell falsche Resultate liefert.
Dennoch hat die Integralmethode die Lösung von nichtlinearen Eigenwertproblemen massiv vereinfacht. Sie ermöglicht bei entsprechender Genauigkeit die zuverlässige Berechnung aller Eigenwerte innerhalb einer gewählten Kurve. Dies ist ein sehr großer Vorteil zu iterativen Verfahren, bei denen typischerweise die Lage der gefundenen Eigenwerte nicht zuverlässig kontrolliert werden kann.

7 Anhang

Hier wollen wir den wesentlichen Backend-Code explizit auflisten. Dieser baut auf den Paketen `numpy` (als `np`), und `linalg` von `scipy` auf.

7.1 Quadraturformeln

```
# quadrature node
omega = lambda m: np.exp(2 * np.pi * 1j / m)

# quadrature on 0-centered ball with radius R
Q_zero = lambda m, f, R: R / m * sum([
    omega(m) ** nu * f(R * omega(m) ** nu)
    for nu in range(m)
])

# quadrature on z-centered ball with radius R
Q = lambda m, f, R, z: Q_zero(m, lambda x: f(x + z), R)
```

7.2 Konturintegral-Methode

```
def integral_method(A, N, j, m, R, z, tol, debug = False):

    """
    A ... matrix-function
    N ... number of rows/columns of A(lambda)
    j ... number of expected eigen values in ball B_R(z)
    m ... number of quadrature nodes
    R ... ball-radius
    z ... ball-center
    tol ... tolerance for SVD reduction
    """

    # random matrix
    V_hat = np.random.random((N, j))

    # ------------------------ #
    # step 1:
    # calculate A_0 and A_1
```

```
# integrand of A_0 and A_1 (with index 0 resp. 1)
def integrand(index, lamda):

    LU, piv = linalg.lu_factor(A(lamda))

    return lamda ** index * np.array([
        linalg.lu_solve((LU, piv), V_hat[:, i])
        for i in range(j)
    ]).T

f_0 = lambda lamda: integrand(0, lamda)
f_1 = lambda lamda: integrand(1, lamda)

# apply quadrature formula to integrand for A_0 and A_1
A_0 = Q(m, f_0, R, z)
A_1 = Q(m, f_1, R, z)

# ------------------------ #
# step 2:
# calculate SVD
# reduce to J singular values

# get full i.e. unreduced SVD
V_tilde_full, Sigma_full, W_tilde_full = linalg.svd(
    A_0,
    full_matrices = False
)

# mask for SVD reduction (kill zero values)
mask = np.abs(Sigma_full) > tol

# apply mask i.e. reduce SVD
Sigma_reduced = Sigma_full[mask]
V_tilde_reduced = V_tilde_full[:, mask]
W_tilde_reduced = W_tilde_full[mask, :]

# ------------------------ #
# step 3:
# calculate eigen values of linearized problem

matrices = (
    V_tilde_reduced.conj().T,
    A_1,
    W_tilde_reduced.conj().T,
```

```
        np.diag(Sigma_reduced ** (-1))
    )

    eigen_values = linalg.eigvals(
        matrices[0] @ matrices[1] @ matrices[2] @ matrices[3]
    )

    if debug:
        return {
            'eigen_values': eigen_values,
            'V_hat': V_hat,
            'integrand': integrand,
            'f_0': f_0,
            'f_1': f_1,
            'A_0': A_0,
            'A_1': A_1,
            'V_tilde_full': V_tilde_full,
            'Sigma_full': Sigma_full,
            'W_tilde_full': W_tilde_full,
            'V_tilde_reduced': V_tilde_reduced,
            'Sigma_reduced': Sigma_reduced,
            'W_tilde_reduced': W_tilde_reduced
        }
    else:
        return eigen_values
```

Literatur

[1] Wolf-Jürgen Beyn. „An integral method for solving nonlinear eigenvalue problems“. In: *Linear Algebra and its Applications* 436.10 (2012). Special Issue dedicated to Heinrich Voss's 65th birthday, S. 3839–3863. ISSN: 0024-3795. DOI: https://doi.org/10.1016/j.laa.2011.03.030. URL: http://www.sciencedirect.com/science/article/pii/S0024379511002540.

[2] Lothar Nannen. „Eigenwertprobleme“. 2020.

[3] Yousef Saad, Mohamed El-Guide und Agnieszka Miedlar. *A rational approximation method for the nonlinear eigenvalue problem*. 2020. arXiv: 1901.01188 [math.NA]. URL: https://arxiv.org/abs/1901.01188.