ANWENDUNG VON PSEUDOSPEKTREN IN DER REGELUNGSTECHNIK

MORITZ WOLTER

Technische Universität Hamburg-Harburg

Eine Bachelorarbeit

Institut für Mathematik
bei Prof. Dr. Marko Lindner
und Dr. Karsten Kruse
19. August 2014 – Version 1.0

„Et ignem regunt numeri.

Même le feu est régi par les nombres.“

(Selbst das Feuer wird von den Zahlen beherrscht.)

Jean Baptiste Joseph Fourier (1768-1830) nach einem Ausspruch Platos (428-348 v.Chr).

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LISTINGS

ACRONYMS

parfor parallel for Schleife

Matlab matrix laboratory

SISO Single-Input Single-Output

Teil I

GRUNDLAGEN

1 EIGENWERTE UND DIE SPEKTRALTHEORIE

1.1 WAS SIND EIGENWERTE?

Multipliziert man die Matrix $\mathbf{A}$ mit den Vektoren $\mathbf{x_1}$ und $\mathbf{x_2}$ [1]:

$$\mathbf{A} = \begin{pmatrix} 1 & 5 \\ 0 & 2 \end{pmatrix}, \ \mathbf{x_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ \mathbf{x_2} = \begin{pmatrix} 5 \\ 1 \end{pmatrix},$$

dann lässt sich bei den Produkten $\mathbf{Ax_1}$ und $\mathbf{Ax_2}$ ein interessantes Phänomen beobachten. Es gilt:

$$\mathbf{Ax_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ \mathbf{Ax_2} = \begin{pmatrix} 10 \\ 2 \end{pmatrix}.$$

Bei näherem Hinsehen stellt sich heraus, dass sich der Vektor $\mathbf{x_1}$ trotz Multiplikation mit der Matrix nicht verändert hat, die Einträge des zweiten Vektors haben sich verdoppelt. Man könnte also stattdessen schreiben:

$$\mathbf{Ax_1} = 1\mathbf{x_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ \mathbf{Ax_2} = 2\mathbf{x_2} = \begin{pmatrix} 10 \\ 2 \end{pmatrix}.$$

In diesem Fall lässt sich die Multiplikation mit einer Matrix durch Multiplikation mit einem einfachen Skalar ersetzen. Diese Skalare werden Eigenwerte genannt. Die Vektoren, bei denen eine solche Gleichung gilt, werden Eigenvektoren genannt. Die Nutzung des oben beobachteten Phänomens bildet das Fundament der Theorie der Eigenwerte, auch Spektraltheorie genannt. Das oben Gesehene lässt sich mathematisch zusammenfassen:

Das Nomen „Skalar“bedeutet einfache Zahl. Gewöhnliche Zahlen werden so von Vektoren oder Matrizen sprachlich klar getrennt.

$$\mathbf{Ax} = \lambda\mathbf{x}. \tag{1}$$

Aus dieser Formel lassen sich zwei weitere Gleichungen zur Berechnung von Eigenwerten und Vektoren herleiten. Bringt man $\lambda\mathbf{x}$ auf die linke Seite und klammert $\mathbf{x}$ aus, so kommt heraus:

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0. \tag{2}$$

Der Vektor $\mathbf{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ löst die Gleichung. Da diese Lösung aber unabhängig von der Fragestellung immer gilt und daher keine neuen

1 Einleitung nach Bronson [4] Seite 177

Informationen beisteuert, ist sie trivial und wird in der Praxis ausgeschlossen. Um nun ohne den Nullvektor das Problem noch lösen zu können, muss also ein Eigenvektor $\mathbf{x}$ die Spalten von $\mathbf{A} - \lambda\mathbf{I}$ zum Nullvektor kombinieren. Anders formuliert muss eine Spalte von $\mathbf{A} - \lambda\mathbf{I}$ eine Linearkombination der anderen Spalten von $\mathbf{A} - \lambda\mathbf{I}$ sein, die Determinante ist in diesem Fall immer 0. [2] Aus dieser Idee kommt die Gleichung zur Berechnung der Eigenwerte [3]:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0. \tag{3}$$

Was erst einmal wie eine Kuriosität anmutet, wird in der angewandten Mathematik an vielen Stellen verwendet. Denn Eigenvektoren sind zusammengefasst in einer Matrix in der Lage, ihre Ausgangsmatrix in einer Ähnlichkeitstransformation zu diagonalisieren [4]. Im vorausgehenden Beispiel ergibt sich die Matrix $\mathbf{S}$ aus den Vektoren $\mathbf{x_1}$ und $\mathbf{x_2}$:

Eine Ähnlichkeitstransformation ist eine Abbildung der Form $\mathbf{B} = \mathbf{C}^{-1}\mathbf{AC}$.

$$\mathbf{x_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ \mathbf{x_2} = \begin{pmatrix} 5 \\ 1 \end{pmatrix}, \ \mathbf{S} = \begin{pmatrix} 1 & 5 \\ 0 & 1 \end{pmatrix}.$$

Die Matrix $\mathbf{S}^{-1}\mathbf{AS}$ lässt sich nun ausrechnen:

$$\mathbf{S}^{-1}\mathbf{AS} = \begin{pmatrix} 1 & -5 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 5 \\ 0 & 2 \end{pmatrix} * \begin{pmatrix} 1 & 5 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}.$$

Diese diagonale Matrix heißt Λ. Mathematisch lässt sich der oben beobachtete Prozess am eingängigsten über Gleichung 1 nachvollziehen. Zuallererst multipliziert man $\mathbf{A}$ mit ihrer Eigenvektormatrix $\mathbf{S}$ [5]. $\mathbf{A}$ wird jetzt mit den Eigenvektoren in den Spalten von $\mathbf{S}$ multipliziert:

$$\mathbf{AS} = \mathbf{A} \begin{pmatrix} \mathbf{x_1} & \dots & \mathbf{x_n} \\ & & \end{pmatrix}.$$

Diese Matrix-Multiplikation lässt sich auch als eine Vielzahl von Matrix-Vektor-Multiplikationen schreiben. Für diese gilt jeweils $\mathbf{A}x_i = \lambda_i x_i$:

$$\mathbf{AS} = \begin{pmatrix} \mathbf{Ax_1} & \dots & \mathbf{Ax_n} \\ & & \end{pmatrix} = \begin{pmatrix} \lambda_1\mathbf{x_1} & \dots & \lambda_n\mathbf{x_n} \\ & & \end{pmatrix}.$$

2 Kleine Enzyklopädie Mathematik [23] Seite 392. Es werden zwei Aussagen benötigt, Zitat:

> „3. Die Determinante hat den Wert Null, wenn eine Zeile eine Linearkombination anderer Zeilen ist.
>
> 5. Die Determinante ändert ihren Wert nicht, beim Vertauschen der Zeilen mit den Spalten. "

3 Strang [25] Seite 287
4 Kleine Enzyklopädie Mathematik [23] Seite 414
5 Strang [25] Seite 298

Jeder Eigenvektor wird jetzt von seinem Eigenwert skaliert. Den gleichen Effekt erhält man, wenn man die Eigenwerte in eine rechtsseitige Diagonalmatrix schreibt:

$$\begin{pmatrix} \lambda_1 \mathbf{x_1} & \dots & \lambda_n \mathbf{x_n} \end{pmatrix} = \begin{pmatrix} \mathbf{x_1} & \dots & \mathbf{x_n} \end{pmatrix} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} = \mathbf{S}\Lambda.$$

Damit ist $\mathbf{A}$ vollständig diagonalisiert. Allgemein gilt also $\mathbf{AS} = \mathbf{S}\Lambda$, $\mathbf{A} = \mathbf{S}\Lambda\mathbf{S}^{-1}$ oder nach Λ aufgelöst:

$$\Lambda = \mathbf{S^{-1}AS}. \tag{4}$$

Die Matrix Λ gilt in vielen Anwendungen als Lösung oder Teil der Lösung eines Problems, da sich an ihrer diagonalen Form Lösungen ablesen lassen. Es folgt ein Beispiel, in dem die Matrix Λ eine wichtige Rolle spielt.

1.1.1 *Hauptachsen-Transformationen in der Mechanik*

In der Mechanik spielen Hauptachsen-Transformationen eine wichtige Rolle. Eine solche Transformation soll hier nun am Beispiel eines hypothetischen elliptischen Objektes durchgeführt werden[6]:

$$ax^2 + 2bxy + cy^2 + d = 0. \tag{5}$$

In Matrixschreibweise lässt sich diese Gleichung in der Form $\mathbf{x}^T * \mathbf{A} * \mathbf{x} + d = 0$ schreiben[7]:

$$\mathbf{A} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

$$\begin{pmatrix} x & y \end{pmatrix} * \begin{pmatrix} a & b \\ b & c \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix} + d = 0.$$

Für $a = 3$, $b = -1$, $c = 3$, $d = -2$ ergibt sich:

$$\begin{pmatrix} x & y \end{pmatrix} * \begin{pmatrix} 3 & -1 \\ -1 & 3 \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix} - 2 = 0.$$

Mithilfe der Matrix A lässt sich nun die Transformation vornehmen. Die Eigenwerte lassen sich mit $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$ bestimmen. Man erhält die Werte $\lambda_1 = 2$ und $\lambda_2 = 4$. In einem nächsten Schritt werden mit $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0$ die Eigenvektoren bestimmt. Die Wahl orthogonaler Eigenvektoren führt auf ein rechtwinkliges Koordinatensystem. Da der Faktor 2b aus 5 zu gleichen Teilen auf b in A_{12} und b in A_{21} in der Matrix $\mathbf{A}$ aufgeteilt wird, ist diese Matrix symmetrisch. Symmetrische Matrizen haben reelle Eigenwerte und ihre Eigenvektoren

6 Kleine Enzyklopädie Mathematik [23] Seite 414
7 Hütte [9] Seite A21

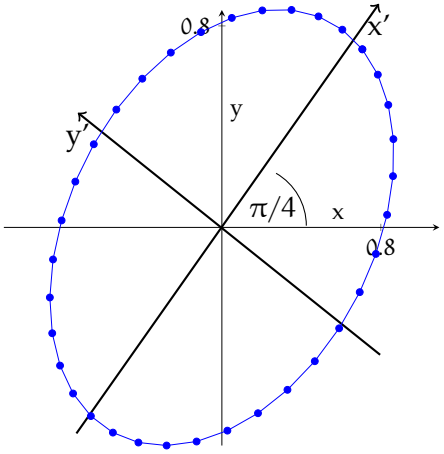


Abbildung 1: Hauptachsentransformation mit Eigenvektoren $\mathbf{x_1} = \mathbf{x}'$ und $\mathbf{x_2} = \mathbf{y}'$. Nach: Kleine Enzyklopädie Mathematik [23] Seite 414.

können immer orthogonal gewählt werden[8], wenn $\lambda_1 \neq \lambda_2$ sind sie sogar automatisch orthogonal. In diesem Beispiel ergibt sich für die Eigenvektoren:

$$x_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \; x_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Die Länge beider Vektoren ist in diesem Beispiel $\sqrt{2}$. Normiert man beide Vektoren, in diesem Fall durch Multiplikation mit $1/\sqrt{2}$ und packt sie in eine Matrix, dann erhält man:

$\mathbf{S}$ beschreibt eine Drehung um 45°, denn $\frac{1}{\sqrt{2}} = \cos(\pi/4) = \sin(\pi/4)$

$$\mathbf{S} = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}, \quad \mathbf{S}^{-1} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}.$$

Jetzt kommt der wichtigste Schritt der Transformation: Die Diagonalisierung mithilfe der Ähnlichkeitstransformation $\mathbf{A}' = \mathbf{S}^{-1}\mathbf{A}\mathbf{S}$:

$$\mathbf{A}' = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}.$$

Da die Eigenvektoren rechtwinklig und normiert gewählt wurden, ist die Matrix $\mathbf{S}$ orthogonal. Es gilt $\mathbf{S}^T = \mathbf{S}^{-1}$.

Damit ist das Problem gelöst. Die Gleichung für die Ellipse im neuen gedrehten Koordinatensystem ergibt sich aus $\mathbf{x}^T\mathbf{S} * \mathbf{A}' * \mathbf{S}^T\mathbf{x} + d = 0$, es gilt $\mathbf{x}^T\mathbf{S} = \mathbf{x'}^T$ und $\mathbf{S}^T\mathbf{x} = \mathbf{x}'$:

$$2x'^2 + 4y'^2 = 2 \;\text{ mit } \mathbf{x}' = \begin{pmatrix} x' \\ y' \end{pmatrix}.$$

In der Festigkeitslehre lassen sich mithilfe von Hauptachsentransformation Belastungen eines Objektes besser einschätzen, für die korrekte Berechnung der Normalspannung ist die Kenntnis der Hauptachsen sogar Vorraussetzung [9].

8 Strang [25] Seite 330
9 Hibbler [16], Seite 310

In der Dynamik rotierender Körper vereinfacht die Verwendung eines Hauptachsen-Koordinatensystems die Berechnungen deutlich [10], da bei Betrachtung in einem Hauptachsen-Koordinatensystem der Trägheitstensor diagonal wird, es verschwinden also die Trägheitsprodukte. Hier muss allerdings auch die Massenverteilung mit einbezogen werden, denn das Trägheitsmoment ist keine rein geometrische Eigenschaft eines Objektes mehr.

Trägheitsprodukte sind alle nichtdiagonalen Einträge des Trägheitstensors. Diese Einträge beziehen sich auf zwei verschiedene Achsen z.B. I_{xy} *oder* I_{xz}*, nicht aber* I_{xx} *oder* I_{yy}.

1.2 HISTORISCHER URSPRUNG DES THEMAS

Historisch betrachtet finden sich im Wesentlichen zwei Fachgebiete, die zur Entstehung der Spektraltheorie geführt haben. Erstens die geometrische Betrachtung von Hauptachsen. Diese finden sich in der Dynamik, spielen aber auch bei der Behandlung von elliptischen Bahnen in der Himmelsmechanik eine Rolle. Zweitens die Lösung von linearen Differentialgleichungssystemen. Im Zentrum steht hier die physikalisch bedeutsame Systemstabilität. Eigenwerte spielten und spielen bei Stabilitätsfragen eine große Rolle [11].

Beim Lösen von Differentialgleichungen treten Terme der Form $e^{\lambda t}$ *auf. Diese sind nur stabil, wenn* $\lambda \leqslant 0$ *und reell ist.*

Im 18. Jahrhundert entdeckte Leonhard Euler (April 1707 - September 1783) [12] während seines Wirkens an den wissenschaftlichen Akademien in Berlin und Sankt Petersburg als erster die Bedeutung der Hauptachsen bei der Rotation starrer Körper [13]. Auch an anderer Stelle traten im 18. Jahrhundert Eigenwerte auf. Zum Beispiel in Arbeiten von D'Alembert sowie von Jean und Daniel Bernoulli [14], die sich mit Schwingungen beschäftigten. Allerdings wurde damals noch nicht im Kontext einer Theorie der Eigenwerte gearbeitet. Die heutzutage geläufige Matrix-Notation war ebenso unbekannt, wie auch die Bezeichnung Eigenwert [15].

Maßgebliche Entwicklungen fanden um 1815 an der École Polytechnique in Paris statt. Augustin-Louis Cauchy (August 1789 - Mai 1857) [16] und andere entdeckten alle wesentlichen Eigenschaften von Determinanten, die zum Schaffen der Spektraltheorie notwendig waren [17]. Neue Probleme im Bezug auf die Lösung von Differentialgleichungssystemen entstanden aus Arbeiten von Jean Baptiste Joseph Fourier (März 1768 - Mai 1830) [18]. In seinem Werk Théorie analytique de la chaleur löste er die Wärmeleitungsgleichung auf einer hohlen Kugel, ein Problem bei dessen Lösung Eigenwerte auftreten.

10 Hibbler [15] Seite 548, Stichwort: Inertia Tensor
11 Artmann [2] Seite 4
12 Gillispie [1] Band 3&4 Seite 467
13 Hibbler [15] Seite 568
14 Hawkins [14] Seite 3
15 Hawkins [14] Seite 2
16 Gillispie [1] Band 3&4 Seite 131
17 Wieleitner [31] Seite 62
18 Gillispie [1] Band 5&6 Seite 93

Es war aber nicht bewiesen, dass diese Eigenwerte immer reell sind und komplexe Eigenwerte wären physikalisch unmöglich gewesen [19].

Ähnliche Probleme warfen Gleichungen auf, die zur Beschreibung himmelsmechanischer Vorgänge neu entworfen worden waren. Hier war man mit dem Hinweis, instabile Planetenbahnen gäbe es in der Realität nicht, von reellen Eigenwerten ausgegangen. Einen rein mathematischen Beweis gab es noch nicht.

Vor diesem Hintergrund gelang es Cauchy 1829 auf rein mathematischem Wege zu beweisen, dass symmetrische Matrizen reelle Eigenwerte haben. Er konnte auch zeigen, dass sich Matrizen durch lineare Transformationen diagonalisieren lassen [20]. Die Bezeichnung Matrix nutzte Cauchy noch nicht, er sprach stattdessen von Systemen.

In Cambridge machte nun Arthur Cayley (August 1821 - Januar 1895) weitere theoretische Fortschritte beim Rechnen mit Matrizen. Er entwickelte die auch heute noch gebräuchliche Matrixnotation [21].

$\mathbf{AX} = \lambda\mathbf{BX}$ *ist das generalisierte Eigenwertproblem.*

Karl Weierstrass (Oktober 1815- Februar 1897) ging von der Differentialgleichung $\mathbf{B\ddot{Y}} = \mathbf{AY}$ aus. Er erhielt $\mathbf{AX} = \lambda\mathbf{BX}$ und konnte beweisen, dass man **B** nach **I** transformieren und gleichzeitig **A** in Diagonalform bringen kann [22].

David Hilbert (Januar 1862 - Februar 1943) schließlich bezog die neue Theorie, die schon als Spektraltheorie gelten darf, auch auf Operatoren. Im Zusammenhang mit Operatoren erfand er die Bezeichnung Eigenwert [23]:

> „Wir führen hier noch folgende Bezeichnungen ein: Die Nullstellen von $\delta(\lambda)$ mögen die zum Kern $K(s,t)$ gehörigen Eigenwerte heißen. “

Im Bezug auf Matrizen werden diese Bezeichungen im deutschen und englischen Sprachraum (Eigenvalue) bis heute verwendet.

19 Gillispie [1] Band 5&6 Seite 97
20 Hawkins [14] Seite 2
21 Wieleitner [31] Seite 63
22 Hawkins [13] Seite 5
23 David Hilbert [17] Seite 64

2

DAS PSEUDOSPEKTRUM

Die Spektraltheorie ist aus mindestens drei Gründen nützlich. Erstens ist sie aus der algorithmischen Perspektive hilfreich, weil sie wie in Teil 1.1.1 beschrieben, Diagonalisieren ermöglicht. So kann die Problemlösung beschleunigt werden. Zweitens lassen Eigenwerte aus physikalischer Perspektive Rückschlüsse auf das Verhalten des betrachteten Systems für $t \to \infty$ zu, diese wesentliche Eigenschaft hatte zur Entdeckung der Eigenwerte beigetragen. Schließlich sind Eigenwerte für Menschen von großer psychologischer Bedeutung, da sie es ermöglichen, abstrakte Matrixgebilde graphisch darzustellen. Unser Gehirn ist gut im Umgang mit visualisierten Informationen. Eigenwerte lassen uns diese Fähigkeit im Bezug auf Matrizen nutzen [1].

Eigenwertanalysen sind zwar hilfreich, aber nur auf normale Matrizen immer zuverlässig anwendbar. Die Menge der normalen reellen Matrizen umfasst die orthogonalen, die symmetrischen und die antisymmetrischen Matrizen [2]. Orthogonale Matrizen haben linear unabhängige Spalten, für sie gilt: $\mathbf{A}\mathbf{A}^T = \mathbf{A}^T\mathbf{A} = \mathbf{I}$ oder $\mathbf{A}^T = \mathbf{A}^{-1}$. Symmetrische Matrizen sind an der Diagonalen spiegelbar, für sie gilt $\mathbf{A} = \mathbf{A}^T$. Bei antisymmetrischen Matrizen ist die transponierte auch die negative Matrix, es gilt $-\mathbf{A} = \mathbf{A}^T$. Formal ist ein Problem dann normal, wenn [3]:

$$\mathbf{A}\overline{\mathbf{A}}^T = \overline{\mathbf{A}}^T\mathbf{A}.$$

Symmetrische Matrix im $\mathbb{R}^2$: $\begin{pmatrix} a & b \\ b & c \end{pmatrix}$.

Antisymmetrische Matrizen im $\mathbb{R}^2$ haben die Form $\begin{pmatrix} 0 & a \\ -a & 0 \end{pmatrix}$.

$\overline{\mathbf{A}}$ bezeichnet das komplexe Konjugat einer Matrix. $\mathbf{A}^T$ die transponierte Matrix. Es gilt $\mathbf{A}^* = \mathbf{A}^H = \overline{\mathbf{A}}^T$

Probleme entstehen immer dann, wenn die betrachtete Matrix nichtnormal ist. Im nichtnormalen Fall lässt sich keine orthogonale Eigenvektor-Basis finden. Dann verzerrt der Eigenraum das Problem. Das Pseudospektrum hilft dabei, die Folgen der Nichtnormalität abzuschätzen. Wie sich später herausstellen wird, liefert das Pseudospektrum jedoch auch zusätzliche Informationen über das Einschwungverhalten eines Systems. Die Eigenwerte der Systemmatrix liefern das Verhalten für $t \to \infty$. Das Pseudospektrum ermöglicht es, zusätzlich Ober- und Untergrenzen zum transienten Verhalten festzulegen [4].

Ein letztes Argument für Pseudospektren sind Fälle, in denen sich das klassische Spektrum nicht berechnen lässt. Bei unendlich großen Matrizen wird die klassische Spektralanalyse schwierig bis unmöglich.

1 Trefethen und Embree [26] Seite 1
2 Strang [25] Seite 341
3 Strang [25] Seite 341
4 Trefethen und Embree[27] Seite 150 und 151

Hier lässt sich jedoch mit Pseudospektren auf das Spektrum rückschließen [5].

Pseudospektren generalisieren klassische Spektren. Sie vergrößern deren Anwendbarkeit. Der Schritt vom Spektrum zum Pseudospektrum ist im Grunde nicht besonders groß. Für klassische Spektren gilt die Gleichung [6]:

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0. \tag{6}$$

Die Null steht hier für den Nullvektor, denn $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x}$ liefert keinen Skalar, sondern einen Vektor. Es ist prinzipiell aber auch möglich, über eine Normierung einen Skalar zu verlangen:

$$\|(\mathbf{A} - \lambda\mathbf{I})\mathbf{x}\| = 0. \tag{7}$$

Eine Mehrheit der Autoren der Pseudospektral-Literatur verwendet z für Elemente des Pseudospektrums σ_ϵ. λ bezeichnet damit ausschließlich Eigenwerte $\in \sigma$.

Um jetzt zum Pseudospektrum zu kommen, ersetzt man einfach die Null durch ein $\epsilon > 0$ und das Gleichheitszeichen durch ein „kleiner als“[7]:

$$\|(\mathbf{A} - z\mathbf{I})\mathbf{x}\| < \epsilon \quad \text{mit } \|\mathbf{x}\| = 1. \tag{8}$$

Jetzt kommt als Lösung keine Menge von endlich vielen Punkten mehr heraus, sondern zu jedem denkbaren ϵ eine Fläche. Zur Berechnung des Pseudospektrums wird nicht $\det(\mathbf{A} - z\mathbf{I}) < \epsilon$ verwendet, wie man vielleicht vermuten würde. Auch Eigenwerte werden in der Praxis nicht über $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$ gefunden, denn das Lösen von Determinanten und dem sich anschließenden Nullstellenproblem dauert selbst am Computer deutlich zu lange (es sei denn, die Dimension des Problems ist sehr klein) [8]. Besser lässt sich mit der sogenannten Resolvente $(\mathbf{A} - z\mathbf{I})^{-1}$ rechnen. Eigenwerte sind genau da, wo die Resolvente nicht existiert, ihre Norm läuft hier gegen Unendlich. Die Definition des Pseudospektrums lässt sich jetzt mit der Resolvente im Hinterkopf umschreiben in [9]:

$$\|(\mathbf{A} - z\mathbf{I})^{-1}\| > \epsilon^{-1}. \tag{9}$$

Es gibt verschiedene Matrixnormen. Hier wird implizit immer die 2-Norm $\|\cdot\|_2$ verwendet. Sie lässt sich mit der Formel $\|\mathbf{A}\|_2 = \max_x \frac{\|\mathbf{Ax}\|_2}{\|x\|_2}$ berechnen. Sie entspricht dem größten Singulärwert.

Diese Beziehung ist in Abbildung 2 für die folgende Matrix $\mathbf{A}$ dargestellt:

$$\mathbf{A} = \begin{pmatrix} 1+i & 1 & 0 \\ 0 & 1-i & 3 \\ 0 & 0 & -1 \end{pmatrix}$$

5 Man nutzt aus, dass $\sigma(\mathbf{A})$ nicht stetig von $\mathbf{A}$ abhängt, aber $\sigma_\epsilon(\mathbf{A})$ schon (auch von ϵ). Daraus folgt, dass $\sigma_\epsilon(\mathbf{A_n}) \xrightarrow{n\to\infty} \sigma_\epsilon(\mathbf{A}) \xrightarrow{\epsilon\to 0} \sigma(\mathbf{A})$ zur Approximation von $\sigma(\mathbf{A})$ mittels $\mathbf{A_n} \to \mathbf{A}$ bei unendlich großen Matrizen verwendet werden kann.

6 Strang [25] Seite 286

7 Trefethen und Embree [27] Seite 16, Third definition of pseudospectra

8 Strang [25] Seite 486

9 Trefethen und Embree [27] Seite 13, First definition of pseudospectra

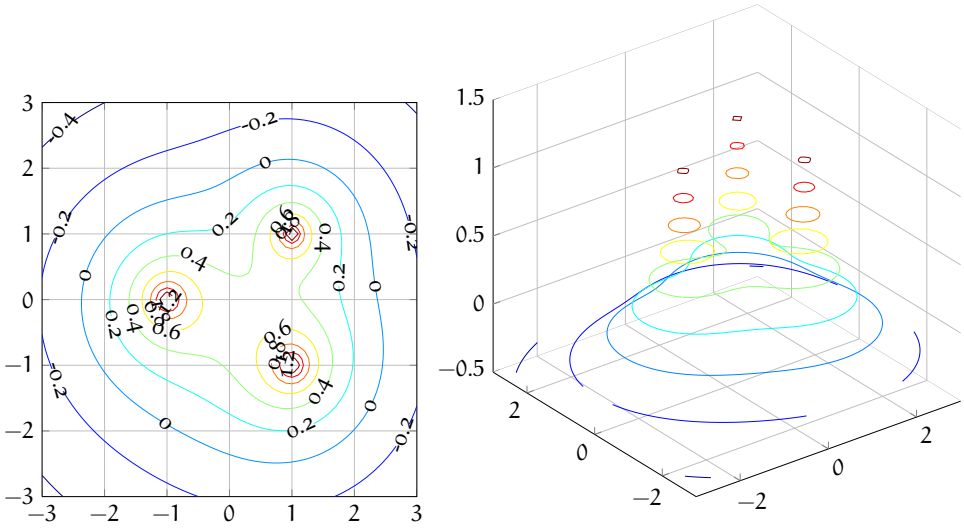


Abbildung 2: Plot des Pseudospektrums der Matrix $\mathbf{A}$ logarithmisch skaliert in der komplexen Ebene, als Draufsicht (links) und in 3 Dimensionen (rechts).

Da $\mathbf{A}$ obere Dreiecksform hat, stehen die Eigenwerte auf der Diagonalen. Sie lauten also $\lambda_1 = 1+i$, $\lambda_2 = 1-i$, $\lambda_3 = -1$. Im Plot 2 erkennt man deutlich, dass die Resolvente hier ihre Maximalwerte annimmt.

Pseudospektren lassen sich jedoch auch auf andere äquivalente Weise definieren. Beispielsweise ist auch eine Definition über zufällige Störungen möglich. In diesem Fall bezeichnet σ die Menge aller Eigenwerte und σ_ϵ das Pseudospektrum. $\sigma_\epsilon(\mathbf{A})$ ist die Menge der Zahlen $z \in \mathbb{C}$, so dass [10]:

$$z \in \sigma(\mathbf{A}+\mathbf{E}). \tag{10}$$

für eine Matrix $\mathbf{E}$ mit $\|\mathbf{E}\| < \epsilon$ gilt. Diese Definition suggeriert eine andere Art, Pseudospektren zu plotten. Im Grunde reicht es, die Eigenwerte der wiederholt zufällig gestörten Matrix $\mathbf{A}$ zu plotten. Die Normen der Störungen dürfen ϵ nicht überschreiten, also gilt $\|\mathbf{E}\| < \epsilon$. Man plottet also [11]:

$$\sigma_\epsilon(\mathbf{A}) = \bigcup_{\|\mathbf{E}\|<\epsilon} \sigma(\mathbf{A}+\mathbf{E}). \tag{11}$$

Die in diesem Abschnitt verwendeten Definitionen von Pseudospektren bezeichnen alle dasselbe Konzept, sie sind `äquivalent`, es gilt: Definition 8 ⇔ Definition 9 ⇔ Definition 10 ⇔ Definition 12.

Dies ist in Plot 3 für die schon bekannte Matrix $\mathbf{A}$ mit $\epsilon = 10^{0.4}$ und $\epsilon = 10^{0.8}$ geschehen. Es ist gut zu erkennen, dass die gestörten Eigenwerte das zugehörige ϵ-Pseudospektrum nicht verlassen.
Abbildung 3 erlaubt es, die Anfälligkeit von Matrizen gegenüber Störungen einzuschätzen. Je weitläufiger sich die gestörten Eigenwerte verteilen, desto sensitiver ist die Matrix. σ_ϵ misst also die Sensitivität des Spektrums $\sigma(\mathbf{A})$ bezüglich Störungen von $\mathbf{A}$ mit $\|\mathbf{E}\| < \epsilon$.

10 Trefethen und Embree [27] Seite 16, Second definition of pseudospectra
11 Trefethen und Embree [27] Seite 378, Stichwort: Poor man's pseudospectrum

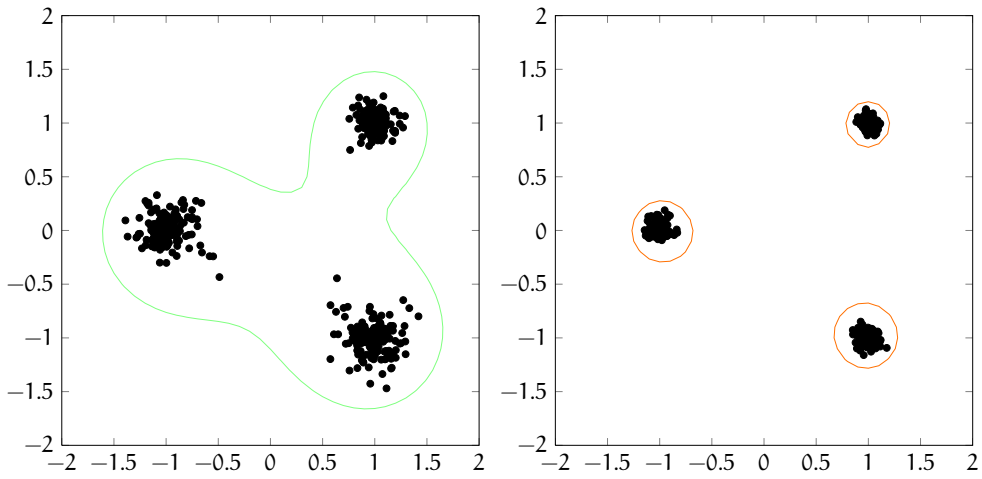


Abbildung 3: Plot des Spektrums zufällig gestörter Matrizen $\mathbf{A}+\mathbf{E}$, wobei die Norm der Störungen $\|\mathbf{E}\|$ $\epsilon = 10^{0.4}$ (links) und $\epsilon = 10^{0.8}$ (rechts) nicht überschreitet.

2.1 BERECHNUNG VON PSEUDOSPEKTREN

Zur Berechnung von Pseudospektren wird eine vierte ebenfalls äquivalente Definition verwendet, die Pseudospektren über die Menge der Singulärwerte s definiert. Dies ist von Vorteil, denn zur Berechnung der Singulärwärte einer Matrix existieren schon optimierte Softwarepakete. Wenn also die 2-Norm $\|\cdot\|_2$ verwendet wird und $\sigma_\epsilon(\mathbf{A})$ das Pseudospektrum bezeichnet, dann ist die Zahl $z \in \mathbb{C}$ Element von $\sigma_\epsilon(\mathbf{A})$ wenn: [12]

$$s_{min}(\mathbf{A} - z\mathbf{I}) < \epsilon. \tag{12}$$

Auf Basis dieser Definition lässt sich ein Algorithmus schreiben, mit dessen Hilfe die Berechnung erfolgen kann. Beispielsweise eignet sich folgender noch nicht optimierter grundlegender Quellcode, der auch zum Errechnen von Abbildung 2 im vorausgehenden Teil dieser Arbeit verwendet wurde [13]:

```
%Die Koordinaten-Eingangsvektoren x und y werden vorgegeben.
%Durchlaufe die Schleife fuer zwei Laufvariablen k und j.
for k=1:1:m, for j =1:1:m
                %Fuer die Eingangsvektoren x und y
                %berechne fuer jeden Punkt den kleinsten
                %Singulaerwert und speichere diese in sigmin.
                sigmin(j,k) = min(svd(A-(x(k)+y(j)*i)*eye(N)));
end, end
%Sende die Ergebnisse an einen Contourplotter.
contour(x,y,-log10(sigmin))
```

Listing 1: Plotten des Pseudospektrums.

12 Trefethen und Embree [27] Seite 17, Fourth definition of pseudospectra
13 Trefethen und Embree[27] Seite 371

Eine weitere Möglichkeit liegt wie eingangs schon beschrieben in der zufälligen Veränderung von Matrizen. Dieser Algorithmus fand Anwendung bei der Berechnung von Abbildung 3. Im Quellcode findet sich wegen der logarithmischen Skalierung $10^{-\epsilon}$. Eine mögliche Implementierung könnte wie folgt aussehen:

```
%Plotte die Eigenwerte von 200 gestoerten Matrizen.
for k=1:200
        %Erstelle eine mit zufaelligen komplexen Zahlen
        %gefuellte Matrix Err.
        Err = randn(dim)+i*randn(dim);
        %Normiere Err durch Multiplikation mit einem
        %Normierungsfaktor sodass Err element (0,10^-epsilon).
        Err = Err*10^(-eps)/norm(Err)*rand();
        %Berechne die Eigenwerte der Zufaellig gestoerten Matrix
        ew = eig(A+Err);
        %Plotte die Eigenwerte
        plot(real(ew),imag(ew),'k.');
end
```

Listing 2: Plotten zufälliger Störungen.

Die vorgestellten Algorithmen funktionieren gut, wenn sie auf einfache Matrizen angewandt werden. Der Algorithmus aus Listing 1 arbeitet im Wesentlichen auf einem Gitter, dass von den Vektoren **x** und **y** aufgespannt wird. Dabei erhöht sich der Rechenaufwand proportional zur Anzahl der Gitterpunkte. Bei komplizierten Problemstellungen, die eine hohe Auflösung erfordern, wird die Berechnung des Pseudospektrums mit dieser Vorgehensweise zu aufwändig. Im Folgenden werden Verfahren vorgestellt, die die Berechnung beschleunigen.

2.1.1 *Parallelisierung*

Glücklicherweise sind die Berechnungen für einzelne Gitterpunkte voneinander unabhängig. Der Quellcode aus Listing 1 lässt sich also so umformulieren, dass parallele Berechnungen auf mehreren Kernen erfolgen können [14]. Auf dem alten Intel i5-430M Prozessor des Autors laufen maximal 4 Threads gleichzeitig auf 2 Kernen [15]. Die parallel for Schleife (parfor) wird in matrix laboratory (Matlab) verwendet, um for-Schleifen zu parallelisieren. Hier erlaubt Matlab keine zweidimensionale Indexierung. Der Schleifenkörper muss daher etwas umformuliert werden [16]:

Threads sind Teile eines Programmes, die parallel zueinander und zum eigentlichen Programm laufen. Mehrere Threads können Berechnungen stark beschleunigen.

```
%Ermittle den aktuellen threadPool.
p = gcp('nocreate');
%Falls noch kein Pool existiert. Erstelle einen neuen.
```

14 Trefethen und Embree [27] Seite 378
15 Intel corporation [8]
16 Shure [24]

```
if isempty(p)
        %Erlaube nur 4 Threads im neuen Pool.
        p = parpool('local',4)
end

%Mehrere Iterationen sollen parallel durchgefuehrt werden.
parfor iX=1:1:m
        %Erstelle eine temporaere Variable.
        sigtemp = zeros(1,m2);
        for iY=1:1:m2
                %Nur eindimensionale Indexierung
                sigtemp(iY) = min(svd((A-x(iX)+y(iY)*i)*eye(N)));
        end
        %Kombiniere Daten zu zweidimensionalem Array.
        sigmin(:,iX) = sigtemp;
end
```

Listing 3: Parallele Implementierung des Gitter-SVD-Ansatzes.

Im Falle der bereits beschriebenen Matrix $\mathbf{A}$ verkürzt sich bei parallelem Rechnen die Rechenzeit von etwa `8.4s` auf etwa `4.8s`, bei einer Gitterauflösung von `0.01` und vier Threads auf besagtem i5-430M Prozessor. Noch größere Einsparungen lassen sich bei höherdimensionalen Problemen erzielen. Um das Pseudospektrum der Grcar-Test-Matrix `gallery('grcar',50,3)` mit der Gitterauflösung `0.01` zu errechenen, benötigt der nichtoptimierte Gitter-Algorithmus etwa `372s`. Durch Anwendung der parallelen Version sinkt die Rechenzeit auf etwa `170s`. So lassen sich $\approx 50\%$ Rechenzeit einsparen.

Im Anhang findet sich ein Plot des hier berechneten Pseudospektrums. Siehe Abbildung 12.

2.1.2 *Optimierung des Gitter-Ansatzes*

2.1.2.1 *Sprung-Ansatz*

Legt man sich im Vorfeld auf eine zu berechnende Niveaukurve fest, so lässt sich der Rechenaufwand durch Überspringen von Gitterpunkten verringern. Dabei nutzt man die Tatsache, dass das Pseudospektrum immer mindestens einen Kreis mit dem Radius ϵ um die Eigenwerte bildet, so dass für eine beliebige Matrix $\mathbf{A}$ gilt [17];

$$\sigma_\epsilon(\mathbf{A}) \supseteq \sigma(A) + \Delta_\epsilon \text{ mit } \Delta_\epsilon = \{z \in \mathbb{C} : \|z\| < \epsilon\}. \tag{13}$$

Bisher wurde σ_{min} immer als $-\log_{10}(\sigma_{min})$ aufgefasst. Hier wird aufgrund der später folgenden Implementierung darauf verzichtet.

Ist man man auf einem Eigenwert, so ist also die Mindestdistanz zur Niveaukurve ϵ. In diesem Spezialfall ist $\sigma_{min} = 0$. An anderen Punkten innerhalb des Pseudospektrums beträgt die Mindestdistanz bis zur Grenzkurve $\delta = \epsilon - \sigma_{min}$ [18] daher gilt:

$$\sigma_\epsilon(\mathbf{A}) \subseteq \sigma_{\epsilon-\delta} + \Delta_\delta \text{ mit } 0 < \delta < \epsilon. \tag{14}$$

17 Trefethen und Embree[27] Seite 19
18 Wie im Korrektur-Schritt bei Brühl [5] Seite 5

Auf Basis dieser Idee lässt sich eine optimierte Version des Gitter-Algorithmus entwickeln:

```
function [] = skipGrid (A,x,dim,mu)
        N = dim;
        m = length(x);
        res = (max(x)-min(x))/length(x);
        %Delogarithmiere die Eingabe:
        eps = 10^(-mu);
        for k=1:1:m
         j = 1;
         while (j <= m)
          sig = min(svd(A-(x(k)+x(j)*i)*eye(N)));
          %sind wir im Pseudospektrum?
          if sig < eps
            %Ja, zaehle den aktuellen Punkt zum Pseudospktr.
            spec(j,k) = 1;
            %Springe die Distanz (epsilon - sigma)
            %Berechne #Schritte in Sprungdistanz:
            jmp = int32((eps - sig)/res);
            %Verlasse den gewuenschten Bildbereich nicht.
            if (j+jmp)<m
              spec(j:(j+jmp),k)= 1;
              j = j + jmp;
            end;
          else
            %Nicht innerhalb des Spektrums.
            spec(j,k) = 0;
          end
          j = j + 1;
         end
        end
        contour(x,x,spec);
```

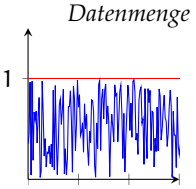
Listing 4: Sprung-Variante des Gitter-SVD-Ansatzes.

Diese Variante benötigt zum Berechnen der Niveaukurve bei $\epsilon = 0.4$ der bekannten Matrix $\mathbf{A}$ `0.22s`, während der klassische Gitter-SVD-Algorithmus in nur `0.17s` durchläuft. Die neue Variante braucht also bei diesem niedrig-dimensionalen Problem etwa 1.3 mal so lang. Die längere Rechendauer ist dem größeren Verwaltungsaufwand geschuldet, der zur Behandlung der Sprünge notwendig wird. Im Code tauchen if-Abfragen auf, die im klassischen Ansatz nicht vorkommen, deren Auswertung benötigt Zeit. Bei hochdimensionalen Problemen zahlt sich das Springen jedoch aus. Um die Grenzkurve zu $\epsilon = 0.3468$ der Grcar-Matrix `gallery('grcar',50,3)` zu berechnen, braucht diese Variante lediglich etwa `251s`. Der nichtoptimierte Gitter-Algorithmus benötigt etwa `368s`. In diesem Fall verkürzt sich die Rechenzeit etwa um den Faktor 0.68.
Damit der Algorithmus parallelisierbar bleibt, wurde auf eine Implementierung verzichtet, in der auch die Daten-Reihen über und unter

der betroffenen Reihe mitbearbeitet werden, obwohl dies mathematisch möglich ist. Auch könnte eine andere Datenstruktur in Frage kommen, zum Beispiel mit Pointern, die die betreffenden Grenzen des Pseudospektrums mit schneller Zugriffszeit speichern. Die beiden Ansätze bieten die Möglichkeit, weitere Zeiteinsparungen zu erzielen.

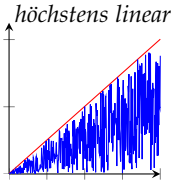
2.1.2.2 *Schur-Zerlegung mit anschließenden Lanczos-Iterationen*

Das Landau-Symbol $\mathcal{O}$ bezeichnet die Rechendauer als Funktion der Eingangsdaten. $\mathcal{O}(1)$ steht für beschränkte Rechenzeit egal wie groß die Datenmenge



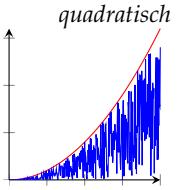
,

$\mathcal{O}(N)$ wächst höchstens linear



,

$\mathcal{O}(N^2)$ höchstens quadratisch



,

...

Zur Reduktion des Rechenaufwandes lässt sich auch eine Schur-Zerlegung auf die Ausgangsmatrix $\mathbf{A}$ anwenden. Da die Schur-Zerlegung eine Ähnlichkeitsabbildung ist [19], bleiben Eigenwerte und Pseudospektrum davon unberührt. Die Schur-Matrix $\mathrm{Schur}(\mathbf{A}) = \mathbf{U}^T\mathbf{A}\mathbf{U}$ liegt nach der Zerlegung für alle Gitter-Punkte z schon in oberer Dreiecksform vor, denn:

$$\mathbf{U}^T(\mathbf{A} - z\mathbf{I})\mathbf{U} = \mathbf{U}^T\mathbf{B}\mathbf{U} - z\mathbf{U}^T\mathbf{I}\mathbf{U} = \mathbf{T} - z\mathbf{I}.$$

$\mathbf{U}$ bezeichnet eine orthogonale Transformationsmatrix und $\mathbf{T}$ ist eine obere rechte Dreiecksmatrix. Die Verschiebung $(\mathbf{A} - z\mathbf{I})$ ändert also lediglich die Diagonale, nicht aber die durch die Schur-Zerlegung hergestellte obere Dreiecksform der Matrix. Damit werden Zerlegungen, die zur Berechnung der Singulärwerte typischerweise in jedem Punkt vorgenommen werden [20], hinfällig. Häufig wird für die anschließende Berechnung der Singulärwerte die Matrix mithilfe des Lanczos-Algorithmus in tridiagonale Form gebracht. Dabei nutzt man die Verbindung zwischen Tridiagonalisierung und QR-Zerlegung [21], durch die sich die Einträge der Tridiagonalmatrix direkt berechnen lassen. Die Schur-Zerlegung kostet einmalig $\mathcal{O}(N^3)$. Geht man von m^2 Gitterpunkten aus, verbessert sich die Rechenzeit damit von $\mathcal{O}(m^2N^3)$ auf $\mathcal{O}(N^3 + m^2N^2)$, denn nach der Zerlegung fällt pro zu berechnendem Singulärwert nur noch $\mathcal{O}(N^2)$ an [22] und der Lanczos-Algorithmus konvergiert in der Regel zügig. Wird die Implementierung des inversen Lanczos-Algorithmus aus Listing 10 verwendet, sinkt die Rechenzeit zur Berechnung von `gallery('grcar',50,3)` mit denselben Parametern wie im vorigen Beispiel im Vergleich zu der Rechenzeit des nichtoptimierten Gitter-Algorithmus aus Listing 1 um etwa den Faktor 0.85.

Die Schur-Zerlegung bietet jedoch zusätzlich auch die Möglichkeit, Rechenzeit durch Kompression der Ausgangsdaten einzusparen [23].

19 Trefethen und Embree [27] Seite 373
20 QR-Zerlegung, Lui [20] Seite 8 oder LU-Zerlegung, Trefethen und Embree [27] Seite 373
21 Golub [11] Seite 472
22 Trefethen und Embree [27] Seite 374
23 Trefethen und Embree [27] Seite 382

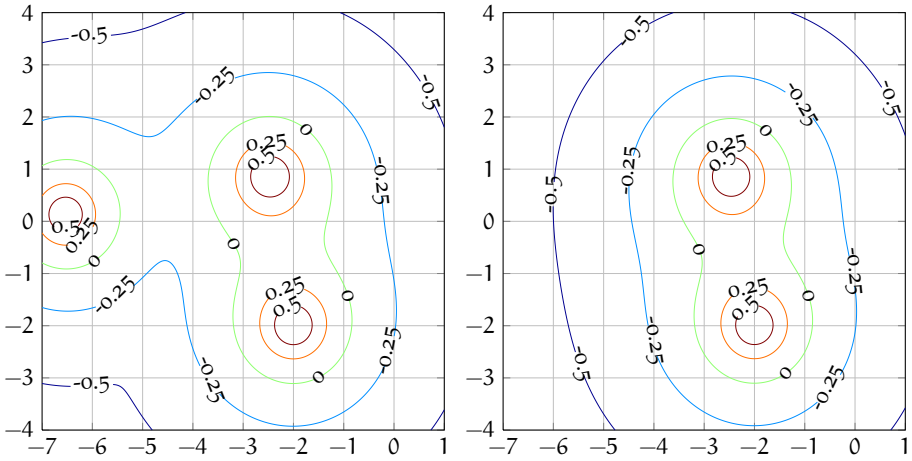


Abbildung 4: Pseudospektrum der Matrix **B** links, Pseudospektrum der komprimierten Schur-Matrix rechts.

Im Folgenden soll diese Möglichkeit anhand der Matrix **B** beleuchtet werden:

$$\mathbf{B} = \begin{pmatrix} -6 & 2 & 0 \\ 1 & -3+\mathrm{i} & 0 \\ 0 & 2 & -2-2*\mathrm{i} \end{pmatrix}.$$

Die Realteile der Eigenwerte von **B** sind etwa -6.5, -2.5 und -2. Dies zeigt sich auch in der Schur-Zerlegung:

$$\mathrm{Schur}(\mathbf{B}) = \begin{pmatrix} -6.5337+0.1312\mathrm{i} & 0.2787+1.0717\mathrm{i} & -0.9095+0.4313\mathrm{i} \\ 0 & -2.4663+0.8688\mathrm{i} & -1.3938+1.1286\mathrm{i} \\ 0 & 0 & -2.0000-2.0000\mathrm{i} \end{pmatrix}.$$

Häufig ist lediglich der Verlauf der Niveaulinien der Resolvente $\|A - z\mathbf{I}\|$ in der Umgebung der imaginären Achse interessant. Da der Eigenwert $-6.5+0.1\mathrm{i}$ deutlich weiter von der imaginären Achse entfernt ist als die anderen Eigenwerte, hat er wahrscheinlich in diesem Bereich wenig Auswirkungen auf das Pseudospektrum. Er kann vernachlässigt werden. Im Bezug auf das Bild in der Nähe der imaginären Achse kann also äquivalent auch nur die oben in rot dargestellte komprimierte Matrix betrachtet werden. Zum Vergleich sind beide Plots in Abbildung 4 dargestellt. Die Rechenzeit verkürzt sich durch Kompression bei Verwendung des einfachen Gitter-Algorithmus von `14.7s` auf `11s`, es können also etwa 25% eingespart werden.

2.1.3 *Kurvenfortsetzung*

Eine weitere Möglichkeit zur Reduktion des Rechenaufwandes besteht darin, sich komplett von Gitter-basierten Methoden zu verab-

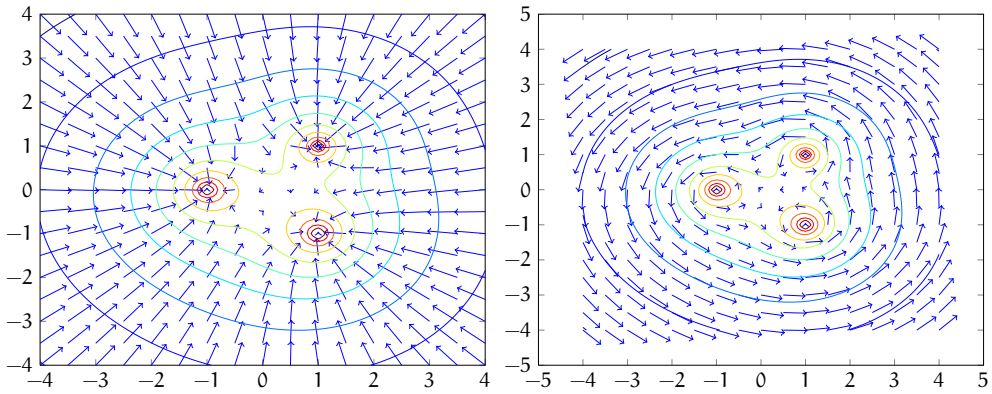


Abbildung 5: Plot des Gradientenfeldes (links) und Plot der um $-\pi/2$ gedrehten Gradienten (rechts).

schieden [24]. Beispielsweise lässt sich die Grenzkurve $\partial\sigma_\epsilon$ eines speziellen Pseudospektrums direkt verfolgen [25]. Dabei nutzt man dieselbe Definition von Pseudospektren, die auch schon für den Gitter-Ansatz entscheidend war. Es gilt 12, also $z \in \sigma_\epsilon$ genau dann, wenn:

$$s_{min}(\mathbf{A} - z\mathbf{I}) < \epsilon.$$

Damit lässt sich eine Funktion für $z \in \mathbb{C}$ definieren:

$$g(z) := s_{min}(\mathbf{A} - z\mathbf{I}) \tag{15}$$

Auf der Grenzkurve $\partial\sigma_\epsilon$ ist die Funktion $g(z)$ konstant gleich ϵ. In diesem Sonderfall liegen alle Punkte z auf der Kurve [26]:

$$\partial\sigma_\epsilon = \{z \in \mathbb{C} | g(z) = \epsilon\} \tag{16}$$

Diese Kurve lässt sich mit Hilfe von Pfadverfolgungstechniken nachzeichnen. Die Hauptrolle bei der Verfolgung spielt der Gradient $\nabla g(z) \in \mathbb{C}$. Dieser lässt sich aus dem Produkt des rechten und linken Singulärvektors bilden, es gilt [27]:

$$\nabla g(z) = \bar{v}^T u \ \in \mathbb{C}. \tag{17}$$

Dabei bezeichnet v den rechten und u den linken Singulärvektor von $(\mathbf{A} - z\mathbf{I})$. Mit Hilfe dieser Gleichung lässt sich das Vektorfeld plotten. Dies ist in Abbildung 5 für die bekannte Matrix $\mathbf{A}$ auf der linken Seite geschehen. Zur Kurvenverfolgung werden jedoch auch die gedrehten Vektoren benötigt:

$$\nabla g(z)_{gedreht} = \bar{v}^T u \cdot e^{-\frac{i\pi}{2}} = (-i)\bar{v}^T u. \tag{18}$$

Die rechte Seite von Abbildung 5 stellt diese Situation dar.

24 Trefethen und Embree [27] Seite 394
25 Brühl [5] Seite 3
26 Brühl [5] Seite 3
27 Brühl [5] Seite 4 und Trefethen und Embree [27] Seite 395

Unter Zuhilfenahme des Gradienten lässt sich nun ein Algorithmus zur Pfadverfolgung entwickeln. Zu Anfang bestimmt man den gedrehten Gradienten im Ausgangspunkt und läuft einen Schritt. Dieser hat gerade die Schrittlänge τ. Da der Rand des Pseudospektrums aller Wahrscheinlichkeit nicht gerade verläuft, entsteht bei diesem Schritt ein Fehler. Die Größe des Fehlers ist etwa gleich der Differenz zwischen dem aktuellen Wert des kleinsten Singulärwerts s_{min} und dem Niveau der gesuchten Kurve ϵ. Daher muss der nächste Schritt ein Korrekturschritt in Richtung des Gradienten sein. Die Länge des Korrekturschritts ist die Größe des Fehlers also $(s_{min} - \epsilon)$. In Matlab könnte eine Implementierung folgende Form annehmen:

```
% Schritt 0: Ein erster Punkt.
z(1) = 1.2-(0.05)*i;
%Berechne Epsilon auf dem aktuellen Niveau und Singulaervektoren.
[u,epsilon,v] = svds((A-(z(1)*eye(N))),1,0);
%Laut Bruehl ergibt sich die Richtung des Gradienten aus:
grad(1) = (v'*u);
for k = 2:1:maxPunkt
        %Richtung ist gefunden. Waehle eine Schrittlaenge:
        tau = 0.01;
        %Schritt 1:     Treffe eine neue Vorhersage:
        zHut(k) = z(k-1) + tau*(grad(k-1)*exp(-i*pi/2));
        %Die neue Richtung:
        [u,sigmin,v] = svds((A-(zHut(k-1)*eye(N))),1,0);
        %Laut Bruehl ergibt sich die Richtung des Gradienten aus:
        grad(k) = (v'*u)/norm(v'*u);
        %Schritt 2:     Korriegiere die Vorhersage
        z(k) = zHut(k)+ (sigmin - epsilon)*grad(k);
end
%Fertig Plotten!
for k=1:1:maxPunkt
        plot(z(k),'b*');
        hold on;
end
```

Listing 5: Eine mögliche Form des Trace-Algorithmus.

Lässt man den oben gegebenen Code laufen, so ergibt sich der in Abblidung 6 dargestellte Plot. Zum Berechnen der Niveaukurve mit $\epsilon \approx 0.45$ ausgehend vom Startpunkt `z1=0.53 - 3.517i` der Matrix `gallery('grcar',50,3)` mit `2100` Punkten benötigt der Algorithmus etwa `94s`. Zum Vergleich brauchte der nichtoptimierte Gitter-Algorithmus etwa `368s` für das gesamte Pseudospektrum. Nur eine Niveaukurve ließ sich schneller mit dem Sprung-Ansatz in etwa `250s` ausrechnen. Im Vergleich zu dem Sprung-Ansatz reduziert sich die Rechenzeit hier um den Faktor 0.376. Leider oszilliert dieser Algorithmus leicht wie aus der angehängten Abbildung 13 hervorgeht.

ϵ ist nicht logarithmiert.

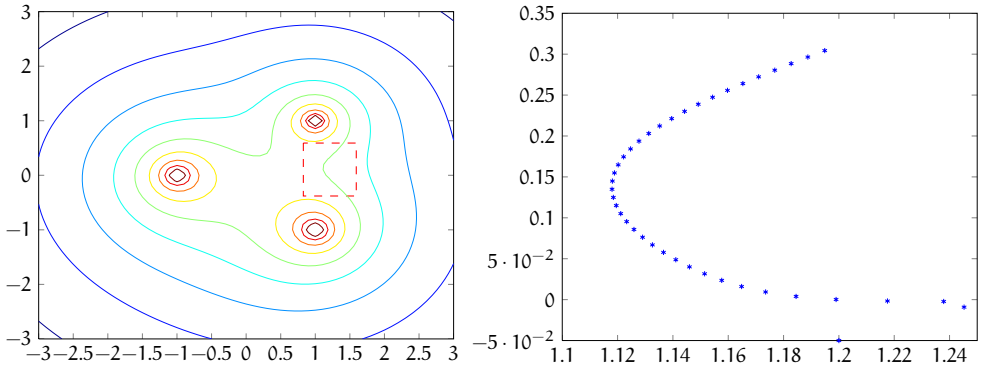


Abbildung 6: Plot der Ausgabe des Trace-Algorithmus (rechts).

2.1.4 *Hamilton'sche Eigenwert-Methode*

Dieser Abschnitt basiert auf Techniken aus Abschnitt 3.3 tiefer-gehende Erläuterungen finden sich dort.

Eine letzte Methode zum schnellen Berechnen einer ausgewählten Niveaukurve basiert auf der Tatsache, dass sich die Schnittpunkte mit gedachten horizontalen und vertikalen Linien mithilfe von Eigenwertproblemen bestimmen lassen. Man verwendet die Matrizen [28]:

$$\mathbf{H}_1 = \begin{pmatrix} x\mathbf{I} - \overline{\mathbf{A}}^T & \epsilon\mathbf{I} \\ -\epsilon\mathbf{I} & \mathbf{A} - x\mathbf{I} \end{pmatrix}, \quad \mathbf{H}_2 = \begin{pmatrix} i\overline{\mathbf{A}}^T - y\mathbf{I} & \epsilon\mathbf{I} \\ -\epsilon\mathbf{I} & i\mathbf{A} + y\mathbf{I} \end{pmatrix}. \tag{19}$$

Der Schlüssel zum Algorithmus ist, dass die imaginären Eigenwerte der Matrix $\mathbf{H}_1$ alle Schnittpunkte von $\partial\sigma_\epsilon$ mit einer gedachten durch x gehenden vertikalen Linie enthalten. Die Funktion `vertikaleSuche` aus Listing 8 nutzt diese Tatsache und liefert uns alle Schnittpunkte in vertikaler Richtung. Jetzt bewegt sich der Algorithmus in horizontaler Richtung weiter und plottet immer alle gefundenen Schnittpunkte. So ergibt sich peu à peu die Niveaukurve zum eingangs festgelegten ϵ. Theoretisch kann man an dieser Stelle aufhören; und auch in der Praxis ergeben sich so schon brauchbare Plots, jedoch wird das Ergebnis voller, wenn auch noch horizontal gesucht wird, während sich der Algorithmus parallel zur imaginären Achse bewegt. Die Funktion `horizontaleSuche` wertet die imaginären Eigenwerte der Matrix $\mathbf{H}_2$ aus. Sie findet sich in Listing 9. Es folgt eine Implementierung, die eine Niveaukurve zu einem Parameter `eps` (für ϵ) zur Eingangsmatrix $\mathbf{A}$ mit Schrittweite `res` produziert.

```
function [] = hamiltonianPlotLevel(A, eps, res)
%Finde den groessten und kleinsten Eigenwert.
eigs = eig(A);
maxEig = max(eigs);
minEig = min(eigs);
%Lege zwei Strecken x und y fest.
x = (real(minEig) - 10*eps):res:(real(maxEig) + 10*eps);
```

28 Siehe auch Gleichung 37

```
y = (imag(minEig) - 10*eps):res:(imag(maxEig) + 10*eps);
%Loese die Eigenwertprobleme entlang der Strecke x.
for i = 1:1:length(x)
    treffer = vertikaleSuche(A,eps,x(i),res);
    for j = 1:1:length(treffer)
        plot(x(i),treffer(j));
        hold on;
    end
end
%Loese die Eigenwertprobleme entlang der Strecke y.
for i = 1:1:length(y)
    treffer = horizontaleSuche(A,eps,y(i),res);
    for j = 1:1:length(treffer)
        plot(treffer(j),y(i));
        hold on;
    end
end
end
```

Listing 6: Eigenwert-Methode zum Berechnen einer Niveaukurve.

Dieser Algorithmus findet die Niveaukurve $\epsilon = 0.45$ der `gallery ('grcar',50,3)`-Test Matrix in etwa `33s`. Und ist damit fast 3-mal so schnell wie der Kurven-Nachverfolgungs-Algorithmus der `94s` benötigte, ein zusätzlicher Vorteil ist, dass keine Oszillationsprobleme auftreten.

2.2 PSEUDOSPEKTREN UND DAS EINSCHWUNGVERHALTEN

Normale Matrizen folgen in ihrem Verhalten im Wesentlichen dem, was man anhand des Spektrums erwarten würde. Im Fall von nichtnormalen Systemen führen Schlüsse, die allein aus dem Spektrum gezogen werden, in die Irre. Zuallererst ist es jedoch wichtig, den Grad des nichtnormalen Verhaltens eines Systems $\mathbf{A}$ quantifizieren zu können. Dies passiert über die Konditionszahl [29]:

$$\kappa = \|\mathbf{S}\|\|\mathbf{S}^{-1}\| = s_{max}(\mathbf{S})/s_{min}(\mathbf{S}). \tag{20}$$

In dieser Gleichung bezeichnet s_{max} den größten und s_{min} den kleinsten Singulärwert. Die Matrix $\mathbf{S}$ steht, wie schon in Kapitel 1, für die Eigenvektormatrix von $\mathbf{A}$, deren Konditionszahlen bestimmt werden sollen. $\mathbf{S}$ ist nur dann eindeutig bestimmt, wenn die Eigenvektoren normiert werden. Die eindeutige Lösung ist jedoch nicht automatisch die mit dem kleinstmöglichen κ, jedoch ist sie maximal um den Faktor $\sqrt{N}$ größer als sie [30]. Es gilt immer $\kappa \geqslant 1$. Wenn das Bezugssystem normal ist gilt, $\kappa = 1$, denn im normalen Fall lässt sich per Definition ein orthogonales System finden. Da dieses anschließend normiert wird, gilt: $1 = \|S^T\| = \|S^{-1}\| = \|S\|$. Die Konditionszahl ermöglicht es lediglich vorherzusagen, ob nicht normales

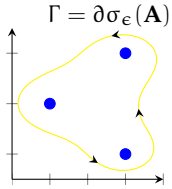
N bezeichnet die Dimension des Problems.

29 Trefethen und Embree [27] Seite 19
30 Trefethen und Embree [27] Seite 19

Schwungverhalten auftreten wird und ob dieses eher intensiv oder eher schwach ausgeprägt sein wird. Genauere Abschätzungen sind nicht möglich. Daher wurden Ungleichungen entwickelt, die es ermöglichen genauere Abschätzungen nach oben und unten zu treffen. Es folgt eine kommentierte Herleitung zweier nützlicher Ungleichungen aus bekannten Gleichungen, die das Zusammenspiel von $e^{t\mathbf{A}}$ und $(\mathbf{A} - z\mathbf{I})^{-1}$ näher beleuchten [31]. Eine bekannte Beziehung ist [32]:

$$e^{t\mathbf{A}} = \frac{1}{2\pi i} \oint_{\Gamma} e^{zt}(z - \mathbf{A})^{-1} \, dz. \tag{21}$$

Ein möglicher Pfad $\Gamma = \partial\sigma_\epsilon(\mathbf{A})$



mit $\mathbf{A} = \mathbf{D}$ *und* $\epsilon = 0.8$.

Hinter dem Ausdruck verbirgt sich die Möglichkeit $e^{t\mathbf{A}}$ auszurechnen, indem man einen geschlossenen Pfad Γ in der komplexen Ebene entlang läuft, unterwegs über $e^{zt}(z - \mathbf{A})^{-1}$ integriert und abschließend mit $\frac{1}{2\pi i}$ multipliziert. Der Pfad Γ muss alle Eigenwerte in mathematisch positiver Laufrichtung einschließen, also das Spektrum umfassen. Zusätzlich muss er geschlossen sein. Um von Gleichung 21 auf eine Obergrenze für $\|e^{t\mathbf{A}}\|$ zu kommen, beginnt man damit, auf beiden Seiten Normstriche zu setzen. Berücksichtigt man, dass $|\frac{1}{2\pi i}| = |\frac{1}{2\pi}| = \frac{1}{2\pi}$, dann kommt heraus:

$$\|e^{t\mathbf{A}}\| = \frac{1}{2\pi} \| \oint_{\Gamma} e^{zt} \cdot (z - \mathbf{A})^{-1} \, dz \|.$$

$|e^{zt}| = e^{Re(zt)} = e^{tRe(z)}$

Nachdem man die Normstriche in das Integral gezogen hat, wird in einem nächsten Schritt z so gewählt, dass $Re(z)$ so groß wie möglich wird, aber nicht außerhalb des Pseudospektrums liegt. Der Realteil eines solchen z heißt auch Spektralabzisse und wird dann als α_ϵ geschrieben. Mit $Re(z) = \alpha_\epsilon$ wird auch die Exponentialfunktion $\|e^{zt}\|$ maximal. Wird anschließend noch der Pfad Γ gleich einer Grenze des Pseudospektrums $\partial\sigma_\epsilon(\mathbf{D})$ gewählt, dann bleibt der Ausdruck $\|(z - \mathbf{A})^{-1}\|$ konstant gleich $\frac{1}{\epsilon}$. Es ergibt sich mit L_ϵ als Pfadlänge [33]:

$$\|e^{t\mathbf{A}}\| \leqslant \frac{1}{2\pi} \frac{e^{t\alpha_\epsilon(A)}}{\epsilon} \cdot L_\epsilon. \tag{22}$$

Ungleichung 22 stellt eine Obergrenze für das Einschwungverhalten eines beliebigen Systems dar.

Logischerweise besteht auch Interesse daran, Untergrenzen abzuschätzen. Die Herleitung einer Ungleichung zu diesem Zweck geht von folgendem Zusammenhang aus [34]:

$$(z - \mathbf{A})^{-1} = \int_0^\infty e^{-zt} e^{t\mathbf{A}} dt \text{ mit } Re(z) > 0, \sigma(\mathbf{A}) \text{ in linker Halbebene.}$$

31 Nach Lindner [unveröffentlichte Gesprächsnotiz] sowie Trefethen und Embree [27] 149-151

32 Trefethen und Embree [27] 149

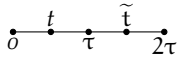
33 Trefethen und Embree [27] 150

34 Trefethen und Embree [27] Seite 149

(23)

Zuallererst wird ein Hilfsausdruck $M_\tau = \sup_{0<t\leqslant\tau} \|e^{t\mathbf{A}}\|$ definiert. M_τ gilt für $0 < t \leqslant \tau$, jedoch würde man den Zeitbereich jetzt gerne verschieben. Nach einem Blick auf den Zeitstrahl ergibt sich:

Zeitstrahl zu M_τ:



$$\|e^{\tilde{t}\mathbf{A}}\| = \|e^{t\mathbf{A}}e^{\tau\mathbf{A}}\| \leqslant \|e^{t\mathbf{A}}\|\|e^{\tau\mathbf{A}}\| \leqslant M_\tau^2,$$

weil $\|e^{t\mathbf{A}}\| \leqslant M_\tau$ und $\|e^{\tau\mathbf{A}}\| = M_\tau$. Wird diese Beobachtung verallgemeinert, dann kommt heraus [35]:

$$\sup_{j\tau\leqslant t\leqslant(j+1)\tau} \|e^{t\mathbf{A}}\| \leqslant M_\tau^{j+1},\ j = 0, 1, 2, \ldots \tag{24}$$

Jetzt ist es an der Zeit, sich an Gleichung 23 zurückzuerinnern. Das Integral zerlegt man in unendlich viele Teile der Länge τ:

$$(z-\mathbf{A})^{-1} = \int_0^\infty e^{-zt}e^{t\mathbf{A}}dt = \sum_{j=0}^{\infty}\int_{j\tau}^{(j+1)\tau} e^{-zt}e^{t\mathbf{A}}dt.$$

Danach wird Ungleichung 24 und $\mathrm{Re}(z) = \alpha$ eingesetzt:

$$\|(z-\mathbf{A})^{-1}\| \leqslant \sum_{j=0}^{\infty}\int_{j\tau}^{(j+1)\tau} e^{-\alpha t}M_\tau^{j+1}dt.$$

Im nächsten Schritt wird M_τ^{j+1} aus dem Integral herausgezogen und dieses dann ausgerechnet. Anschließend wird $e^{-\alpha j t}$ ausgeklammert:

$$\sum_{j=0}^{\infty} M_\tau^{j+1}\int_{j\tau}^{(j+1)\tau} e^{-\alpha t}dt = \sum_{j=0}^{\infty} M_\tau^{j+1}e^{-\alpha j\tau}\frac{1-e^{-\alpha\tau}}{\alpha}.$$

Jetzt lässt sich M_τ ausklammern und die geometrische Reihe anwenden. Dies ist nur möglich, wenn der Betrag des Ausdrucks in der Summe kleiner eins ist [36], also muss $|M_\tau e^{-\alpha\tau}| < 1$ oder $M_\tau < e^{\alpha\tau}$ angenommen werden. Das ist unproblematisch, denn im anderen Fall $M_\tau \geqslant e^{\alpha\tau}$, gilt Ungleichung 25 auf jeden Fall[37]:

Die Summe der unendlichen geometrischen Reihe: $\sum_{n=0}^{\infty} q^n = \frac{1}{1-q}$ *wenn* $|q| < 1$

$$\frac{1-e^{-\alpha\tau}}{\alpha}M_\tau\sum_{j=0}^{\infty}(M_\tau e^{-\alpha\tau})^j = \frac{1-e^{-\alpha\tau}}{\alpha}\frac{M_\tau}{1-M_\tau e^{-\alpha\tau}}.$$

Dann werden beide Seiten mit α multipliziert und anschließend rechts umgestellt:

$$\alpha\|(z-\mathbf{A})^{-1}\| \leqslant (1-e^{-\alpha\tau})\frac{M_\tau}{1-M_\tau e^{-\alpha\tau}} = \frac{e^{\alpha\tau}-1}{e^{\alpha\tau}/M_\tau - 1}.$$

35 Nach Lindner [unveröffentlichte Gesprächsnotiz]
36 Kleine Enzyklopädie Mathematik [23] Seite 426
37 Trefethen und Embree [27] Seite 152

Abschließend wird $a\|(z-\mathbf{A})^{-1}\|$ zu K zusammengefasst, ein letztes Mal umgestellt und die Hilfsdefinition $M_\tau = \sup_{0<t\leqslant\tau} \|e^{t\mathbf{A}}\|$ eingesetzt [38]:

$$\sup_{0<t\leqslant\tau} \|e^{t\mathbf{A}}\| \geqslant e^{a\tau}/(1+\frac{e^{a\tau}-1}{K}). \tag{25}$$

Eine Untergrenze für Schwingungsvorgänge ist hergeleitet. Im Folgenden wird sie im Gegensatz zu Untergrenzen aus der Eigenwerttheorie, „Pseudospektralgrenze“genannt.

Nun soll anhand der zwei Beispielmatrizen **C** und **D** der Effekt der Nichtnormalität veranschaulicht werden.

$$\mathbf{C} = \begin{pmatrix} -3 & 2 & 3 \\ 0 & -2+2i & 1 \\ 0 & 0 & -2-2i \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} -3 & 8 & 0 \\ 0 & -1+2i & 7 \\ 0 & 0 & -1-2i \end{pmatrix}.$$

Die Realteile aller Eigenwerte beider Matrizen sind negativ. Die Spektral-Abszissen beider Matrizen sind damit auch negativ, denn die Spektral-Abszisse ist das Supremum der Realteile der auftretenden Eigenwerte [39] sie wird als $\alpha(\mathbf{A})$ notiert, damit erhält man die Eigenwertgrenze [40]:

Supremum und Maximum unterscheiden sich darin, dass das Supremum auch Werte erlaubt, die nur asymptotisch erreicht werden.

$$\|e^{t\mathbf{A}}\| \geqslant e^{t\alpha(\mathbf{A})} \text{ mit } \alpha(A) = \sup_{z\in\mathbb{C}}(\mathrm{Re}(z) \ : \ z \in \sigma(A)). \tag{26}$$

Für $\|e^{\mathbf{C}t}\|$ und $\|e^{\mathbf{D}t}\|$ würde man also schnelles Abklingen erwarten. Jedoch widerspricht Abbildung 7 diesem Befund. Die auf Basis der Eigenwerte gefundenen Untergrenzen (rot) geben das tatsächliche Verhalten nur unvollständig wieder. Einen ersten Eindruck bringt die Konditionszahl. Für **C** ist κ =`3.2576`, bei **D** ergibt sich κ =`38.8080`. Ein Blick auf die Pseudospektren in Plot 7 zeigt, dass die Niveaukurve zu `0.6` (grün) in die rechte Halbebene im Plot von Matrix **D** herüberreicht. Für Matrix **C** ist das nicht der Fall. Anhand dieser Beobachtung lassen sich Pseudospektral-Abszissen berechnen, die es ermöglichen, das Verhalten beider Matrizen noch besser einzuschätzen [41]. Die Pseudospektral-Abszisse ist das Supremum des Realteils der Niveaukurve $\partial\sigma_\epsilon$ [42]:

$$\alpha_\epsilon = \sup_{z\in\mathbb{C}}(\mathrm{Re}(z) \ : \ z \in \sigma_\epsilon(A)). \tag{27}$$

Die folgende Ungleichung ermöglicht es, eine untere Grenze des Schwingungsverhaltens in Abhängigkeit von der Zeit τ, der Eingangsmatrix **A** und der Größe des Pseudospektrums ϵ zu plotten [43]:

Ungleichung 28 ergibt sich aus 25, mit größtmöglichem $a = \alpha_\epsilon$ und $K = \alpha_\epsilon/\epsilon$

38 Trefethen und Embree [27] Seite 151
39 Trefethen und Embree [27] Seite 136
40 Trefethen und Embree [27] Seite 150
41 Burke et al., [6]
42 Trefethen und Embree [27] Seite 136
43 Trefethen und Embree [27] Seite 151

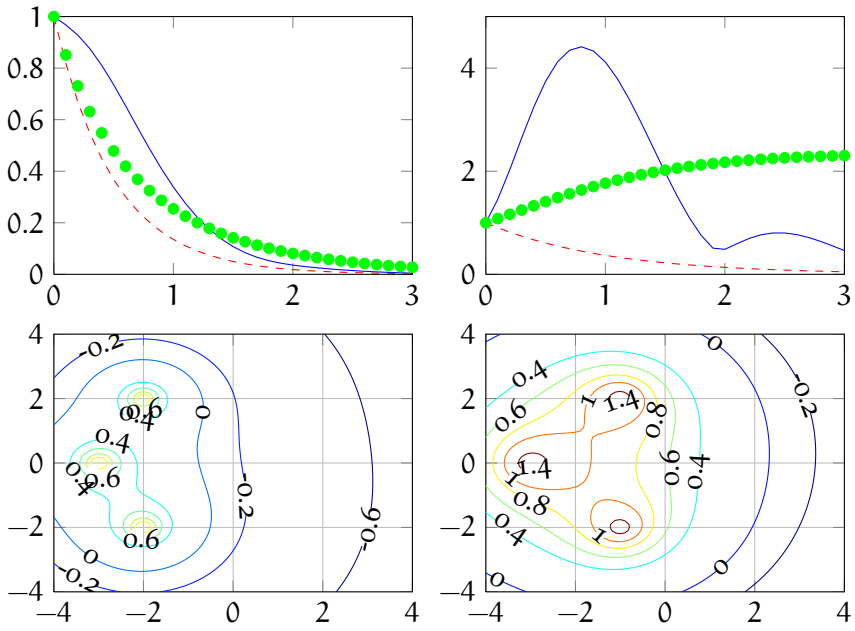


Abbildung 7: Plot des Schwungverhaltens $\|e^{tA}\|$ der Matrizen $\mathbf{A} = \mathbf{C}$ (links) und $\mathbf{A} = \mathbf{D}$ (rechts) mit zugehörigem Pseudospektrum. Eigenwertgrenzen sind in rot, auf Basis von $\epsilon = 0.6$ berechnete Pseudospektralgrenzen sind grün dargestellt.

$$\sup_{0<t\leqslant\tau} \|e^{tA}\| \geqslant \frac{e^{\alpha_\epsilon \tau}}{1 + \frac{e^{\alpha_\epsilon \tau}-1}{\alpha_\epsilon/\epsilon}}. \tag{28}$$

Ungleichung 28 bedeutet, dass die Transiente $\|e^{tA}\|$ (blau in Abbildung 7) zu einem Zeitpunkt vor τ größer sein muss als der Ausdruck auf der rechten Seite (einer der grünen Punkte in Abbildung 7). Mithilfe dieser Grundlage ist eine Funktion zum Errechnen der einzelnen Grenzwerte in Matlab konzipierbar:

```
function [grenze] = pseudoLowBound (tau,A,eps)
%pseudoLowBound (t,A,eps)
%       liefert Grenzwerte nach Gleichung 17
%       t : Der Zeitvektor
%       A : Die Matrix zu der Grenzwerte gefunden werden sollen.
%       eps : Legt die Groesse des Pseudospektrums fest.

%       Nutze Burke et al.s Funktion zur bestimmung der Pseudo-
    Abszisse:
        [alpha,z] = pspa_2way(A,eps,0,0,0)

%       Berechne die Obergrenzen:
        for i=1:1:length(tau)
                grenze(i) = exp(alpha*tau(i))/(1 + (exp(alpha*tau
                    (i)) - 1) / (alpha/eps));
        end
```

Listing 7: Funktion zur Berechnung der Grenzwerte einer Transiente einer Matrix.

Die vorgestellte Methodik ermöglicht es, das Verhalten eines Systems abzuschätzen, ohne das Pseudospektrum dezidiert plotten zu müssen.

2.3 HISTORISCHE EINORDNUNG DES THEMAS

Tabelle 1 kann nur als grober Indikator dienen, da sich die Inhalte der Datenbanken in Teilen überlappen.

Im folgenden Abschnitt soll eine historische Einordnung der Thematik versucht werden. Tabelle 1 vermittelt einen ersten Eindruck zum zeitlichen Verlauf der Forschungsintensität auf diesem Themengebiet. Leider fallen unter dem Label „Pseudospektren“ eine Reihe verschiedener mathematischer Konzepte zusammen. Der Terminus Pseudospektrum wird in westlicher Literatur auch in einem statistischen Kontext verwendet, um z.B. stochastische Prozesse besser zu verstehen [44]. Statistische „Pseudospektren“ haben jedoch mit denen aus der Algebra nichts gemeinsam. Zusätzlich existiert sowjetische und westliche Literatur in der „Pseudospektren“ im Kontext von Differentialgleichungen [45] verwendet werden. Die Konzepte sind ebenfalls nicht verwandt.

Zeitraum	MathSciNet	WebOfScience	IEEE Explore
1950-1960	0	0	0
1960-1970	2	0	0
1970-1980	3	6	0
1980-1990	71	70	2
1990-2000	234	601	44
2000-2010	275	1255	178

Tabelle 1: Publikationen zu Pseudospectr* in verschiedenen wissenschaftlichen Datenbanken.

Das Interesse an der Thematik im Sinne dieser Bachelorarbeit hat wahrscheinlich in der Mathematik begonnen. Die mathematische Datenbank MathSciNet der American Mathematical Society, die in Teilen auch europäische und sowjetische Publikationen erfasst, enthält die ersten Einträge zum Thema. Es folgen die Naturwissenschaftler (WebOfScience) in den siebziger Jahren und schließlich die Ingenieure (IEEE Explore) in den achtziger Jahren. Seit ihrer Konzeption sind Pseudospektren in den mannigfaltigsten Gebieten zum Einsatz gekommen, die größten Anwendungsgebiete liegen wohl in der Mathematik, Physik und den Ingenierswissenschaften [46].

Inhaltlich kam es zur Entwicklung von Pseudospektren von der mathematischen Seite durch James Varah in Stanford und an der

44 Hatanaka [12]
45 Mihajlović [21] und Pasciak[22]
46 Web of Science Core Collection, Research Areas by record count

University of British Columbia [47]. Varah definierte Pseudospektren unter dem Titel r-approximierte Eigenwerte und Vektoren in seiner Master-Arbeit 1967 in Stanford [48] und griff die Idee in einer Arbeit zur Sylvester-Gleichung 1979 wieder auf.

Unabhängig von Varah arbeitete Henry Landau in den Bell Laboratories an Lasern und Resonatoren. Im Zuge seiner Arbeiten in physikalischem Kontext definierte er ϵ-approximierte Eigenwerte und Vektoren um 1975 [49].

Ausgehend von Arbeiten Varahs forschte James Weldon Demmel in New York an Fragestellungen um Matrix-Stabilität. Dabei konnte er einen frühen Plot eines Pseudospektrums produzieren [50].

Schließlich begann Trefethen, zum Thema zu publizieren und die Thematik einer größeren Leserschaft zugänglich zu machen. Vermutlich geht die Namensgebung Pseudospektrum im Kontext linearer Algebra auf ihn zurück.

47 Varah, [28] Seite 5
48 Trefethen und Embree [27] Seite 41
49 Landau, [19] Seite 1 sowie Trefethen und Embree [27] Seite 44
50 Demmel, [10] Seite 3

Teil II

ANWENDUNG

3 PSEUDOSPEKTREN IN DER REGELUNGSTECHNIK

In der Regelungstechnik wird der Einsatz von Pseudospektren erst im Bezug auf Zustandsraum-Modelle sinnvoll. Allgemein stellt man ein Zustandsraum-Modell wie folgt auf [1]:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad y(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t). \tag{29}$$

In einem allgemeinen System werden die möglichen m-Eingänge im Eingangsvektor $\mathbf{u} \in \mathbb{R}^m$ gesammelt. Die l-Ausgänge beschreibt der Ausgangsvektor $\mathbf{y}(t) \in \mathbb{R}^l$. Der Vektor $\mathbf{x}(t) \in \mathbb{R}^n$ heißt Zustandsvektor, $\mathbf{A} \in \mathbb{R}^{n\times n}$ bezeichnet die Systemmatrix, $\mathbf{B} \in \mathbb{R}^{n\times m}$ steht für die Eingangsmatrix, $\mathbf{C} \in \mathbb{R}^{l\times n}$ ist die Ausgangsmatrix und zuletzt symbolisiert $\mathbf{D} \in \mathbb{R}^{l\times m}$ die Durchgangsmatrix.
Betrachtet man nur jeweils einen Ausgang sowie Eingang, dann sieht man sich einem Single-Input Single-Output (SISO) System gegenüber. In physikalisch realisierbaren Systemen, wird die Durchgangsmatrix stets mit Null angenommen. Das Zustandraum-Modell vereinfacht sich dann zu[2]:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t), \qquad y(t) = \mathbf{c}\mathbf{x}(t) \tag{30}$$

In diesem Fall ist $\mathbf{A}$ eine Matrix, $\mathbf{b}$ und $\mathbf{c}$ sind Vektoren.

Pseudospektren können den klassischen Ansatz sinnvoll ergänzen. Wurzelortskurven und Pseudospektren hängen über die Systemmatrix $\mathbf{A}$ zusammen. Aus einem Zustandsraum-Modell lässt sich die Übertragungsfunktion $H(s)$ erzeugen, indem man das Zustandsraum-Modell Laplace transformiert. Da in der Regelungstechnik immer alle Anfangsbedingungen mit 0 angenommen werden, gilt [3]:

Häufig ist man an Polstellen von Systemen zweiter Ordnung $H(s) = K \cdot \frac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}$ interessiert. Nach Schluss des Regelkreises liefert die pq-Formel $s_{1/2} = -\xi\omega_n \pm i\omega_n\sqrt{(1+K)-\xi^2}$. Beim Plotten der Pole ist die Wurzel besonders wichtig. Daher spricht man vom `Wurzelort`.

$$s\mathbf{x}(s) = \mathbf{A}\mathbf{x}(s) + \mathbf{b}u(s)$$

Löst man für $\mathbf{x}(s)$, dann erhält man:

$$\mathbf{x}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}u(s)$$

Jetzt bietet es sich an, in die zweite Gleichung des Zustandsraum-Modells einzusetzen:

$$y(s) = \mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}u(s)$$

Die Übertragungsfunktion ist definiert als $H(s) = y(s)/u(s)$. Division

Je nach Quelle findet sich $G(s)$ oder $H(s)$, es gilt $G(s) = H(s)$.

1 Werner [29] Seite 2
2 Werner [29] Seite 2
3 Werner [29] Seite 14

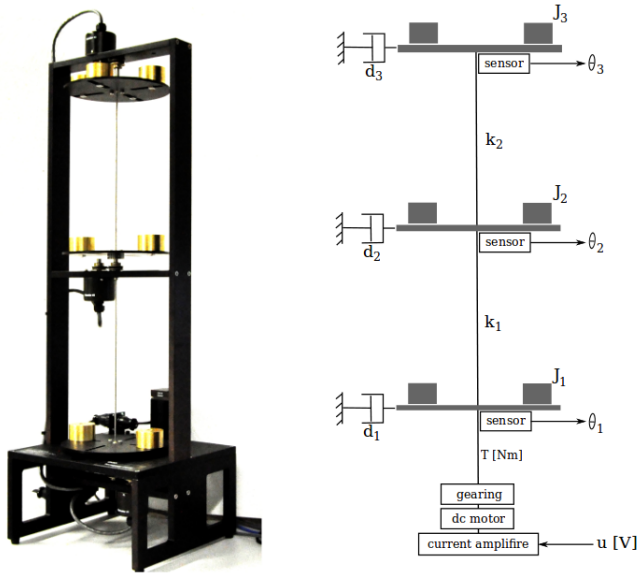


Abbildung 8: Aufbau der Torsionsregelstrecke.

durch $u(s)$ ergibt [4]:

$$H(s) = \mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}. \tag{31}$$

Um den Zusammenhang zwischen den Polen der Übertragungsfunktion und den Eigenwerten der Systemmatrix zu verdeutlichen, wird die Resolvente $(s\mathbf{I} - \mathbf{A})^{-1}$ umgeschrieben in [5]:

$$(s\mathbf{I} - \mathbf{A})^{-1} = \frac{1}{\det(s\mathbf{I} - \mathbf{A})} \operatorname{adj}(s\mathbf{I} - \mathbf{A}). \tag{32}$$

Wenn man jetzt den Nenner dieser Form gleich Null setzt, kommt man auf die Pole. Man erhält:

$$\det(s\mathbf{I} - \mathbf{A}) = 0. \tag{33}$$

Zur Störungsanfälligkeit siehe auch die Gleichungen 10 und 11 aus Kapitel 2.

Die Pole der Transferfunktion sind also die Eigenwerte der Systemmatrix. In der Regel kommt es zu Abweichungen in der Systemmatrix zum Beispiel aufgrund von Ungenauigkeiten in der Fertigung einer Maschine, deren Verhalten von der Matrix $\mathbf{A}$ beschrieben wird. Da die Eigenwerte der Systemmatrix die Pole der Transferfunktion sind, lässt sich die Pseudospektraltheorie anwenden, um zu entscheiden, wie anfällig verschiedene Eigenwerte für Ungenauigkeiten sind. Im Folgenden soll dies am Beispiel einer sich selbst tordierenden Regelstrecke veranschaulicht werden [6]. Ausgehend von vorgegebenen Differentialgleichungen soll ein Zustandsraum-Modell hergeleitet werden. Die Systemmatrix $\mathbf{A}$ kann dann anschließend analysiert werden.

4 Werner [29] Seite 6
5 Werner [29] Seite 6
6 Die Torsions-Regelstrecke sollte im Proseminar Regelungstechnik modelliert werden. Im Folgenden wird ein System verwendet, welches aus diesem Modellierungsprozess hervorgegangen ist.

Das in Abbildung 8 dargestellte System wird ausgehend vom zweiten Newtonschen Gesetz in Winkelbeschleunigungsform von folgenden Gleichungen beschrieben:

$$T = J_1\ddot{\theta_1} + d_1\dot{\theta_1} + k_1(\theta_1 - \theta_2),$$
$$0 = J_2\ddot{\theta_2} + d_2\dot{\theta_2} - k_1(\theta_1 - \theta_2) + k_2(\theta_2 - \theta_3),$$
$$0 = J_3\ddot{\theta_3} + d_3\dot{\theta_3} + k_2(\theta_3 - \theta_2).$$

T beschreibt das Motor-Drehmoment, J_i steht für das Massenträgheitsmoment der iten Platte, θ_i symbolisiert den Rotationswinkel der iten Platte, d_i beschreibt den Dämpfungskoeffizienten der auf die ite Platte wirkt und schließlich bilden k_1 und k_2 die Federsteifigkeit der unteren und der oberen Torsionsfeder ab. Durch Umstellen dieser Gleichungen in Zustandsraum-Form 30 erhält man für die Systemmatrix **A**:

Mit $\dot{\mathbf{x}} = \begin{pmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \\ \theta_3 \\ \dot{\theta}_3 \end{pmatrix}$

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{k1}{J_1} & -\frac{d_1}{J_1} & \frac{k_1}{J_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k_1}{J_2} & 0 & -\frac{k_1+k_2}{J_2} & -\frac{d_2}{J_1} & \frac{k_2}{J_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k_2}{J_3} & 0 & -\frac{k_2}{J_3} & -\frac{d_3}{J_3} \end{pmatrix}. \tag{34}$$

Setzt man jetzt die numerischen Werte $J_1 = 0.0091$, $J_2 = 0.0084$, $J_3 = 0.0075$, $d_1 = 0.0028$, $d_2 = 0.0007 = d_3$, $k_1 = 2.8234$ und $k_2 = 2.6728$ ein, dann lässt sich das System analysieren. Im Hinblick auf den Einfluss der Verstärkung auf das System kann dies mit der üblichen Wurzelortskurve geschehen. Aufgrund der Zusammenhänge 31 und 32 lassen sich auch bei Betrachtung des Pseudospektrums der Eigenwerte von **A** Schlüsse ziehen, die mit den Resultaten der klassischen Methoden vergleichbar sind. Das Pseudospektrum liefert zusätzliche Informationen bezüglich des Einflusses von Abweichungen auf das System. Im aktuell betrachteten Beispiel liegen die Eigenwerte der Matrix **A** in der Nähe der imaginären Achse und einer im Ursprung. Wollte man dieses System gegen den Einfluss ungenauer Produktion absichern, würde man intuitiv meinen, der Eigenwert im Ursprung stelle ein großes Problem dar, denn schließlich reicht hier schon eine winzige Verschiebung nach rechts und das System ist instabil. Aus dem rechten Plot in Abbildung 9 geht jedoch hervor, dass der Eigenwert im Ursprung erstaunlich robust gegenüber Fehlern ist, und dass Maßnahmen, die die Eigenwerte mit großem Imaginärteil nach links verschieben, angebracht sind. Verschiebt man die Eigenwerte mit großem Realteil nach rechts, so sinkt bei konstantem ϵ die vom Pseudospektrum umschlossene Fläche in der rechten Halbebene und damit die Wahrscheinlichkeit, dass zufällige Störungsmatrizen, deren Normen kleiner als ϵ sind, die Eigenwerte in die instabile rechte Halbebene verschieben.

$eig(\mathbf{A}) =$
$\{-0.0595 +$
$31.4292i,$
$-0.0595 - 31.4292i,$
$-0.0987 + 18.2516i,$
$-0.0987 - 18.2516i,$
$0,$
$-0.1680 + 0.0000i\ \}$

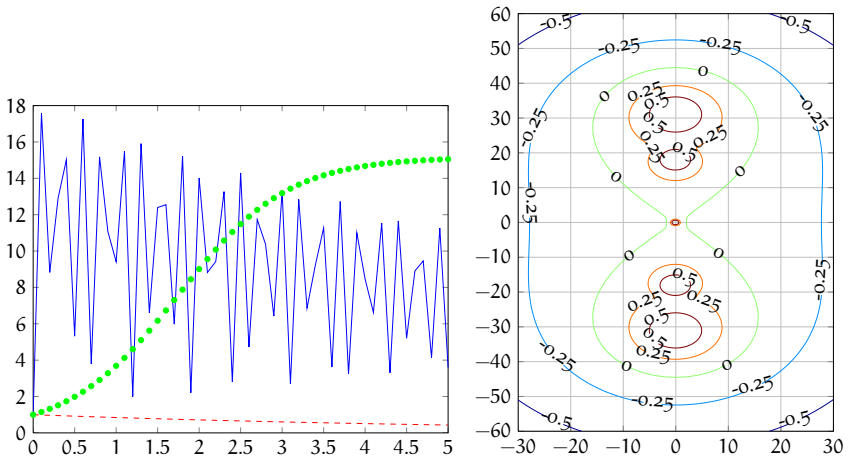


Abbildung 9: Plot des Schwungverhaltens ohne Regler (links) und des Pseudospektrums (rechts).

3.1 BETRACHTUNG DES SCHWUNGVERHALTENS DER UNGEREGELTEN TORSIONSREGELSTRECKE

Mit Hilfe der Eigenwertgrenzen aus Kapitel 2.2 lässt sich auch die Stabilität des Ausdrucks $\|e^{\mathbf{A}t}\|$ besser einschätzen. Das hier betrachtete System verhält sich in hohem Maße nichtnormal, wie aus Plot 9 hervorgeht. Potentielle Regler können mit den vorgestellten Methoden evaluiert werden, indem das Gesamtsystem aus Regler und Regelstrecke auf nichtnormales Verhalten untersucht wird. Der Regler kann dabei im Frequenzbereich mit konventionellen Verfahren entworfen werden, anschließend wird das Pseudospektrum der Matrix $\mathbf{A}$ des Gesamtsystems betrachtet und der Einfluss der Nichtnormalität beurteilt.

3.2 STÖRUNGSPLOTS ZUM ABSCHÄTZEN DES EINFLUSSES EINZELNER BAUTEILE

Um aus dem Modell Schlüsse auf die Realität ziehen zu können, reicht es nicht allein, abstrakte Störungsmatrizen zu untersuchen. Vielmehr ist der Effekt von Änderungen an einzelnen Bauteilen interessant. So könnte beispielsweise der Effekt einer steiferen unteren Feder beurteilt werden. Dazu wurde k_1 zwischen 1.5 und 3.5 variiert, es gilt $k_1 \in \{1.5, 1.6, \ldots, 3.5\}$. In Abbildung 10 ist zu erkennen, dass die Eigenwerte beim Erhöhen der Federsteifigkeit sukzessive nach links laufen. Dementsprechend würden wir ein stabileres System erwarten. Diese Erwartung deckt sich mit dem Faktum, dass Federn weniger stark schwingen, wenn ihre Steifigkeit erhöht wird.

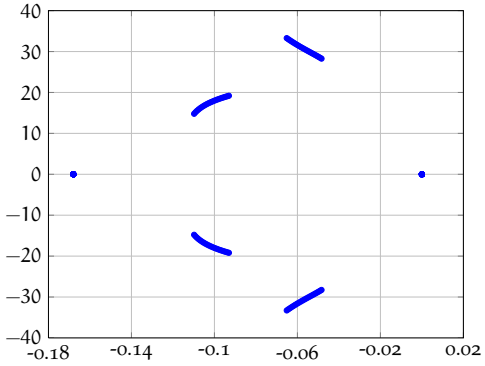


Abbildung 10: Plot der Eigenwerte der Systemmatrix bei ausschließlicher Störung der Systemgröße k_1.

3.3 $\mathbf{H}_\infty$ UND DIE PSEUDOSPEKTRAL-ABSZISSE

Regelungstechnische Systeme werden durch Zustandsräume, wie in 29 beschrieben. Wenn mehrere Ein- und Ausgänge betrachtet werden, dann beschreibt Gleichung 31 die sogenannte Transfermatrix $\mathbf{H}(s)$. Im Folgenden wird die Transfermatrix auf $\mathbf{H}(s) = (s\mathbf{I} - \mathbf{A})$ vereinfacht. Es wird also die Durchgangs-Matrix $\mathbf{D}$, sowie die Eingangsmatrix $\mathbf{B}$ und die Ausgangsmatrix $\mathbf{C}$ vernachlässigt. Die vereinfachte Gleichung beschreibt dennoch das betrachtete System adäquat, wenn die Durchgangsmatrix $\mathbf{D}$ verschwindet, in physikalisch realisierbaren Systemen ist das immer der Fall [7], und die konstanten Matrizen $\mathbf{B}$ und $\mathbf{C}$ kleine Normen haben. Die $\mathbf{H}_\infty$-Norm dient in der Regelungstechnik als Maß für robuste Stabilität. Sie ist definiert als [8]:

Zur Erinnerung: $\mathbf{H}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$.

$$\|\mathbf{H}\|_\infty = \sup_{Re s > 0} \sigma_{max}(\mathbf{H}(s)) = \sup_{\omega \in \mathbb{R}} \sigma_{max}(\mathbf{H}(i\omega)) \tag{35}$$

Robuste Stabilität bedeutet, dass man Systemstabilität gewährleisten kann, obwohl bestimmte Systemparameter Schwankungen unterworfen sind. Um trotz dieser Schwankungen Stabilität gewährleisten zu können, müssen sich diese innerhalb gewisser Grenzen bewegen [9]. Typischerweise werden für $\mathbf{H}_\infty$ zu Beginn eines Entwicklungsprozesses Grenzen festgelegt, die das System erfüllen muss. Je robuster ein System ist, desto kleiner ist seine $\mathbf{H}_\infty$-Norm.

In diesem Abschnitt firmieren zwei unterschiedliche Dinge unter ähnlichen Namen, einerseits die $\mathbf{H}_\infty$-Norm benannt, nach dem Mathematiker Godfrey Hardy, und andererseits Hamilton'schen Matrizen $\mathbf{H}$, benannt nach dem Physiker und Mathematiker Willam Hamilton.

Da die $\mathbf{H}_\infty$-Norm in der Regelungstechnik eine wichtige Rolle spielt, sind schnelle Methoden notwendig, um sie zu berechnen. Dabei ist die Theorie der Pseudospektren besonders hilfreich. Denn aus Sicht der Pseudospektraltheorie bedeutet Stabiliät, dass zu einem gewissen ϵ die Grenzkurve $\partial\sigma_\epsilon$ nicht in die rechte Halbebene hineinläuft. Oder in anderen Worten muss die Pseudospektral-Abszisse, also der Real-

7 Werner [29], Seite 10
8 Boyd [3], Seite 1 und Werner [30], Seite 88
9 Werner [30] Seite 69

teil des am weitesten rechts liegenden Punktes auf der Grenzkurve, kleiner als Null sein. Im Bezug zur $\mathbf{H}_\infty$ Norm gilt [10]:

$$\alpha_\epsilon < 0 \;\Leftrightarrow\; \|\mathbf{H}\|_\infty < \frac{1}{\epsilon} \tag{36}$$

Um die $\mathbf{H}_\infty$-Norm abzuschätzen, reicht es also, alternativ herauszufinden, ob die Pseudospektral-Abszisse kleiner als Null ist. Dazu lässt sich ein Criss-Cross-Algorithmus verwenden [11]. Dieser basiert auf dem wiederholten Bestimmen der Schnittpunkte von $\partial\sigma_\epsilon$ mit horizontalen ($\updownarrow$) und vertikalen ($\leftrightarrow$) Linien. In einem ersten Schritt wird der Eigenwert mit dem größten Realteil, also die Spektralabszisse bestimmt. Ausgehend von diesem Punkt bewegt man sich von dort horizontal weiter bis zum Schnittpunkt mit $\partial\sigma_\epsilon$. Der Realteil dieses Punktes stellt einen ersten Kandidaten für α_ϵ dar. Es sind aber bessere Lösungen denkbar. Um noch genauere Lösungen zu finden, wechseln sich ab jetzt horizontale (criss) und vertikale (cross) Suche von Iteration zu Iteration ab. Es werden wieder die vertikalen Schnittpunkte mit $\partial\sigma_\epsilon$ gesucht. Die gefundenen Punkte werden in Zweiergruppen zusammengefasst. Zu jeder Zweiergruppe wird ein Mittelpunkt gebildet. Von jedem Mittelpunkt in σ_ϵ wird nacheinander horizontal weiter gesucht. Das größte Ergebnis dieser horizontalen Suche wird Ausgangspunkt der nächsten Iteration. Dieses Prozedere wird wiederholt, bis eine zufriedenstellende Genauigkeit erreicht ist.

Bisher ist offen geblieben, wie eigentlich nach den Schnittpunkten von $\partial\sigma_\epsilon$ mit einer vertikalen durch den aktuellen Punkt gezogenen Linie gesucht wird. Die vertikale Suche basiert auf folgender Annahme: Die Matrix $(\mathbf{A} - (x + iy)\mathbf{I})$ hat nur dann einen Singulärwert ϵ, wenn iy ein Eigenwert von $\mathbf{H}_1$ ist [12]:

$$\mathbf{H}_1 = \begin{pmatrix} x\mathbf{I} - \overline{\mathbf{A}}^\top & \epsilon\mathbf{I} \\ -\epsilon\mathbf{I} & \mathbf{A} - x\mathbf{I} \end{pmatrix}. \tag{37}$$

Diese interessante Eigenschaft der Matrix $\mathbf{H}_1$ lässt sich aus den Definitionen der Eigen- und Singulärwerte herleiten [13]. Der Übersicht halber definiert man sich eine Matrix $\mathbf{B} = \mathbf{A} - x\mathbf{I}$ und einen Vektor $\mathbf{x} = (x + iy)$. Diese Definition führt auf:

$$\mathbf{H}_1 = \begin{pmatrix} -\overline{\mathbf{B}}^\top & \epsilon\mathbf{I} \\ -\epsilon\mathbf{I} & \mathbf{B} \end{pmatrix}. \tag{38}$$

10 Burke et al. [6], Seite 2

11 Burke et al. [6]

12 Burke et al. [6], Seite 3 sowei Trefethen und Embree [27] Seite 399. Die beiden Literaturstellen unterscheiden sich in einem Minuszeichen. Burke et al. setzen ein Minus bei Position $\mathbf{H}_{21}$ Trefethen und Embree bei $\mathbf{H}_{12}$. Hier wird die erstgenannte Notation verwendet, da dann der Beweis besser funktioniert.

13 Siehe auch Byers [7], Seite 2

Wenn $\mathrm{i}y \in \sigma(\mathbf{H_1})$, also wenn $\mathrm{i}y$ Eigenwert ist, dann gilt:

$$\mathbf{H_1}\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\end{pmatrix} = \mathrm{i}y\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\end{pmatrix} \Leftrightarrow \begin{pmatrix}-\overline{\mathbf{B}}^{\mathsf{T}} & \epsilon\mathbf{I}\\ -\epsilon\mathbf{I} & \mathbf{B}\end{pmatrix}\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\end{pmatrix} = \mathrm{i}y\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\end{pmatrix}.$$

Im Wesentlichen steht hier Gleichung 1, sie lautete $\mathbf{Ax} = \lambda\mathbf{x}$. Werden nun Matrix und Vektor multipliziert, und nur die obere Zeile betrachtet, so ergibt sich:

$$-\overline{\mathbf{B}}^{\mathsf{T}}\mathbf{u} + \epsilon\mathbf{v} = \mathrm{i}y\mathbf{u}$$

Umstellen und Ausklammern:

$$-(\overline{\mathbf{B}}^{\mathsf{T}} + \mathrm{i}y\mathbf{I})\mathbf{u} = -\epsilon\mathbf{v}$$

Nach beidseitigem Multiplizieren mit -1 sowie Konjugieren und Transponieren der gesamten Klammer kommt abschließend heraus:

$$\overline{(\mathbf{B} - \mathrm{i}y\mathbf{I})}^{\mathsf{T}}\mathbf{u} = \epsilon\mathbf{v} \tag{39}$$

Wenn $\mathbf{u}$ und $\mathbf{v}$ linker sowie rechter Singulärvektor sind, dann hat die Gleichung 39 die Form des Singulärwertproblems $\overline{\mathbf{A}}^{\mathsf{T}}\mathbf{u} = s\mathbf{v}$ [14]. Damit ist ϵ nur Singulärwert von $(\mathbf{B} - \mathrm{i}y\mathbf{I}) = (\mathbf{A} - x\mathbf{I} - \mathrm{i}y\mathbf{I}) = (\mathbf{A} - \mathbf{xI})$, wenn $\mathrm{i}y$ Eigenwert von $\mathbf{H_1}$ ist.

Mithilfe dieser Eigenschaft lässt sich die Matrix $\mathbf{H_1}$ zur vertikalen Suche nach Schnittpunkten mit einer Pseudopsektral-Niveaukurve verwenden. Eine x-Koordinate dient dabei als Eingangsparameter, x legt die Position einer vertikalen, also parallel zur imaginären Achse verlaufenden Linie fest. Die y-Koordinaten aller Schnittpunkte der Niveaukurve $\partial\sigma_\epsilon$ mit der gedachten Linie sind im Spektrum $\sigma(\mathbf{H_1})$ enthalten. Der Algorithmus braucht also nichts weiter zu unternehmen, als alle rein imaginären Eigenwerte λ aus dem Spektrum herauszusuchen und zu prüfen, ob $s_{min}(\mathbf{A} - (x + \lambda) = \epsilon$. In der numerischen Praxis muss man diese Gleichungen zu Ungleichungen umschreiben. Man prüft also tatsächlich $|\mathrm{Re}(\lambda)| < \mathrm{tol}$ und $|s_{min}(\mathbf{A} - (x + \lambda)) - \epsilon| < \mathrm{tol}$. Das Kürzel tol legt hier den Wert fest, bis zu dem Differenzen und Realteile als „so gut wie Null“ gelten. In Matlab könnte eine Implementierung wie folgt aussehen:

Die Zeitersparnis dieser Vorgehensweise im Gegensatz zu einer „Brute-Force“-Variante mit `for`-Schleife ist enorm, da der Suchprozess ganz ohne Schleife auskommt.

```
function [hits] = vertikaleSuche(A,eps,x,tol)
dim = length(A);
%Definiere Schift Matrix.
B = A - x*eye(dim);
%Definire die Epsilon Matrix.
Eps = eps*eye(dim);
%Erstelle die Hamitlon'sche test Matrix.
testHam = [-B' Eps; -Eps B];
hits = [];
```

14 Hogben [18], Abschnitt 58-1

```
testSigma = eig(testHam);
%pruefe fuer alle Eigenwerte
for k = 1:1:length(testSigma);
    currentEig = testSigma(k);
    %ist der Eigenwert imaginaer?
    if abs(real(currentEig)) < tol
        %Liegt dieser Punkt auf der Niveaukurve?
        sigmin = min(svd(A-(x+currentEig)*eye(dim)));
        if abs(sigmin-eps) <= tol;
            hits(end+1) = imag(currentEig);
        end
    end
end
```

Listing 8: Eine mögliche Funktion zur vertikalen Suche.

Die horizontale Suche basiert analog zur vertikalen auf einem ähnlichen Theorem: Die Matrix $(\mathbf{A} - \mathbf{xI})$, hat nur dann einen Singulärwert ϵ, wenn ix Eigenwert von $\mathbf{H_2}$ ist [15]:

$$\mathbf{H_2} = \begin{pmatrix} i\overline{\mathbf{A}}^T - y\mathbf{I} & \epsilon\mathbf{I} \\ -\epsilon\mathbf{I} & i\mathbf{A} + y\mathbf{I} \end{pmatrix} \tag{40}$$

Die horizontale Suche unterscheidet sich von der vertikalen darin, dass nun eine y-Koordinate die Position einer gedachten horizontalen Linie festlegt. Die Eigenwerte $\sigma(\mathbf{H_2})$ liefern potentielle x-Koordinaten. Analog zur vertikalen Suche ergibt sich in Matlab beispielsweise:

```
function [hits] = horizontaleSuche(A,eps,y,tol)
dim = length(A);
%Definiere die Schift Matrix.
B = A - y*1i*eye(dim);
%Definire die Epsilon Matrix.
Eps = eps*eye(dim);
%Erstelle die Hamitlon'sche test Matrix.
testHam = [1i*B' Eps;-Eps 1i*B];
hits = [];
testSigma = eig(testHam);
%pruefe fuer alle Eigenwerte.
for k = 1:1:length(testSigma);
    currentEig = testSigma(k);
    %ist der Eigenwert imaginaer?
    if abs(real(testSigma(k))) < tol
        %Liegt dieser Punkt auf der Niveaukurve
        sigmin = min(svd(A-(imag(currentEig)+y*i)*eye(dim)));
        if abs(sigmin-eps) <= tol;
            hits(end+1) = (imag(currentEig));
        end
end,end
```

Listing 9: Eine mögliche Funktion zur horizontalen Suche.

15 Burke et al. [6], Seite 5

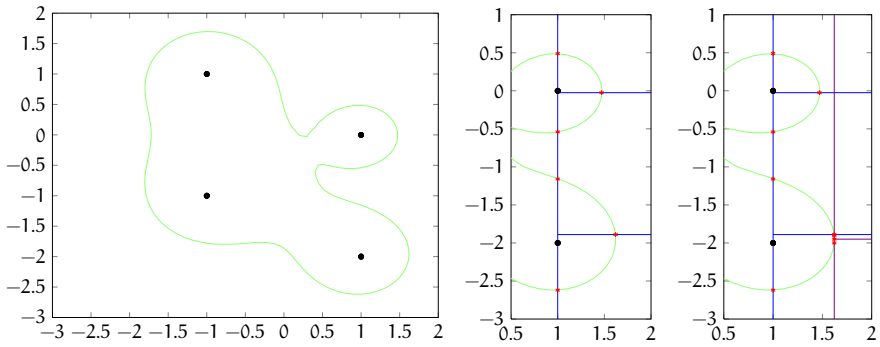


Abbildung 11: Die einzelnen Schritte des Criss-Cross-Algorithmus. Eigenwerte sind schwarz. Die erste Iteration ist blau, die zweite violett dargestellt. Treffer horizontaler und vertikaler Suchen sind mit * markiert.

In Abbildung 11 sind zwei Iterationen dargestellt. Die Suche beginnt typischerweise bei dem größten Eigenwert. Anschließend wird bei dem Treffer mit dem größten Realteil der letzten Iteration weiter gesucht. Ein Matlab-Script, in dem dies umgesetzt ist, findet sich im Anhang in Listing 11. Der Algorithmus terminiert, wenn der Betrag der Differenz der aktuell größten x-Koordinate und der aus der vorausgehenden Iteration kleiner ist als ein vorgegebener Sollwert, also wenn $|x - x_{alt}| < \text{soll}$.

Teil III

ANHANG

LITERATURVERZEICHNIS

[1] American Council of Learned Societies: *Dictionary of scientific biography*. Scribner, New York, 1981, ISBN 0684169622.

[2] Artmann, Benno und Günter Törner: *Bemerkungen zur Geschichte der Linearen Algebra*, 1981. http://logistik.math.uni-duisburg.de/pdf/1981-bemerkungen_zur_geschichte_der_lin_algebra.pdf, besucht: 2014-08-18.

[3] Boyd, S., V. Balakrishnan und P. Kabamba: *A bisection method for computing the h norm of a transfer matrix and related problems*. Mathematics of Control, Signals and Systems, 2(3):207–219, September 1989, ISSN 0932-4194, 1435-568X. http://link.springer.com/article/10.1007/BF02551385, besucht: 2014-07-01.

[4] Bronson, Richard: *Matrix methods :*. Elsevier/AP, 3. ed. Auflage, 2009, ISBN 012374427X.

[5] Brühl, Martin: *A curve tracing algorithm for computing the pseudospectrum*. BIT Numerical Mathematics, 36(3):441–454, September 1996, ISSN 0006-3835, 1572-9125. http://link.springer.com/article/10.1007/BF01731926, besucht: 2014-05-06.

[6] Burke, J., A. Lewis und M. Overton: *A robust gradient sampling algorithm for nonsmooth, nonconvex optimization*. SIAM Journal on Optimization, 15(3):751–779, Januar 2005, ISSN 1052-6234. http://epubs.siam.org/doi/abs/10.1137/030601296, besucht: 2014-06-18.

[7] Byers, Ralph: *A bisection method for measuring the distance of a stable matrix to the unstable matrices*. SIAM Journal on Scientific and Statistical Computing, 9(5):875–881, September 1988, ISSN 0196-5204, 2168-3417. https://katalog.tub.tu-harburg.de/Record/PCsiam10__D__1137__S__0909059, besucht: 2014-08-15.

[8] Corporation, Intel: *ARK | intel® core™ i5-430m processor (3m cache, 2.26 GHz)*, Dezember 2014. http://ark.intel.com/products/43537/Intel-Core-i5-430M-Processor-(3M-Cache-2_26-GHz), besucht: 2014-05-12.

[9] Czichos, Horst und Manfred Hennecke (Herausgeber): *Hütte - Das Ingenieurwissen*. Springer, Berlin, Heidelberg, 2008, ISBN 978-3-540-71851-2, 978-3-540-71852-9.

http://link.springer.com/10.1007/978-3-540-71852-9, besucht: 2014-04-28.

[10] Demmel, J.W.: *A counterexample for two conjectures about stability*. IEEE Transactions on Automatic Control, 32(4):340–342, April 1987, ISSN 0018-9286.

[11] Golub, Gene Howard und Charles Van Loan: *Matrix computations*. Johns Hopkins studies in the mathematical sciences. Johns Hopkins Univ. Press, 3. ed. Auflage, 1996, ISBN 080185413X.

[12] Hatanaka, Michio: *A theory of the pseudospectrum and its application to nonstationary dynamic econometric models,*. Princeton University, Princeton, N.J., 1963. https://www.princeton.edu/~erp/ERParchives/archivepdfs/M52.pdf, besucht: 2014-08-18.

[13] Hawkins, Thomas: *The theory of matrices in the 19th century*. Canadian Mathematical Congress, 1974. http://www.mathunion.org/ICM/ICM1974.2/Main/icm1974.2.0561.0570.ocr.pdf, besucht: 2014-08-18.

[14] Hawkins, Thomas: *Cauchy and the spectral theory of matrices*. Historia Mathematica, 2(1):1–29, Februar 1975, ISSN 0315-0860. http://www.sciencedirect.com/science/article/pii/0315086075900324, besucht: 2014-04-22.

[15] Hibbeler, Russell C.: *Engineering mechanics: dynamics*. Prentice Hall, Singapore [u.a.], 12. ed. in SI units. Auflage, 2010, ISBN 981-068137-2, 978-981-068137-1.

[16] Hibbeler, Russell Charles: *Mechanics of materials*. Prentice Hall, 3. ed Auflage, 1997, ISBN 0132569833.

[17] Hilbert, David: *Grundzüge einer allgemeinen Theorie der linearen Integralgleichungen*. Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse, 1:49–91, 1904. http://gdz.sub.uni-goettingen.de/dms/load/pdf/?PPN=PPN252457811_1904&DMDID=DMDLOG_0001&LOGID=LOG_0001&PHYSID=PHYS_0003, besucht: 2014-08-04.

[18] Hogben, Leslie (Herausgeber): *Handbook of linear algebra*. Discrete mathematics and its applications. CRC Press/Taylor & Francis Group, Boca Raton, Florida, second edition Auflage, 2014, ISBN 9781466507289.

[19] Landau, H. J.: *Loss in unstable resonators*. Journal of the Optical Society of America, 66(6):525–529, Juni 1976. http://www.opticsinfobase.org/abstract.cfm?URI=josa-66-6-525, besucht: 2014-06-23.

[20] Lui, S.: *Computation of pseudospectra by continuation*. SIAM Journal on Scientific Computing, 18(2):565–573, März 1997, ISSN 1064-8275. http://epubs.siam.org/doi/abs/10.1137/S1064827594276035, besucht: 2014-06-17.

[21] Mihajlović, Borivoj: *The application of (mihail petrović's) modified pseudospectra to the solution of differential equations*. Mat. Vesnik, 4 (19):119–122, 1967. http://www.digizeitschriften.de/dms/img/?PPN=PPN311571026_0019&DMDID=dmdlog35, besucht: 2014-08-16.

[22] Pasciak, Joseph E.: *Spectral and pseudospectral methods for advection equations*. Mathematics of Computation, 35(152):1081–1092, 1980, ISSN 0025-5718, 1088-6842. http://www.ams.org/mcom/1980-35-152/S0025-5718-1980-0583488-0/, besucht: 2014-06-24.

[23] Reichard, Hans (Herausgeber): *Kleine Enzyklopaedie Mathematik*. Verlag Harry Deutsch, 1980, ISBN 3871443239.

[24] Shure, Loren: *MATLAB central - loren on the art of MATLAB » using parfor loops: Getting up and running » using parfor loops: Getting up and running*, Oktober 2009. http://blogs.mathworks.com/loren/2009/10/02/using-parfor-loops-getting-up-and-running/#4, besucht: 2014-05-12.

[25] Strang, Gilbert: *Introduction to linear algebra*. Wellesley-Cambridge Press, 4. ed. Auflage, 2009, ISBN 0980232716.

[26] Trefethen, L.: *Pseudospectra of linear operators*. SIAM Review, 39(3):383–406, Januar 1997, ISSN 0036-1445. http://epubs.siam.org/doi/abs/10.1137/S0036144595295284, besucht: 2014-05-03.

[27] Trefethen, Lloyd Nicholas und Embree Mark: *Spectra and pseudospectra*. Princeton Univ. Press, 2005, ISBN 0691119465.

[28] Varah, J.: *On the separation of two matrices*. SIAM Journal on Numerical Analysis, 16(2):216–222, April 1979, ISSN 0036-1429. http://epubs.siam.org/doi/abs/10.1137/0716016, besucht: 2014-06-23.

[29] Werner, Herbert: *Control Systems Theory and Design*. Technische Universität Hamburg-Harburg, Oktober 2013.

[30] Werner, Herbert: *Optimal and Robust Control*. Technische Universität Hamburg-Harburg, Mai 2014. https://e-learning.tu-harburg.de/studip/sendfile.php?type=0&file_id=1ca126b29c5932fc4b52f68939b9a0f2&file_name=orc.pdf, besucht: 2014-08-18.

[31] Wieleitner, Heinrich: *Geschichte der Mathematik von 1700 bis zur Mitte des 19. Jahrhunderts*. Wieleitner, Heinrich / von Heinrich Wieleitner Geschichte der Mathematik. de Gruyter, neue bearb Auflage, 1923.

.1 ZUSATZABBILDUNGEN

In diesem Teil des Anhanges finden sich im Rahmen dieser Arbeit erstellte interessante Abbildungen, die im Text keinen Platz mehr gefunden haben.

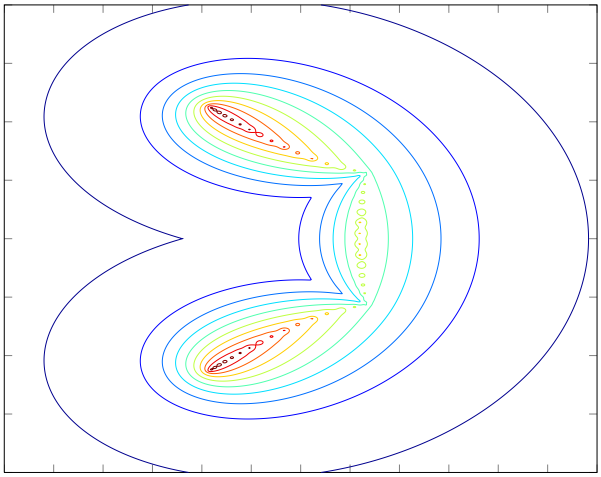


Abbildung 12: Pseudospektrum der `gallery('grcar',50,3)`-Test-Matrix.

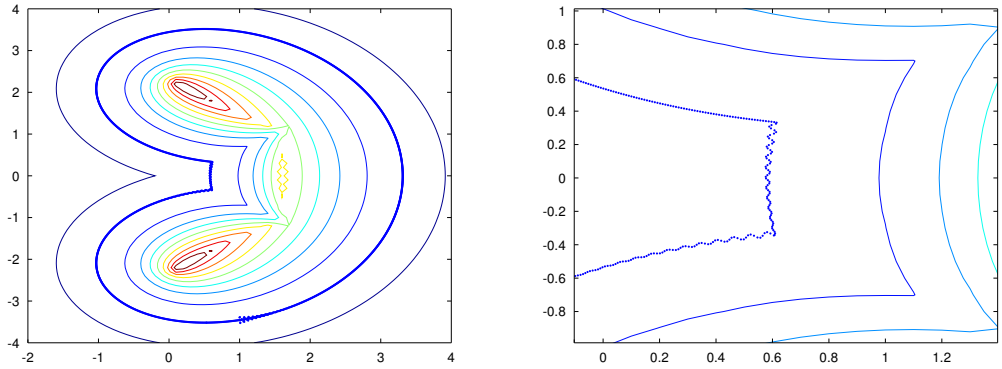


Abbildung 13: Trace einer Niveaukurve in dickem Blau der `gallery('grcar',50,3)`-Test-Matrix (links). Beim Tracing auftretende Oszillation (rechts). Beide Ausgaben einer abgewandelten Form von Listing 5 wurden mit Abbildung 12 unterlegt.

Im Rahmen dieser Arbeit entstandene Quellcode Dateien, die zu groß für den Fließtext sind. Im sich anschließenden Listing findet sich der in Kapitel 2 beschriebene Lanczos Algrithmus:

```
function [] = invLanc(A,x,y,dim,maxit)
        %invLanc(A,x,y,maxit)
        %Nach LNT Seite 375
        %Berechne und plotte das Pseudospectrum mit
        %inversen Lanczos-Iterationen.
        tic;
        m = length(x);
        n = length(y);
        N = dim(1);

        %Schur-Zerlegung in obere Dreiecksform.
        T = schur(A,'complex')
        %Laufe durch die komplexe Ebene in for-Schleife.
        for k=1:m, for j=1:n
                %Beginne mit der iterativen berechnung des
                    Pseudospektrums
                %Initialisiere Lanczos variablen.
                T1 = (x(k)+y(j)*i)*eye(N)-T;
                T2 = T1';
                sigold = 0; qold = zeros(N,1); beta = 0; H = [];
                %Waehle einen zufaelligen genormten Lanczos
                    vektor.
                q = randn(N,1)+i*randn(N,1); q = q/norm(q);
                %Fuehre Lanczos iterationen bis maxit aus.
                %Ergebins equivalent zu min(svd(x(k)+y(k)*i-exe(N
                    )))
                for p = 1:maxit
                        %Berechne die Eintraege der
                            Bidiagonalmatrix H.

                        %Achtung: a \ b = a^(-1) * b
                                        %ziehe den alten Vektor
                                            ab.
                        v = T1\(T2\q) - beta*qold;
                        alpha = real(q'*v);
                        v = v - alpha*q;
                        beta = norm(v);
                        qold = q;
                        q = v/beta; % = v/norm(v);

                        %Setzte die neu gefundenen Eintraege in
                            die Vergleichsmatrix H ein.
                        H(p+1,p) = beta;
                        H(p,p+1) = beta;
                        H(p,p) = alpha;

                        %Bestimme den groessten Eigenwert von H.
```

```
                        try
                                sig = max(eig(H(1:p, 1:p)));
                        catch
                        %Wenn sig singulaer verwende 0. Denn -
                            log_10(0) -> inf.
                                warning('Berechnung von sig
                                    gescheitert.');
                                sig=0;
                                break;
                        end;
                        %Konvergenzkriterium wenn (sigold/sig-1)
                            <0.001 dann brich ab.
                        if abs(sigold/sig-1)<1e-3
                         break
                        end
                        sigold = sig;
                end
                sigmin(j,k) = sqrt(sig);
        end, end
        toc;
        contour(x,y,-log10(sigmin))
```

Listing 10: Inverser Lanczos-Algorithmus.

Eine mögliche Form des in Kapitel 3 beschriebenen Criss-Cross Algorithmus:

```
%Matlab zum Verstaendis von Bruke et al.'s crissCross Algorithmus
    .
clear all;

%Beispiel Matrix A
A = [1 0.5 0 0; 0 -1-1i 2 0; 0 0 -1+1i 3; 0 0 0 1-2i];
%x = 0.5:0.1:2;
%y = -3:0.1:1;
x  = -3:0.1:2;
y  = -3:0.1:2;
labels = 0.45;
drawPseudo(A,x,y,labels);
hold on;

eps = 0.45;
dim = length(A);
%setze eine Toleranz;
tol = 0.1;

%Finde die Eigenwerte
sigma = eig(A);
%Nimm den mit dem groessten Realteil.
sigmaMax = max(real(sigma));
%Suche von hier aus nach Schnittpunkten mit der Grenzlinie.
x = sigmaMax;
xalt = 0;
```

```
while abs(x-xalt) > 0.01
    %vertikale Suche.
    vertHits = vertikaleSuche(A,eps,x,tol);
    for i = 1:1:length(vertHits)
        plot(x,vertHits(i),'r*')
        hold on;
    end
    %Setzte x neu als den Mittelpunk aus vorherigen Treffern.
    yToCheck = [];
    vertHits = sort(vertHits);
    for i=1:1:(length(vertHits)-1)
        %Berechne potentielle x Startpunkte
        potY =  (vertHits(i)+vertHits(i+1))/2;
        if min(svd(A-(x+1i*potY)*eye(dim))) <= eps
            yToCheck(end+1) = potY;
        end
    end
    %Leere die Ergebnisliste der vertikalen suche.
    vertHits = [];
    horHits = [];
    %Horizontale Suche
    for i=1:1:length(yToCheck)
        potHorHits = horizontaleSuche(A,eps,yToCheck(i),tol);
        potHorHits = sort(potHorHits);
        %Das groesste Element steht am Ende.
        horHits(end+1) = potHorHits(end);
        plot(potHorHits(end),yToCheck(i),'r*')
        hold on;
    end
    horHits = sort(horHits);
    xalt = x;
    x = horHits(end)
end
```

Listing 11: CrissCross-Algorithmus.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

VERSICHERUNG

Ich versichere hiermit diese Arbeit selbstständig angefertigt zu haben. Literaturstellen anderer Autoren habe ich zitiert.

Hamburg, 19. August 2014

Moritz Wolter