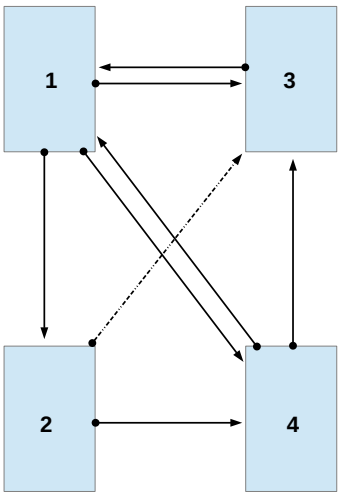
Kapitel 2

Eigenwerte und Eigenvektoren

Gemäß einer Aussage von Kurt Bryan muss ein Suchmaschinenanbieter, zum Beispiel *Google*, das Netz (*World Wide Web*) durchforsten und Information auf den Webseiten sammeln und (in einem geeignetem Format) speichern.[1] Anschließend muss er in der Lage sein, auf Anfrage, die relevaten Links zu finden und sie in einer geeigneten Reihenfolge listen. Aber was ist eine "geeignete Reihenfolge"? Ein erster Ansatz die Relevanz (Wichtigkeit / *importance*) einer Seite einzuschätzen besteht darin, die Anzahl ihrer Rückverweise (*backlinks*) zu zählen; dies ist auch Teil des *Google PageRank* Algorithmus).



Bezeichne x_k die Anzahl der Rückverweise auf Seite k:

- $x_1 = 2$
- $x_2 = 3$ (nahezu irrelevant)
- $x_3 = 3$ (Listenplatz 1)
- $x_4 = 2$

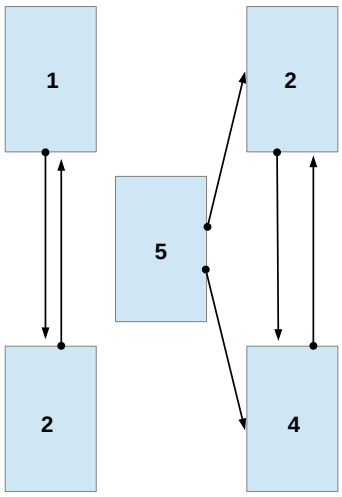
Aber sollte nicht ein Link von *spiegel.de* mehr zählen als einer von *tu-ilmenau.de/de/analysis/team/karl-worthmann*?

Diese Frage regt die folgende Verbesserung an: der Beitrag eines Rückverweises wird durch die Wichtigkeit der verweisenden Seite (x_j) dividiert durch die Anzahl ihrer Rückverweise (n_j) bestimmt, also x_j/n_j. Dann ergibt für unser Beispiel

$$\begin{aligned} x_1 &= x_3/1 + x_4/2 \\ x_2 &= x_1/3 \\ x_3 &= x_1/3 + x_2/2 + x_4/2 \\ x_4 &= x_1/3 + x_2/2 \end{aligned} \qquad \text{bzw.} \qquad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix}}_{=:A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}}_{=:x}.$$

[1] http://www.rose-hulman.edu/~bryan/invprobs/usafagoogle.pdf

Es ist also $x = Ax$ zu lösen beziehungsweise es ist der Eigenvektor x der Matrix A zum Eigenwert 1 gesucht. Eine Lösung ist beispielsweise $x \approx (0.387\ 0.129\ 0.290\ 0.194)^T$ bzw. beliebige Vielfache dieses Eigenvektors; solch ein Eigenvektor existiert immer, falls A eine *stochastische Matrix* (quadratische Matrix, deren Spaltensummen Eins betragen und deren Elemente zwischen Null und Eins liegen) ohne sogenannte *dangling nodes* (Knoten ohne ausgehende Kante) ist. Gewünschte Eigenschaften dieses Eigenvektors wären gemäß Bryan, dass er nicht negativ (was ist negative Relevanz?), eindeutig bis auf Skalierung (eindimensionaler Eigenraum) sowie "leicht" zu berechnen ist (für eine sehr große Matrix). Dazu betrachten wir das Beispiel



mit Verweismatrix

$$A = \left(\begin{array}{cc|ccc} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 1/2 \\ 0 & 0 & 1 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right).$$

$\rightsquigarrow$ zwei linear unabhängige Eigenvektoren $x_1 = (0.5\ 0.5\ 0\ 0\ 0)^T$ und $x_2 = (0\ 0\ 0.5\ 0.5\ 0)^T$ zum Eigenwert 1.

Theorem: Besitzt ein Netz r Zusammenhangskomponenten, dann hat der von Eigenvektoren zum Eigenwert 1 aufgespannte Unterraum mindenstens Dimension r.

Als Ausweg kann man jeder Seite eine "schwache Verknüpfung" mit jeder anderen geben; A wird dann durch die gewichtete Kombination $M = (1-m)A + mS$ mit $m \in (0,1)$ und $S = (s_{ij})$, $s_{ij} = 1/n$ für alle $(i,j) \in \{1,2,\ldots,n\} \times \{1,2,\ldots,n\}$, ersetzt. Diese Matrix besitzt den dominanten Eigenwert 1, dessen Eigenraum Dimension 1 hat. Zudem existiert ein eindeutiger ($\rightsquigarrow$ Reihenfolge im Ranking eindeutig) nicht negativer Eigenvektor x mit $\sum_{i=1}^{n} x_i = 1$.[2] Folglich wird eine Methode benötigt, die es erlaubt einen Eigenvektor einer 8.000.000.000 $\times$ 8.000.000.000-Matrix zu finden. Für Details (auch bzgl. weiterer Anmerkungen zu diesem Beispiel sei auf den sehr gut lesbaren Zeitschriftenbeitrag [1] verwiesen.

Tatsächlich spielen Eigenwerte (und Eigenvektoren) von Matrizen in vielen Anwendungen eine Rolle; so geben die Eigenwerte bei der Betrachtung iterativer Verfahren Auskunft über die Konvergenz. Dies ist ein generelles Prinzip linearer Iterationen (ähnlich ist dies bei linearen Differentialgleichungen) und ein wichtiges Beispiel für eine Problemklasse, bei der die Kenntnis der Eigenwerte einer Matrix wichtig ist. Eine weitere Anwendung in der Bildverarbeitung wird auf dem 6. Übungsblatt behandelt.

Gesucht sind bei einem Eigenwertproblem diejenigen $\lambda \in \mathbb{C}$, für die die Gleichung

$$Av = \lambda v$$

für einen *Eigenvektor* $v \in \mathbb{C}^n$ erfüllt ist. In vielen Anwendungen, zum Beispiel im oben beschriebenen Seitenranking von Google ist man darüberhinaus an den zugehörigen Eigenvektoren v interessiert.

[2] Der Suchmaschinenanbieter *Google* nutzt angeblich $m = 0.15$.

Wir werden in diesem relativ kurzen Kapitel einige Algorithmen zur Berechnung von Eigenwerten und zugehörigen Eigenvektoren für spezielle Matrizen (z.B. symmetrische Matrizen) kennen lernen. Bevor wir mit konkreten Algorithmen beginnen, wollen wir uns allerdings mit der Kondition des Eigenwertproblems beschäftigen.

Bevor wir zu numerischen Verfahren zur Berechnung von Eigenwerten kommen, wollen wir kurz die vielleicht naheliegendste Methode untersuchen, nämlich die Berechnung der λ über die Nullstellen des charakteristischen Polynoms. Diese Methode ist numerisch äußerst schlecht konditioniert (unabhängig von der Kondition der Eigenwertberechnung selbst) und bereits kleinste Rundungsfehler können sehr große Fehler im Ergebnis nach sich ziehen. Als Beispiel betrachte das Polynom

$$P(\lambda) = (\lambda - 1)(\lambda - 2) \cdots (\lambda - 20)$$

mit den Nullstellen $\lambda_i = i$ für $i = 1, \ldots, 20$.[3] Wenn dieses Polynom als charakteristisches Polynom einer Matrix berechnet wird (z.B. ist es gerade das charakteristische Polynom $\chi_A(\lambda)$ der Matrix $A = \mathrm{diag}(1, 2, \ldots, 20)$), liegt es üblicherweise nicht in der obigen "Nullstellen"-Darstellung sondern in anderer Form vor, z.B. ausmultipliziert. Wenn man das obige $P(\lambda)$ ausmultipliziert, ergeben sich Koeffizienten zwischen 1 (für λ^{20}) und $20! \approx 10^{20}$ (der konstante Term). Stört man nun den Koeffizienten vor λ^{19} (der den Wert 210 hat) mit dem sehr kleinen Wert $\varepsilon = 2^{-23} \approx 10^{-7}$, so erhält man die folgenden Nullstellen für das gestörte Polynom $\widetilde{P}(\lambda) = P(\lambda) - \varepsilon\lambda^{19}$:

$$\begin{aligned}
\lambda_1 &= 1.000\,000\,000 & \lambda_{10/11} &= 10.095\,266\,145 \pm 0.643\,500\,904\,i \\
\lambda_2 &= 2.000\,000\,000 & \lambda_{12/13} &= 11.793\,633\,881 \pm 1.652\,329\,728\,i \\
\lambda_3 &= 3.000\,000\,000 & \lambda_{14/15} &= 13.992\,358\,137 \pm 2.518\,830\,070\,i \\
\lambda_4 &= 4.000\,000\,000 & \lambda_{16/17} &= 16.730\,737\,466 \pm 2.812\,624\,894\,i \\
\lambda_5 &= 4.999\,999\,928 & \lambda_{18/19} &= 19.502\,439\,400 \pm 1.940\,330\,347\,i \\
\lambda_6 &= 6.000\,006\,944 & \lambda_{20} &= 20.846\,908\,101 \\
\lambda_7 &= 6.999\,697\,234 & & \\
\lambda_8 &= 8.007\,267\,603 & & \\
\lambda_9 &= 8.917\,250\,249 & &
\end{aligned}$$

Die winzige Störung bewirkt also beachtliche Fehler, insbesondere sind 10 Nullstellen durch die Störung komplex geworden.

2.1 Vektoriteration

Die einfachste Möglichkeit der Berechnung von Eigenwerten ist die Vektoriteration, die sich entweder als *direkte Iteration* (auch bekannt als *von Mises-Iteration* oder *power iteration*) oder als *inverse Iteration* (auch *inverse power iteration*) durchführen lässt.

Gegeben sei eine reelle Matrix $A \in \mathbb{R}^{n \times n}$. Die Idee der direkten Iteration beruht darauf, für einen beliebigen Startwert $x^{(0)} \in \mathbb{R}^n$ die Iteration

$$x^{(i+1)} = Ax^{(i)} \tag{2.1}$$

[3] Das Beispiel stammt von dem englischen Numeriker James H. Wilkinson (1919–1986), der die Entdeckung dieses Polynoms angeblich als "the most traumatic experience in my career as a numerical analyst" bezeichnet hat.

durchzuführen. Dass dieses einfache Verfahren tatsächlich unter gewissen Bedingungen einen Eigenwert liefert, zeigt der folgende Satz. Wir erinnern hierbei daran, dass symmetrische reelle Matrizen nur reelle Eigenwerte besitzen.

Satz 2.1 Sei $A \in \mathbb{R}^{n\times n}$ eine symmetrische Matrix und $\lambda_1 = \lambda_1(A)$ ein einfacher Eigenwert, für den die Ungleichung

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|$$

für alle anderen Eigenwerte $\lambda_j = \lambda_j(A)$ gilt. Sei weiterhin $x^{(0)} \in \mathbb{R}^n$ ein Vektor, für den $\langle x^{(0)}, v_1\rangle \neq 0$ für den zu $\lambda_1(A)$ gehörigen (normierten) Eigenvektor v_1 gilt. Dann konvergiert die Folge $y^{(i)} := x^{(i)}/\|x^{(i)}\|$ für $x^{(i)}$ aus (2.1) gegen $\pm v_1$, also gegen einen normierten Eigenvektor zum Eigenwert λ_1. Insbesondere konvergiert damit der sogenannte *Rayleigh'sche Quotient*

$$\lambda^{(i)} := \frac{\langle Ax^{(i)}, x^{(i)}\rangle}{\langle x^{(i)}, x^{(i)}\rangle} = \langle Ay^{(i)}, y^{(i)}\rangle$$

gegen den Eigenwert λ_1.

Beweis: Wegen der Symmetrie von A existiert eine Orthonormalbasis von Eigenvektoren $v_1, \ldots, v_n$ von A. Damit gilt

$$x^{(0)} = \sum_{j=1}^{n} \alpha_j v_j \quad \text{mit } \alpha_i = \langle x^{(0)}, v_i\rangle,$$

insbesondere also $\alpha_1 \neq 0$. Daraus folgt

$$x^{(i)} = A^i x^{(0)} = \sum_{j=1}^{n} \alpha_j \lambda_j^i v_j = \alpha_1 \lambda_1^i \underbrace{\left(v_1 + \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1}\right)^i v_j \right)}_{=:z^{(i)}}.$$

Da $|\lambda_j| < |\lambda_1|$ ist für $j = 2, \ldots, n$, gilt $\lim_{i\to\infty} z^{(i)} = v_1$ und damit

$$y^{(i)} = \frac{x^{(i)}}{\|x^{(i)}\|} = \pm \frac{z^{(i)}}{\|z^{(i)}\|} \to \pm v_1.$$

Die Konvergenz $\lambda^{(i)} \to \lambda_1$ folgt, indem wir $y^{(i)} = v_1 + r^{(i)}$ mit $r^{(i)} \to 0$ schreiben. Dann gilt

$$\begin{aligned} \langle Ay^{(i)}, y^{(i)}\rangle &= \langle A(v_1 + r^{(i)}), v_1 + r^{(i)}\rangle \\ &= \langle Av_1, v_1\rangle + \underbrace{\langle Ar^{(i)}, v_1\rangle + \langle Av_1, r^{(i)}\rangle + \langle Ar^{(i)}, r^{(i)}\rangle}_{\to 0} \\ \to \langle Av_1, v_1\rangle &= \langle \lambda_1 v_1, v_1\rangle = \lambda_1 \|v_1\| = \lambda_1. \end{aligned}$$

□

Beachte, dass die Symmetrie der Matrix A hier nur eine hinreichende aber keine notwendige Bedingung für die Konvergenz des Verfahrens ist. So ist z.B. in [1] bewiesen, dass das Verfahren auch für die (nicht symmetrische) Matrix aus dem Seitenranking funktioniert.

Hier kann überdies gezeigt werden, dass $\lambda_2 = 1 - m$ gilt; dies führt auf $\lambda_i/\lambda_1 \leq 0.85$ für $i = 2, 3, \ldots, n$. Rechnentechnisch ergibt sich aber scheinbar das Problem, dass die Matrix M dicht besetzt (keine Nulleinträge; *dense*) ist. Dies kann basierend auf der Beobachtung, dass sich $M = (1 - m)A + mS$ aus der dünn besetzten (*sparse*) Matrix A (Links auf andere Webseiten im Verhältnis zur Gesamtzahl aller Webseiten) und der Matrix S, deren Einträge alle gleich $1/n$ sind, zusammensetzt wie folgt gelöst werden:

- Für einen Vektor v mit $\sum_{i=1}^{n} v_i = 1$ gilt deshalb $Sv = (1/n\ 1/n\ \ldots 1/n)^T$.
- Av kann effizient berechnet werden (so sind zum Beispiel unter der Annahme, dass jede Seite im Durchschnitt 10 Links enthält nur $1.6 \cdot 10^{11}$ Operationen nötig. Für eine direkte Berechnung von Mv wären es ungefähr $1.3 \cdot 10^{20}$ gewesen; $n = 8 \cdot 10^9$).

Folglich kann die Matrix-Vektor-Multiplikation Mv mittels $(1-m)Av+(m/n\ m/n\ \ldots m/n)^T$ mit vertretbarer Rechenzeit ausgeführt werden.

Das Vorgehen von *Google* hat weitere Anwendungen gefunden, zum Beispiel in der Spieltheorie oder der Medizin. Zudem zog K. Bryan zwei weitere Schlußfolgerungen:

- "Sophisticated linear algebra is at the core of much scientific computation, and pops up when you least expect it."
- "If you pay attention to your math professors, you might become a billionaire."

Die direkte Vektoriteration ist zwar aufgrund ihrer Einfachheit attraktiv, birgt aber mehrere Nachteile: Erstens erhalten wir nur den (betragsmäßig) größten (und kleinsten, siehe Übungsblatt 6) Eigenwert $|\lambda_1|$ und den zugehörigen Eigenvektor, zweitens hängt die Konvergenzgeschwindigkeit davon ab, wie schnell die Terme $|\lambda_j/\lambda_1|^i$, also insbesondere $|\lambda_2/\lambda_1|^i$ gegen Null konvergieren. Falls also $|\lambda_1| \approx |\lambda_2|$ und damit $|\lambda_2/\lambda_1| \approx 1$ gilt, ist nur sehr langsame Konvergenz zu erwarten.

Die *inverse Vektoriteration* vermeidet diese Nachteile. Sei A wiederum eine reelle symmetrische Matrix. Wir setzen voraus, dass wir einen Schätzwert $\tilde{\lambda} \in \mathbb{R}$ für einen Eigenwert $\lambda_j = \lambda_j(A)$ kennen, für den die Ungleichung

$$|\tilde{\lambda} - \lambda_j| < |\tilde{\lambda} - \lambda_k| \text{ für alle } k = 1, \ldots, n,\ k \neq j$$

mit $\lambda_k = \lambda_k(A)$ gilt. Dann betrachten wir die Matrix $\widetilde{A} = (A - \tilde{\lambda}\mathrm{Id})^{-1}$. Diese besitzt die Eigenwerte $1/(\lambda_k - \tilde{\lambda})$ für $k = 1, \ldots, n$, also ist $1/(\lambda_j - \tilde{\lambda})$ der betragsmäßig größte Eigenwert.

Die inverse Vektoriteration ist nun gegeben durch

$$x^{(i+1)} = (A - \tilde{\lambda}\mathrm{Id})^{-1} x^{(i)}. \tag{2.2}$$

Aus Satz 2.1 (angewendet auf $(A - \tilde{\lambda}\mathrm{Id})^{-1}$ an Stelle von A) folgt, dass diese Iteration gegen einen normierten Eigenvektor v_j von $(A - \tilde{\lambda}\mathrm{Id})^{-1}$ zum Eigenwert $1/(\lambda_j - \tilde{\lambda})$ konvergiert. Wegen

$$\begin{aligned} & (A - \tilde{\lambda}\mathrm{Id})^{-1} v_j &&= 1/(\lambda_j - \tilde{\lambda}) v_j \\ \Leftrightarrow \quad & (\lambda_j - \tilde{\lambda}) v_j &&= (A - \tilde{\lambda}\mathrm{Id}) v_j \\ \Leftrightarrow \quad & \lambda_j v_j &&= A v_j \end{aligned}$$

ist dies gerade ein Eigenvektor von A zum Eigenwert λ_j. Die Konvergenzgeschwindigkeit ist bestimmt durch den Term

$$\max_{\substack{k=1,\dots,n \\ k\neq j}} \frac{|\lambda_j - \tilde{\lambda}|}{|\lambda_k - \tilde{\lambda}|}.$$

Je kleiner dieser Term ist, d.h. je besser der Schätzwert ist, desto schneller wird die Konvergenz.

Die tatsächliche Implementierung der Iteration (2.2) ist hier etwas komplizierter als bei der direkten Iteration (2.1). Während dort in jedem Schritt eine Matrix-Vektor Multiplikation mit Aufwand $O(n^2)$ durchgeführt werden muss, geht hier die Inverse $(A-\tilde{\lambda}\mathrm{Id})^{-1}$ ein. In der Praxis berechnet man nicht die Inverse (weil dies numerisch sehr aufwändig ist), sondern löst das lineare Gleichungssystem

$$(A - \tilde{\lambda}\mathrm{Id})x^{(i+1)} = x^{(i)}, \tag{2.3}$$

wobei bei der Verwendung eines direkten Verfahrens die Matrix $(A - \tilde{\lambda}\mathrm{Id})$ nur einmal am Anfang der Iteration faktorisiert werden muss und dann in jedem Iterationsschritt einmal Vorwärts- bzw. Rückwärtseinsetzen durchgeführt werden muss. Der Aufwand $O(n^3)$ der Zerlegung kommt also hier zum Aufwand des Verfahrens dazu, die einzelnen Iterationsschritte haben bei diesem Vorgehen allerdings keinen höheren Rechenaufwand als bei der direkten Iteration, da das Vorwärts- bzw. Rückwärtseinsetzen wie die Matrix-Vektor Multiplikation den Aufwand $O(n^2)$ besitzen.

Für sehr gute Schätzwerte $\tilde{\lambda} \approx \lambda_j$ wird die Matrix $(A-\tilde{\lambda}\mathrm{Id})$ "fast" singulär (für $\tilde{\lambda} = \lambda_j$ wäre sie singulär), weswegen die Kondition von $(A-\tilde{\lambda}\mathrm{Id})$ sehr groß wird. Wegen der besonderen Struktur des Algorithmus führt dies hier aber nicht auf numerische Probleme, da zwar die Lösung $x^{(i+1)}$ des Gleichungssystems (2.3) mit großen Fehlern behaftet sein kann, sich diese Fehler aber in der hier eigentlich wichtigen *normierten Lösung* $x^{(i+1)}/\|x^{(i+1)}\|$ nicht auswirken. Wir wollen dies an einem Beispiel illustrieren.

Beispiel 2.2 Betrachte

$$A = \begin{pmatrix} -1 & 3 \\ -2 & 4 \end{pmatrix}$$

mit den Eigenwerten $\lambda_1(A) = 2$ und $\lambda_2(A) = 1$. Wir wählen $\tilde{\lambda} = 1 - \varepsilon$ für ein sehr kleines $\varepsilon > 0$. Dann ist die Matrix

$$(A - \tilde{\lambda}\mathrm{Id}) = \begin{pmatrix} -2+\varepsilon & 3 \\ -2 & 3+\varepsilon \end{pmatrix} \quad \text{mit} \quad (A - \tilde{\lambda}\mathrm{Id})^{-1} = \frac{1}{\varepsilon(\varepsilon+1)} \underbrace{\begin{pmatrix} 3+\varepsilon & -3 \\ 2 & -2+\varepsilon \end{pmatrix}}_{=:B}$$

fast singulär und man sieht leicht, dass für die Kondition z.B. in der Zeilensummennorm die Abschätzung $\mathrm{cond}_\infty(A-\tilde{\lambda}\mathrm{Id}) > 1/\varepsilon$ gilt. Die Inverse besitzt aber eine typische spezielle Struktur: die großen Einträge, die der Grund für die große Kondition sind, entstehen lediglich durch einen skalaren Vorfaktor. Daher ist die Berechnung von $y^{(i+1)} = x^{(i+1)}/\|x^{(i+1)}\|$ mittels (2.3) nicht stark anfällig für Rundungsfehler, denn es gilt

$$y^{(i+1)} = \frac{(A - \tilde{\lambda}\mathrm{Id})^{-1}x^{(i)}}{\|(A - \tilde{\lambda}\mathrm{Id})^{-1}x^{(i)}\|} = \frac{Bx^{(i)}}{\|Bx^{(i)}\|_2},$$

d.h. der ungünstige große Faktor $1/(\varepsilon(\varepsilon+1))$ kürzt sich heraus. Ein Zahlenbeispiel illustriert dies noch einmal: Betrachten wir beispielsweise $x^{(i)} = (1,0)^T$ und $\varepsilon = 1/1000$, so erhält man

$$x^{(i+1)} = (A - \tilde{\lambda}\mathrm{Id})^{-1}x^{(i)} = \begin{pmatrix} 2998.001998 \\ 1998.001998 \end{pmatrix} \text{ und } y^{(i+1)} = \frac{x^{(i+1)}}{\|x^{(i+1)}\|_2} = \begin{pmatrix} 0.832135603 \\ 0.554572211 \end{pmatrix}$$

während man für die gestörte Lösung mit $\tilde{x}^{(i)} = (1.1, -0.1)^T$

$$\tilde{x}^{(i+1)} = (A - \tilde{\lambda}\mathrm{Id})^{-1}\tilde{x}^{(i)} = \begin{pmatrix} 3597.502498 \\ 2397.502498 \end{pmatrix} \text{ und } \tilde{y}^{(i+1)} = \frac{\tilde{x}^{(i+1)}}{\|\tilde{x}^{(i+1)}\|_2} = \begin{pmatrix} 0.832117832 \\ 0.554598875 \end{pmatrix}$$

erhält. Die Störung in der ersten Nachkommastelle der rechten Seite bewirkt in $\tilde{x}^{(i+1)}$ zwar einen Fehler von etwa 600 und verstärkt sich daher sichtlich. In dem für den Algorithmus eigentlich wichtigen Vektor $\tilde{y}^{(i+1)}$ ist der Effekt der Störung hingegen erst in der fünften Nachkommastelle der Lösung sichtbar. □

Bemerkung 2.3 [QR-Algorithmus] Zwar kann man mit der inversen Vektoriteration im Prinzip alle Eigenwerte berechnen, benötigt dafür aber geeignete Schätzwerte. Eine Alternative ist der QR-Algorithmus [3, Algorithmus 3.5], der in einer einzigen Rechnung alle Eigenwerte einer Matrix approximiert. Wie bereits bei linearen Gleichungssystemen spielt auch hier eine Faktorisierung mittels orthogonaler Matrizen eine wichtige Rolle, was auch die Nomenklatur erklärt.

Eine Herleitung für reelle symmetrische Matrizen findet sich in [3]; dort wird ebenfalls kurz auf eine Verallgemeinerung auf allgemeine Matrizen eingehen. Die Grundidee für reelle symmetrische Matrizen besteht darin, dass für solche Matrizen eine Orthonormalbasis aus Eigenvektoren $v_1, v_2, \ldots, v_n$ besteht, so dass für die orthogonale Matrix $Q = (v_1, \ldots, v_n) \in \mathbb{R}^{n\times n}$ die Gleichung

$$Q^T AQ = \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

gilt, wobei die λ_i gerade die Eigenwerte von A sind. Dann können die Eigenwerte direkt aus der resultierenden Diagonalmatrix Λ und die zugehörigen Eigenvektoren aus Q abgelesen werden.

Um den Rechenaufwand zu reduzieren wird die Matrix A in einem vorbereitenden Schritt zunächst auf eine möglichst einfache Form zu bringen, um die Anzahl der Rechenoperationen in der Iteration klein zu halten $\rightsquigarrow \mathcal{O}(n^2)$, siehe [3, Bemerkung 3.9 (i)]. Hierbei wählt man die *Tridiagonalgestalt* und nutzt aus, dass man Householder-Matrizen dazu verwenden kann, Matrizen auf Tridiagonalgestalt zu konjugieren. Grundlage dafür ist [3, Lemma 3.3], das – für eine reelle symmetrische Matrix A – (konstruktiv) die Existenz einer orthogonalen Matrix P zeigt, so dass $P^T AP$ symmetrisch und in Tridiagonalgestalt ist.[4] □

[4] Für nicht symmetrische Matrizen wählt man die sogenannte obere Hessenberg-Gestalt.