

Methoden der Numerik

Christina Eilers, Julian Lüken

4. April 2019

Mathematisches Institut Göttingen

Aufgabe 1 - Wärmegleichung

Wärmegleichung

Die Wärmegleichung lautet

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u$$

Mit $u : \Omega \times \mathbb{R}^+ \rightarrow \mathbb{R}$ mit folgenden Randbedingungen:

- $u(x, t) = R$ für $x \in \partial\Omega$
- $u(x, 0) = f(x)$, wobei f beliebig aber fest.

Diskretisierung der Wärme Gleichung

Nimm endlich viele, äquidistante Stellen aus Ω , sodass Folgen entstehen mit $x_i = ih + x_0$ und $y_j = jh + y_0$. Wähle zusätzlich für die Zeit $t_k = k\Delta t + t_0$ Wir schreiben $u_{i,j}^k$ für die (i,j) -te Stelle zum Zeitpunkt k . Für jedes $k \in \mathbb{N}$ entsteht eine $m \times m$ Matrix. Zum Zeitpunkt k haben wir dann:

$$\begin{pmatrix} u_{0,0}^k & u_{0,1}^k & \cdots & u_{0,m}^k \\ u_{1,0}^k & \ddots & & \vdots \\ \vdots & & & \\ u_{m,0}^k & \cdots & & u_{m,m}^k \end{pmatrix}$$

Diskretisierung der Wärme Gleichung

Schreibe diese Matrix als Vektor, damit wir einen linearen Operator in Form einer Matrix darauf anwenden können, folgendermaßen:

$$u^k = \begin{pmatrix} u_{0,0}^k \\ u_{0,1}^k \\ \vdots \\ u_{0,m}^k \\ u_{1,0}^k \\ u_{1,1}^k \\ \vdots \\ u_{m,m}^k \end{pmatrix}$$

Diskretisierung

Aus der Taylor-Entwicklung folgt für die Ableitung nach t

$$\frac{\partial u}{\partial t}(x, y, t) = \frac{u(x, y, t + \Delta t) - u(x, y, t)}{\Delta t} + O(\Delta t^2)$$

und für den Laplace-Operator

$$\begin{aligned} \nabla^2 u(x, y, t) = \frac{1}{4h^2} & \left(u(x - h, y, t) + u(x + h, y, t) \right. \\ & \left. + u(x, y - h, t) + u(x, y + h, t) - 4u(x, y, t) \right) + O(h^4) \end{aligned}$$

Für diese Terme können wir im diskreten Fall unter Vernachlässigung des Fehlers schreiben

$$\frac{u_{i,j}^{k+1} - u_{i,j}^k}{\Delta t} \quad \text{und} \quad \frac{1}{4h^2} \left(u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k - 4u_{i,j}^k \right)$$

Explizites Euler-Verfahren

Sei ab jetzt $h = 1$ und $\Delta t = 1$. Für das explizite Euler-Verfahren suchen wir eine Matrix $A \in \mathbb{R}^{m^2 \times m^2}$ für die gilt:

$$u^{k+1} = u^k + \alpha A u^k$$

Mit den aus der Taylor-Entwicklung gewonnenen Diskretisierungen für unsere Differentialoperatoren erhalten wir mit ein paar Umformungen die Iterationsvorschrift

$$u_{i,j}^{k+1} = \alpha \left(\frac{u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k}{4} \right) + (1 - \alpha) u_{i,j}^k$$

Explizites Euler-Verfahren

Im expliziten Euler-Verfahren sieht die Verfahrensmatrix dann so aus:

$$A = \frac{1}{4} \begin{bmatrix} L & I & 0 & \dots & 0 \\ I & L & I & & \\ 0 & I & L & & \\ \vdots & & & \ddots & \vdots \\ & & & & L & I \\ 0 & \dots & & I & L \end{bmatrix} \quad \text{wobei} \quad L = \text{tridiag}(1, -4, 1)$$

Damit wird das explizite Euler-Verfahren in unserem Fall zu

$$u^{k+1} = u^k + \alpha A u^k = (\alpha A + I) u^k$$

Implizites Euler-Verfahren

Im impliziten Euler-Verfahren gilt es nun, eine Iterationsvorschrift der Form

$$u^{k+1} = u^k + \alpha B u^{k+1}$$

zu finden. Aus dem expliziten Euler-Verfahren lässt sich dafür

$$B = A(\alpha A + I)^{-1}$$

herleiten. Wir gewinnen daraus ein Gleichungssystem in der Form $Cx = d$, welches wir in jedem Schritt nach x lösen können

$$u^k = (I - \alpha A(\alpha A + I)^{-1})u^{k+1}$$

Bemerkung:

Hierfür eignet sich das CG-Verfahren.

Weiterhin ist das LGS nur lösbar, wenn $\alpha \neq 1$

Durch Umstellen der Wärmegleichung mit A von vorhin können wir durch folgende Iterationsvorschrift mit gegebenen Startwerten u^0 den Verlauf der Wärmegleichung simulieren:

$$u^{k+1} = (\alpha A + I)u^k$$

Bemerkung:

Wenn man u^k als Matrix schreibt, so kann man ebenfalls mit $L_\alpha = 0.5 \cdot \alpha \cdot \text{tridiag}(1, -2, 1) + I$ den gleichen Schritt folgendermaßen durchführen:

$$u^{k+1} = \frac{1}{2}(L_\alpha u^k + u^k L_\alpha)$$

Zwar werden so pro Schritt zwei Matrixmultiplikationen und eine Addition durchgeführt, allerdings nur mit $m \times m$ Matrizen statt mit $m^2 \times m^2$ Matrizen.

Konjugierte Gradienten

Wir erinnern uns an das implizite Euler-Verfahren mit dem Gleichungssystem

$$u^k = (I - \alpha A(\alpha A + I)^{-1})u^{k+1}$$

Um dieses zu lösen, definieren wir

$$K = (I - \alpha A(\alpha A + I)^{-1})$$

In jedem Schritt unserer Iteration lösen wir nun das Gleichungssystem

$$u^k = Ku^{k+1}$$

mithilfe des Algorithmus der konjugierten Gradienten.

Konjugierte Gradienten

Wir wählen $b = u^k$. In jedem Schritt führen wir aus

Algorithm 1 Konjugierte-Gradienten Verfahren

- 1: $x_0 \leftarrow b, k \leftarrow 0, r_0 \leftarrow b - Kx_0, d_0 \leftarrow r_0$
 - 2: **while** $r_k^T r_k > \varepsilon$ **do**
 - 3: $\alpha_k = \frac{r_k^T r_k}{d_k^T K d_k}$
 - 4: $x_{k+1} \leftarrow x^k + \alpha_k d_k$
 - 5: $r_{k+1} \leftarrow r_k - \alpha_k K d_k$
 - 6: $\beta_k \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$
 - 7: $d_{k+1} \leftarrow r_{k+1} + \beta_k d_k$
 - 8: $k \leftarrow k + 1$
 - 9: **end while**
 - 10: **return** x_k
-

Gauss-Elimination

Man kann das LGS ebenfalls mit Gauss-Elimination lösen. Zunächst brauchen wir dafür die LU-Zerlegung $K = LU$, wobei L untere Dreiecksmatrix und U obere Dreiecksmatrix.

Algorithm 2 LU-Zerlegung

```
1:  $U \leftarrow K, L \leftarrow I$ 
2: for  $i \in \{1 \dots n\}$  do
3:   for  $k \in \{i + 1 \dots n\}$  do
4:      $L_{ki} \leftarrow \frac{U_{ki}}{U_{ii}}$ 
5:     for  $j \in \{i \dots n\}$  do
6:        $U_{kj} \leftarrow U_{kj} - L_{ki} \cdot U_{ij}$ 
7:     end for
8:   end for
9: end for
10: return  $L, U$ 
```

Gauss-Elimination

Jetzt löst man durch Vorwärtselimination

$$Ly = u^k$$

und danach durch Rückwärtselimination

$$Uu^{k+1} = y$$

Bemerkung:

Die LU -Zerlegung von K müssen wir für unsere Zwecke nur einmal bestimmen.

Aufgabe 2 - Newton-Verfahren

Löse

$$\min_{x \in \mathbb{R}^7} f(x) = \frac{1}{2} \|g(x)\|_2^2$$

mit $g : \mathbb{R}^7 \rightarrow \mathbb{R}^8$, $\nabla g : \mathbb{R}^7 \rightarrow \mathbb{R}^{7 \times 8}$ und $\nabla^2 g : \mathbb{R}^7 \rightarrow \mathbb{R}^{7 \times 7 \times 8}$. $x_0 = 0$ und für die Lösung soll gelten $\|\nabla f(\bar{x})\|_2 < 10^{-11}$

Newton-Verfahren

Im Newton-Verfahren nutzen wir die Iterationsvorschrift

$$x_{n+1} = x_n - (\nabla^2 f(x))^{-1} \nabla f(x)$$

Oder in Pseudocode:

Algorithm 3 Newton-Verfahren

```
1:  $x \leftarrow 0 \in \mathbb{R}^7$ 
2: while  $\|\nabla f(x)\|_2 \geq 10^{-11}$  do
3:    $H_{\text{inv}} \leftarrow \nabla^2 f(x)$ 
4:    $x \leftarrow x - H_{\text{inv}} \nabla f(x)$ 
5: end while
6: return  $x$ 
```

Bemerkung:

Das funktioniert nur, wenn $\nabla^2 f$ existiert und bekannt ist.

Quasi-Newton-Verfahren

Algorithm 4 Quasi-Newton-Verfahren

```
1:  $x_0 \leftarrow 0 \in \mathbb{R}^7$ 
2:  $H_0 = I$ 
3:  $p_0 = -\nabla f(x_0)$ 
4:  $k = 0$ 
5: while  $\|\nabla f(x)\|_2 \geq 10^{-11}$  do
6:    $\alpha \leftarrow \operatorname{argmin}_{\alpha \in \mathbb{R}} f(x_k + \alpha p_k)$ 
7:    $s_k \leftarrow \alpha p_k$ 
8:    $x_{k+1} \leftarrow x_k + s_k$ 
9:    $y_k \leftarrow \nabla f(x_{k+1}) - \nabla f(x_k)$ 
10:   $H_k = \text{update}(H_k)$ 
11:   $k \leftarrow k + 1$ 
12: end while
13: return  $x_k$ 
```

Quasi-Newton-Updates

Für die Funktion update auf der letzten Folie gibt es verschiedene Möglichkeiten, beispielsweise:

$$\begin{array}{l|l} \text{BFGS} & H_{k+1} = H_k + \frac{y_k y_k^T}{y_k^T s_k} + \frac{H_k s_k s_k^T H_k^T}{s_k^T H_k s_k} \\ \text{Broyden} & H_{k+1} = H_k + \frac{s_k - H_k y_k}{y_k^T y_k} y_k^T \end{array}$$

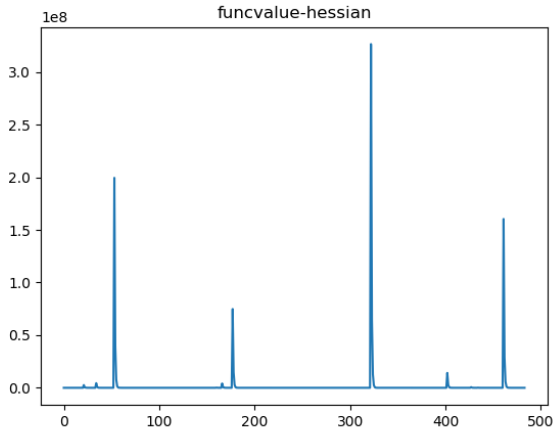
Lösungen

	klassisch	BFGS	Broyden
$f(x)$	2291.020503182124	2291.0205031821224	2291.020503182123
$\nabla f(x)$	$\begin{pmatrix} 0 \\ -1.137 \\ 0.941 \\ -0.284 \\ 5.684 \\ 0 \\ 0 \end{pmatrix} \cdot 10^{-13}$	$\begin{pmatrix} -2.274 \\ 67.075 \\ 21.245 \\ -7.248 \\ 37.232 \\ -2.256 \\ -2.487 \end{pmatrix} \cdot 10^{-13}$	$\begin{pmatrix} -3.411 \\ -14.780 \\ -4.814 \\ 1.705 \\ 86.118 \\ -3.606 \\ 0.249 \end{pmatrix} \cdot 10^{-13}$

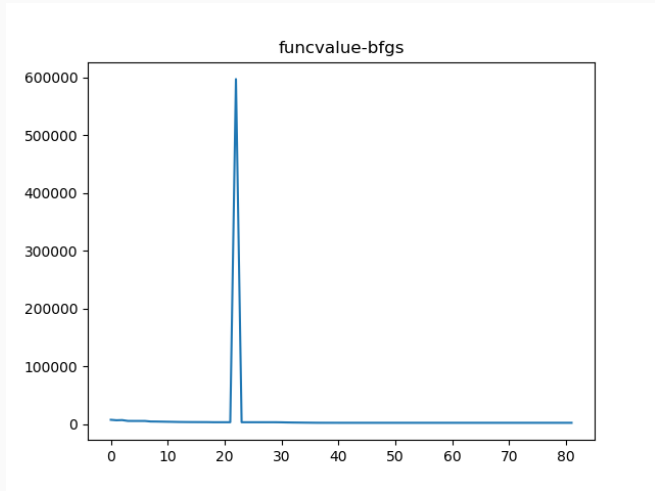
Hesse-Matrizen

$$H^{-1} = \begin{pmatrix} 2.151 & 0.598 & 3.538 & 16.956 & -0.078 & -2.615 & -0.039 \\ 0.598 & 3.288 & 0.121 & 22.659 & 0.310 & 13.016 & -0.011 \\ 3.538 & 0.121 & 13.622 & 45.657 & -0.220 & -8.099 & -0.260 \\ 16.956 & 22.659 & 45.657 & 323.998 & 1.294 & 58.392 & -0.897 \\ -0.078 & 0.310 & -0.220 & 1.294 & 0.148 & 2.717 & 0.001 \\ -2.615 & 13.016 & -8.099 & 58.392 & 2.717 & 93.675 & 0.047 \\ -0.039 & -0.011 & -0.260 & -0.897 & 0.001 & 0.047 & 3.256 \end{pmatrix} \cdot 10^{-3}$$
$$H_{\text{BFGS}}^{-1} = \begin{pmatrix} 1.992 & 0.337 & 3.517 & 14.757 & -0.068 & -3.207 & -0.025 \\ 0.337 & 2.532 & 0.301 & 17.282 & 0.273 & 10.557 & -0.003 \\ 3.517 & 0.301 & 13.449 & 46.290 & -0.189 & -7.443 & -0.278 \\ 14.757 & 17.282 & 46.290 & 284.436 & 1.163 & 42.147 & -0.838 \\ -0.068 & 0.273 & -0.189 & 1.163 & 0.135 & 2.584 & -0.014 \\ -3.207 & 10.557 & -7.443 & 42.147 & 2.584 & 80.445 & 0.562 \\ -0.025 & -0.003 & -0.278 & -0.838 & -0.014 & 0.562 & 3.115 \end{pmatrix} \cdot 10^{-3}$$
$$H_{\text{Broyden}}^{-1} = \begin{pmatrix} 1.003 & 0.353 & 2.534 & 10.787 & -0.024 & -0.808 & -0.305 \\ 0.016 & 0.559 & 1.185 & 4.623 & 0.082 & 4.472 & -0.130 \\ 0.035 & 0.129 & 9.330 & 30.184 & -0.035 & -1.636 & -0.773 \\ 0.301 & 4.001 & 62.421 & 234.475 & -0.381 & 2.729 & -5.924 \\ -0.171 & 0.436 & -1.168 & -0.179 & 0.145 & 2.657 & 0.025 \\ 1.258 & -0.488 & 22.002 & 72.069 & 0.185 & 7.744 & -1.962 \\ 0.009 & -0.057 & -1.015 & -3.702 & 0.007 & -0.004 & 1.718 \end{pmatrix} \cdot 10^{-3}$$

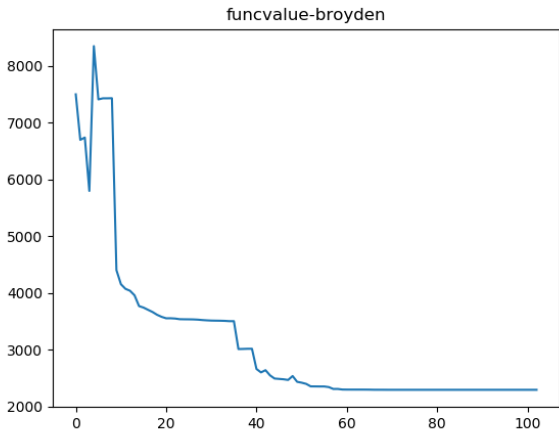
Funktionswerte für Newton-Verfahren



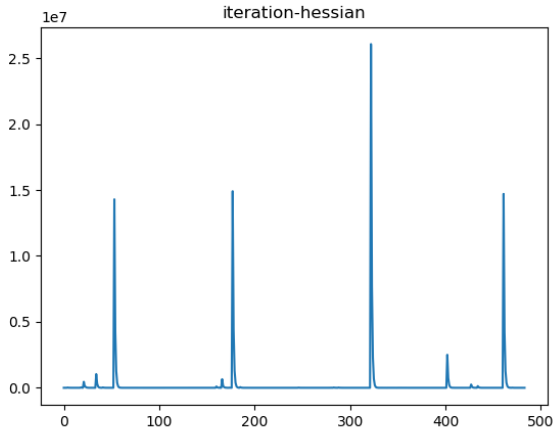
Funktionswerte für Quasi-Newton: BFGS



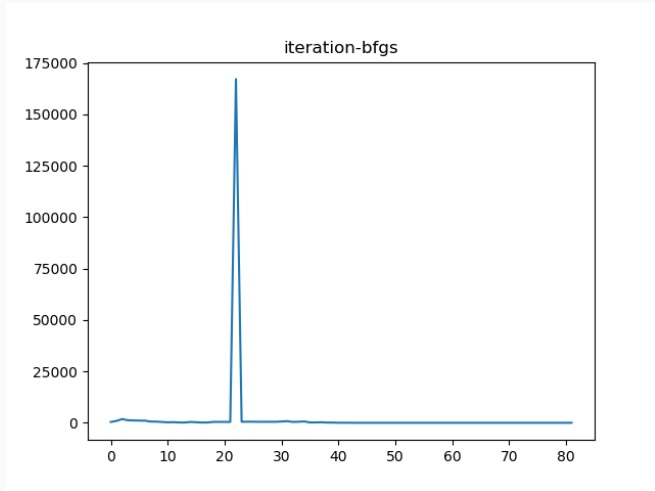
Funktionswerte für Quasi-Newton: Broyden



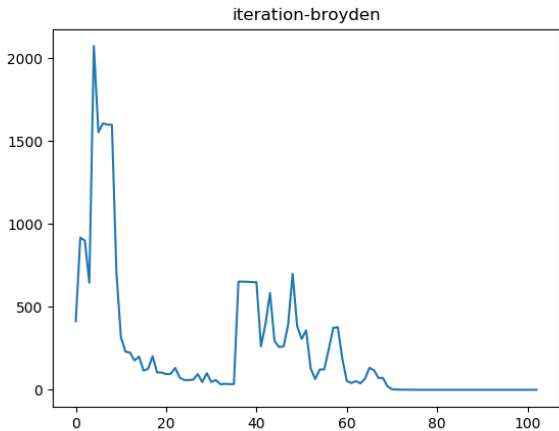
Grad für Newton-Verfahren



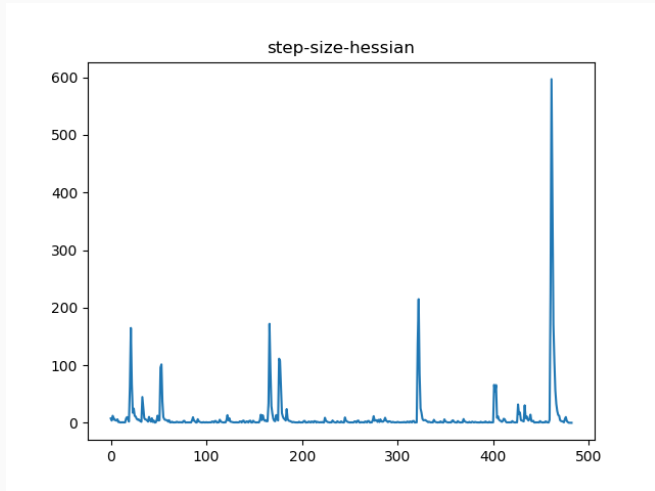
Grad für Quasi-Newton: BFGS



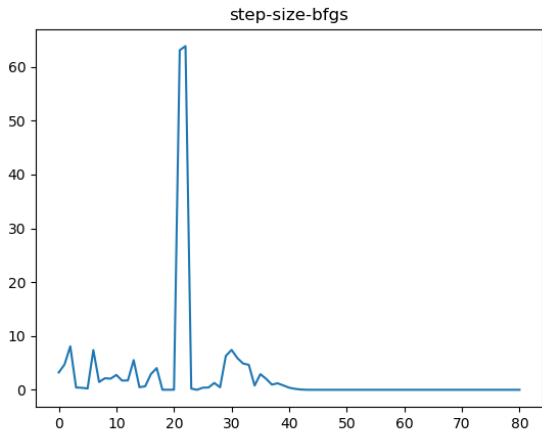
Grad für Quasi-Newton: Broyden



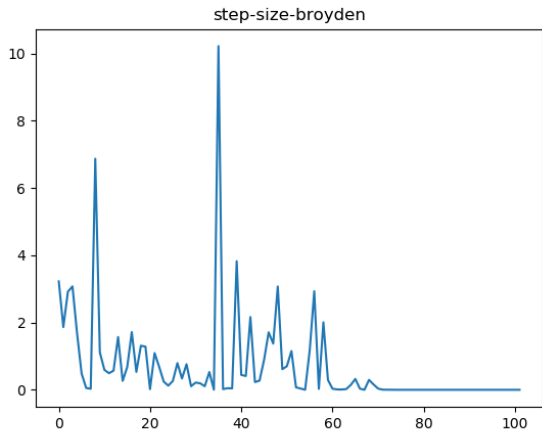
Schrittweite für Newton-Verfahren



Schrittweite für Quasi-Newton: BFGS



Schrittweite für Quasi-Newton: Broyden



Aufgabe 3 - Norm

Problemstellung

Sei

$$A = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{7} \\ \frac{1}{3} & \frac{1}{5} & \frac{1}{8} & \\ \frac{1}{6} & \frac{1}{9} & & \ddots \\ \frac{1}{10} & & \ddots & \end{pmatrix}$$

Was ist $\|A\|_2$ auf 10 signifikante Stellen genau?

Lösungsverfahren: Teil I

Es gilt nach Vorlesung

$$\|A_n\|_2 \leq \|A_m\|_2 \leq \lim_{k \rightarrow \infty} \|A_k\|_2 = \|A\|_2 \leq \|A\|_F = \frac{\pi}{\sqrt{6}}$$

mit $n \leq m$ und A_n der linken oberen $n \times n$ -Teilmatrix von A .

Sei jetzt A_k die linke obere $k \times k$ Teilmatrix von A und $B := A_k^T A_k$. B ist symmetrisch \rightarrow alle Eigenwerte sind reell. Sei $v \in \mathbb{R}^k$ der EV von B zum betragsgrößten EW λ .

$$\begin{aligned} Bv &= \lambda v \rightarrow \|Bv\|_2 = \|\lambda v\|_2 = |\lambda| \|v\|_2 \\ \rightarrow \sqrt{\|A_k^T A_k v\|_2} &= \sqrt{\|Bv\|_2} = \sqrt{|\lambda| \|v\|_2} = \sqrt{|\lambda|} \sqrt{\|v\|_2} \\ \rightarrow \frac{\sqrt{\|A_k^T A_k v\|_2}}{\sqrt{\|v\|_2}} &= \sqrt{|\lambda|} = \sqrt{\rho(B)} = \|A_k\|_2 \end{aligned}$$

Untere Grenze

Erstelle A_n für ein $n \in \mathbb{N}$ und berechne

$$\|A_n\|_2 = \sqrt{\rho(A_n^T A_n)} = \sqrt{\rho(B)}$$

Das von-Mises-Verfahren liefert schnell eine Abschätzung an den betragsgrößten Eigenwert λ von B und gibt auch den dazugehörigen Eigenvektor v zurück. $\alpha = \sqrt{|\lambda|}$ ist untere Grenze von $\|A\|_2$

Lösungsverfahren: Teil II

Der EV v ist normiert und hat nur positive Einträge. Außerdem sind die Einträge mit steigendem Index monoton fallend. Ist in

$$\frac{\sqrt{\|A_k^T A_k v\|_2}}{\sqrt{\|v\|_2}} = \sqrt{|\lambda|} = \|A_k\|_2$$

$\|v\|_2$ etwas größer als 1, so $\sqrt{\|A_k^T A_k v\|_2} > \|A_k\|_2$

Ist $\|v\|_2$ groß genug, so ist sogar $\sqrt{\|A_k^T A_k v\|_2} > \|A\|_2$

Obere Grenze

Das von Mises-Verfahren gibt einen EV $v \in \mathbb{R}^n$ zurück. Erweitere diesen auf $m \gg n$ Dimensionen und nenne ihn \tilde{v} folgendermaßen:

$$\tilde{v}_i := \begin{cases} v_i & \text{für } i \in \{1, \dots, n\} \\ v_n & \text{für } i \in \{n+1, \dots, m\} \end{cases}$$

Bemerkung:

$\|\tilde{v}\|_2$ ist etwas größer als 1

Algorithm 5 Finden einer oberen Grenze für $\|A\|_2$

- 1: $x \leftarrow A_m \tilde{v}$
 - 2: $y \leftarrow A_m^T x$
 - 3: $\beta = \sqrt{\|y\|_2}$
 - 4: **return** β , eine obere Grenze von $\|A\|_2$
-

Vergleich der Grenzen ergibt:

$$\|A\|_2 = 1.2742241528 \quad (11 \text{ Stellen})$$

Um hinreichend genaue Ergebnisse zu erreichen, genügt $n = 1800$ und $m = 2n$

n	m	alpha	beta	Genauigkeit
8000	10000	1.2742241528116438	1.2742241528249134	11
8000	16000	1.2742241528116438	1.2742241528221396	11
12000	24000	1.2742241528182108	1.2742241528211977	11
6200	20000	1.2742241528011655	1.2742241528269063	11
6200	50000	1.2742241528011655	1.2742241528300235	11
1800	100000	1.2742241520093243	1.2742241533011174	9
1800	4500	1.2742241520093243	1.2742241529847291	10
1800	3600	1.2742241520093243	1.2742241528986895	10