III Das Symmetrische Eigenwertproblem (SEP)

III.3 Algorithmen für symmetrische tridiagonale Eigenwertprobleme

Sei im folgenden

$$A = \begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-2} & a_{n-1} & b_{n-1} \\ & & & b_{n-1} & a_n \end{bmatrix} \in \mathbb{R}^{n\times n}, \tag{III.1}$$

z.B. nach Householder- oder Lanczos(im Kapitel IV)-Tridiagonalisierung.

Bisektionsverfahren Sei $A \in \mathbb{R}^{n\times n}$; definiere $A_r := A(1:r, 1:r)$ für $r = 2, \ldots, n$ mit charakteristischem Polynom

$$p_r(x) = \det(A_r - \lambda I).$$

Definiert man weiter $p_0(x) \equiv 1$, sowie $p_{-1}(x) \equiv 0$, dann folgt

$$p_r(x) = (a_r - x)p_{r-1}(x) - b_{r-1}^2 p_{r-2}(x),$$

denn

$$\det(A_r - \lambda I) = \det \begin{bmatrix} a_1-\lambda & b_1 & & & \\ b_1 & a_2-\lambda & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{r-2} & a_{r-1}-\lambda & b_{r-1} \\ & & & b_{r-1} & a_r-\lambda \end{bmatrix}$$

$$\overset{\text{Entw. nach}}{\underset{\text{letzter Zeile}}{=}} (a_r - \lambda)\det(A_{r-1} - \lambda I) - b_{r-1} \underbrace{\det \begin{bmatrix} a_1-\lambda & b_1 & & & \\ b_1 & a_2-\lambda & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{r-3} & a_{r-2}-\lambda & 0 \\ & & & b_{r-2} & b_{r-1} \end{bmatrix}}_{\overset{\text{Entw. nach}}{\underset{\text{letzter Spalte}}{=}} b_{r-1}\det(A_{r-2}-\lambda I)}$$

$$= (a_r - \lambda)\det(A_{r-1} - \lambda I) - b_{r-1}^2 \det(A_{r-2} - \lambda I).$$

Diese Rekursionsformel erlaubt die Auswertung von $p_n(x) = \chi_A(x)$ in $\mathcal{O}(n)$ `flops`. Die Berechnung der Eigenwerte durch Bisektion (Intervallschachtelung, Algorithmus III.6) liefert damit ein Verfahren die Eigenwerte direkt aus dem charakteristischen Polynom zu bestimmen.

Algorithmus III.6 Bisektionsverfahren

Input: $y < z \in \mathbb{R}$ mit $p_n(y)p_n(z) < 0 \quad (\Rightarrow \exists \xi \in (y,z) : p_n(\xi) = 0)$
Output: ξ mit $p_n(\xi) = 0$

```
while |z − y| > TOL(|y| + |z|) do
  x= (y+z)/2
  if p_n(x)p_n(y) < 0 then
    z = x
  else
    y = x
  end if
end while
```

Die Konvergenz im Algorithmus III.6 ist linear, denn $|\lambda - x_{k+1}| \leq \frac{1}{2}|\lambda - x_k|$.

Der folgende Satz liefert uns eine Möglichkeit bestimmte Eigenwerte von A zu berechnen. Suchen wir z.B. zu $\mu \in \mathbb{R}$ den nächstkleineren Eigenwert, dann können wir im Algorithmus III.6 $z = \mu$ wählen und mit dem Satz y so bestimmen, dass genau eine Nullstelle zwischen y und z liegt (also eine weniger links von y als links von z). Seien im folgenden die Eigenwerte immer angeordnet nach $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

Satz III.7 (Sturm'sche Folge). *Sei $A = A_n$ unreduzierte symmetrische Tridiagonalmatrix, dann separieren die Eigenwerte von A_{r-1} die Eigenwerte von A_r derart, dass*

$$\lambda_r(A_r) < \lambda_{r-1}(A_{r-1}) < \lambda_{r-1}(A_r) < \cdots < \lambda_2(A_r) < \lambda_1(A_{r-1}) < \lambda_1(A_r)$$

Weiter sei

$$a(\mu) = \textit{Anzahl Vorzeichenwechsel in } \{p_0(\mu), p_1(\mu), \ldots, p_n(\mu)\}$$

mit der Konvention, dass eine Nullstelle ($p_j(\mu) = 0$) einem Vorzeichenwechsel entspricht, dann gilt

$$a(\mu) = \#\{\lambda \in \Lambda(A) \mid \lambda < \mu\}$$

Beweis. [GV96, Theorem 8.5.1] □

Bemerkung (Trennungseigenschaft). *Es gilt $\forall A = A^T$, $A_r := A(1:r, 1:r)$:*

$$\lambda_{r+1}(A_{r+1}) \leq \lambda_r(A_r) \leq \lambda_r(A_{r+1}) \leq \cdots \leq \lambda_2(A_{r+1}) \leq \lambda_1(A_r) \leq \lambda_1(A_{r+1})$$

Beispiel III.8. *Sei $\mu = 2$ wir suchen die Anzahl der Eigenwerte links von μ zu*

$$A = \begin{bmatrix} 1 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 3 & -1 \\ & & -1 & 4 \end{bmatrix} . \overset{\mu=2}{\Rightarrow} \begin{array}{l} p_0(2) = 1 \\ p_1(2) = (1-2)\cdot 1 - (-1)^2 \cdot 0 = -1 \\ p_2(2) = (2-2)(-1) - (-1)^2 \cdot 1 = -1 \\ p_3(2) = (3-2)(-1) - (-1)^2 \cdot (-1) = 0 \\ p_4(2) = (4-2)\cdot 0 - (-1)^2 \cdot (-1) = 1 \end{array}$$

$\Rightarrow$ *2 Vorzeichenwechsel* $\Rightarrow$ *2 Eigenwerte von* A *kleiner als 2.*
In der Tat ist $\Lambda(A) \approx \{0.254, 1.823, 3.177, 4.745\}$.

Mit Hilfe des Satzes von Gershgorin:

$$\Lambda(A) \subset \bigcup_{i=1}^{n} [a_{ii} - r_i, a_{ii} + r_i] \text{ mit } r_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|)$$

angewendet auf A tridiagonal wie in (III.1) erhalten wir eine einfache Methode y und z so zu bestimmen, dass $\Lambda(A) \subset [y, z]$:
Wähle $y = \min\left(\{a_1 - |b_1|, a_n - |b_{n-1}|\} \cup \{a_i - |b_i| - |b_{i-1}| : 1 < i < n\}\right)$, sowie
$z = \max\left(\{a_1 + |b_1|, a_n + |b_{n-1}|\} \cup \{a_i + |b_i| + |b_{i-1}| : 1 < i < n\}\right)$.

III.4 Divide-and-Conquer Algorithmus

Der (schnellste) Algorithmus zur Berechnung aller Eigenwerte und Eigenvektoren einer tridiagonalen, symmetrischen Matrix $A \in \mathbb{R}^{n \times n}$ wie in (III.1) für $n > 25$ ist derzeit der Divide-and-Conquer Algorithmus. Wir möchten dabei wieder $Q^T A Q = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ mit $Q^T Q = I$ berechnen. Dabei wollen wir das Problem rekursiv in Teilprobleme für Matrizen $T_1 \mathbb{R}^{m \times m}$ und $T_2 \in \mathbb{R}^{n-m \times n-m}$ zerlegen, bis $m = 1$ und damit das Eigenwertproblem trivial ist.

$$A = \left[\begin{array}{cccc|cccc}
a_1 & b_1 & & & & & & \\
b_1 & \ddots & \ddots & & & & & \\
& \ddots & \ddots & b_{m-1} & & & & \\
& & b_{m-1} & a_m & b_m & & & \\
\hline
& & & b_m & a_{m+1} & b_{m+1} & & \\
& & & & b_{m+1} & \ddots & \ddots & \\
& & & & & \ddots & \ddots & b_{n-1} \\
& & & & & & b_{n-1} & a_n
\end{array}\right] \text{(unreduziert)}$$

$$= \left[\begin{array}{cccc|cccc}
a_1 & b_1 & & & & & & \\
b_1 & \ddots & \ddots & & & & & \\
& \ddots & \ddots & b_{m-1} & & & & \\
& & b_{m-1} & a_m - b_m & & & & \\
\hline
& & & & a_{m+1} - b_m & b_{m+1} & & \\
& & & & b_{m+1} & \ddots & \ddots & \\
& & & & & \ddots & \ddots & b_{n-1} \\
& & & & & & b_{n-1} & a_n
\end{array}\right] + \left[\begin{array}{c|c}
b_m & b_m \\
\hline
b_m & b_m
\end{array}\right]$$

$$=: \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \underbrace{\begin{bmatrix} 0 & \cdots & 0 & 1 & 1 & 0 & \cdots & 0 \end{bmatrix}}_{=:v^T}$$

$$= \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m v v^T$$

Angenommen, $Q_i^T T_i Q_i = D_i = \begin{bmatrix} \diagdown \end{bmatrix}$ sind die Eigenwertzerlegungen der T_i, dann

$$\begin{aligned} \Rightarrow A &= \begin{bmatrix} Q_1 D_1 Q_1^T & \\ & Q_2 D_2 Q_2^T \end{bmatrix} + b_m v v^T \\ &= \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix} \left(\begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} + b_m u u^T \right) \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}^T \end{aligned}$$

Dabei ist

$$u := \begin{bmatrix} Q_1^T & \\ & Q_2^T \end{bmatrix} v = \begin{bmatrix} Q_1^T(:,n) \\ Q_2^T(:,1) \end{bmatrix} = \begin{bmatrix} Q_1(n,:) \\ Q_2(1,:) \end{bmatrix}$$

und es gilt

$$\Lambda(A) = \Lambda(D + \varrho u u^T), \text{ mit } D = \begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix} \text{ und } \varrho = b_m$$

Wir wenden das Verfahren rekursiv an, bis $T_1, T_2 \in \mathbb{R}$. Dabei müssen wir die Eigenwerte von $D + \varrho u u^T$ berechnen. $\Rightarrow$ Benötigen billige Berechnung von Spektra für Diagonalmatrizen mit Rang-1-Aufdatierungen. Dazu nehmen wir o.B.d.A an, dass $d_1 > d_2 > \cdots > d_n$ und setzen $\lambda \neq d_j \forall j$. Dann folgt:

$$\det(D + \varrho u u^T - \lambda I) = \det\big((D - \lambda I)(I + \varrho (D - \lambda I)^{-1} u u^T)\big)$$

und wir sehen, dass

$$\det(I + \varrho (D - \lambda I)^{-1} u u^T) = 0 \Leftrightarrow \lambda \in \Lambda(A) \tag{III.2}$$

Zur Bestimmung der Determinante müssen wir Determinanten der Form $\det(I + x y^T)$ für $x, y \in \mathbb{R}^n$ berechnen. Dabei hilft uns das folgende Lemma.

Lemma III.9. *Seien $x, y \in \mathbb{R}^n$. Dann gilt:*

$$\det(I + x y^T) = 1 + y^T x$$

.

Beweis. Sei $\{\lambda_1, \dots, \lambda_r\} = \Lambda(I + xy^T)$; Dann ist

$$\lambda_j(I + xy^T) = \lambda_j(xy^T) + 1$$

Andererseits hat xy^T Rang 1 und es gilt $\Lambda(xy^T) = \{y^T x, 0, \dots, 0\}$, denn

$$(xy^T)x = x\underbrace{(y^T x)}_{\in\mathbb{R}} = (y^T x)x \Rightarrow x \text{ ist Eigenvektor zum Eigenwert } y^T x$$

Es folgt also $\Lambda(I + xy^T) = \{y^T x, 1, \dots, 1\}$ und damit $\det(I + xy^T) = 1 + y^T x$ □

Definiere $f(\lambda) := 1 + \varrho \sum\limits_{i=1}^{n} \frac{u_i^2}{d_i - \lambda}$. Es gilt

$$f(\lambda) = 1 + \varrho[u_1, \dots, u_n] \begin{bmatrix} \frac{1}{d_1-\lambda} u_1 \\ \vdots \\ \frac{1}{d_n-\lambda} u_n \end{bmatrix} = 1 + \underbrace{\varrho u^T}_{=:y^T} \underbrace{(D - \lambda I)^{-1} u}_{=:x}$$
$$\overset{\text{Lemma III.9}}{=} \det\left(I + \varrho\,(D - \lambda I)^{-1} uu^T\right),$$

d.h. $f(\lambda) = 0 \Leftrightarrow \lambda \in \Lambda(A)$.

Wir suchen also die Nullstellen von f. Falls $u_i \neq 0$ (sonst ist nichts zu tun)und $\varrho > 0$, dann ist wegen $\lambda \neq d_i$

$$f'(\lambda) = \varrho \sum_{i=1}^{n} \frac{u_i^2}{(d_i - \lambda)^2} > 0.$$

Nach Definition gilt damit:

$$\lim_{\lambda \to -\infty} f(\lambda) = 1 \qquad \text{und} \qquad \lim_{\lambda \to \infty} f(\lambda) = 1$$

außerdem

$$\lambda < d_m \Rightarrow f(\lambda) > 1 \qquad \text{und} \qquad \lambda > d_1 \Rightarrow f(\lambda) > 1$$

Siehe etwa Abbildung III.1 für ein Beispiel mit $n = 4$.

Insgesamt erhalten wir damit:

- In jedem Intervall (d_{j+1}, d_j), $j = n-1, \dots, 1$ liegt genau eine Nullstelle von f.
- falls $\varrho > 0$ liegt eine weitere Nullstelle rechts von d_1 , für $\varrho < 0$ liegt eine weitere Nullstelle links von d_n.

Wir setzen daher ein Newton–Verfahren zum Berechnen der Nullstellen $\hat{=}$ Eigenwerte auf jedem Intervall an. Auswertungen von $f(\lambda)$ und $f'(\lambda)$ kosten $\mathcal{O}(n)$ `flops`. Damit liegt der Aufwand zur Berechnung aller Eigenwerte bei $\mathcal{O}(n^2)$. Falls nur die Eigenwerte gesucht sind ist dieses Verfahren also teurer als der symmetrische QR–Algorithmus! Allerdings kann der zu $\lambda \in \Lambda(A)$ gehörige Eigenvektor günstig wie folgt bestimmt werden:

Lemma III.10. $\lambda \in \Lambda(D + \varrho uu^T) \Rightarrow (D - \lambda I)^{-1} u$ *ist der zugehörige Eigenvektor.*

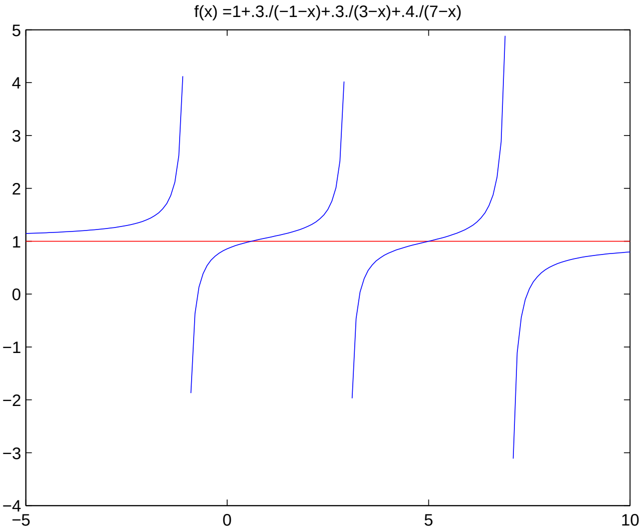

Abbildung III.1: Beispielgraph für $n = 4$

Beweis.

$$\begin{aligned}(D + \varrho uu^T)(D - \lambda I)^{-1}u &= (D - \lambda I + \lambda I + \varrho uu^T)(D - \lambda I)^{-1}u \\ &= u + \lambda(D - \lambda I)^{-1}u + u[\underbrace{\varrho u^T(D - \lambda I)^{-1}u}_{=\underbrace{f(\lambda)}_{=0}-1}] \\ &= \lambda(D - \lambda I)^{-1}u\end{aligned}$$

□

Die Kosten zur Berechnung aller Eigenvektoren belaufen sich damit also auf $2n^2$.

Wir haben bisher vorausgesetzt, dass $d_i \neq d_{i+1}$ und $u_i \neq 0$ gilt. Im Fall $u_i = 0$ folgt aber sofort $d_i \in \Lambda(D + \varrho uu^T)$, denn $(D + \varrho uu^T)e_i = d_i e_i$. Im Fall $d_i = d_{i+1}$ definiere $U = G(i, i+1, \theta)$ so, dass

$$G^T(i, i+1, \theta)u = \begin{bmatrix} u_1 \\ \vdots \\ u_{i-1} \\ \tilde{u}_i \\ 0 \\ u_{i+2} \\ \vdots \\ u_n \end{bmatrix}.$$

$$\Rightarrow U^T(D+\varrho uu^T)U = D + \varrho\tilde{u}\tilde{u}^T \text{ mit } \tilde{u}_{i+1} = 0$$

$$\Rightarrow Ue_{i+1} \text{ ist Eigenvektor von } D+\varrho uu^T \text{ zum Eigenwert } d_{i+1}$$

In beiden Fällen kann d_i abgespalten werden und wir fahren dann mit dem verkleinerten Problem fort. Der Divide-and-Conquer Algorithmus besteht nun darin, die Teilungsprozedur rekursiv anzuwenden bis $T_1, T_2 \in \mathbb{R}$ gilt.

Bemerkung. *Die Berechnung der Eigenvektoren mit Lemma III.10 ist numerisch instabil, falls $\lambda \approx d_i$. Ein numerisch stabiler Algorithmus nach [GE94] verwendet den*

Satz III.11 (von Löwner)**.** *Sei* $D = \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{pmatrix}$ *und* $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ *so, dass* $d_n < \alpha_n < d_{n-1} < \cdots < d_1 < \alpha_1$. *Mit*

$$\hat{u}_i = \pm \left[\frac{\prod\limits_{j=1}^{n} (\alpha_j - d_j)}{\prod\limits_{\substack{j=1 \\ j\neq i}}^{n} (d_j - d_i)} \right]^{\frac{1}{2}}, \qquad \hat{u} = \begin{bmatrix} \hat{u}_1 \\ \vdots \\ \hat{u}_n \end{bmatrix}$$

gilt:

$$\Lambda\left(D + \hat{u}\hat{u}^T\right) = \{\alpha_1, \ldots, \alpha_n\}.$$

Wir berechnen also die Eigenwerte wie zuvor, wenden aber Lemma III.10 auf $D+\hat{u}\hat{u}^T$ an. Diese Lösung ist dann numerisch stabil, da $(D-\lambda_i I)^{-1}\hat{u}$ orthogonaler ist, als $(D-\lambda_i I)^{-1}u$

Beweis zum Satz von Löwner. Setze $\hat{D} = D + \hat{u}\hat{u}^T$.

$$\Rightarrow p_{\hat{D}}(\lambda) = \det(\hat{D} - \lambda I) =: \prod_{j=1}^{n} (\alpha_j - \lambda)$$

Andererseits gilt unter Verwendung von Lemma III.9:

$$\begin{aligned}
\det(\hat{D} - \lambda I) &= \det(D-\lambda I)\det(I + (D-\lambda I)^{-1}\hat{u}\hat{u}^T) \\
&= \prod_{j=1}^{n}(d_j - \lambda)\left(1 + \sum_{j=1}^{n} \frac{\hat{u}_j^2}{d_j - \lambda}\right) \\
&= \prod_{j=1}^{n}(d_j - \lambda)\left(1 + \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{\hat{u}_j^2}{d_j - \lambda}\right) + \prod_{\substack{j=1 \\ j\neq i}}^{n}(d_j - \lambda)\hat{u}_i^2.
\end{aligned}$$

Setzt man nun $\lambda = d_i$, dann folgt

$$\begin{aligned}
\prod_{j=1}^{n} (\alpha_j - d_j) &= \underbrace{\prod_{j=1}^{n} (d_j - d_i)}_{=0} \left(1 + \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{\hat{u}_i^2}{d_j - d_i} \right) + \prod_{\substack{j=1 \\ j \neq i}}^{n} (d_j - d_i) \hat{u}_i^2 \\
&= \prod_{\substack{j=1 \\ j \neq i}}^{n} (d_j - d_i) \hat{u}_i^2
\end{aligned}$$

und damit:

$$\hat{u}_i^2 = \frac{\prod\limits_{\substack{j=1 \\ j \neq i}}^{n} (\alpha_j - d_i) \hat{u}_i^2}{\prod\limits_{\substack{j=1 \\ j \neq i}}^{n} (d_j - d_i) \hat{u}_i^2} > 0.$$

Der Quotient ist positiv, da sowohl im Zähler als auch im Nenner genau $n - i$ negative Faktoren auftreten. $\square$

Algorithmus III.7 Divide-and-Conquer (`dcsep`)

Input: $A = A^T \in \mathbb{R}^{n\times n}$ tridiagonal

Output: $\Lambda := \{\lambda_1, \dots, \lambda_n\} = \Lambda(A), Q = [q_1, \dots, q_n] \in \mathbb{R}^{n\times n}$ orthogonal mit $Aq_j = \lambda_j q_j,\ j = 1, \dots, n$

Bestimme n.
if $n > 1$ **then**
  Forme $A = \begin{bmatrix} A_1 & \\ & A_2 \end{bmatrix} + \varrho v v^T$ $\{$mit $\varrho = b_m\}$
  Berechne Q_1, Λ_1 durch Aufruf von `dcsep`(A_1)
  Berechne Q_2, Λ_2 durch Aufruf von `dcsep`(A_2)
  Berechne $D + \varrho u u^T$ aus $Q_1, Q_2, \Lambda_1, \Lambda_2$.
  Finde die Eigenwerte Λ von $D + \varrho u u^T$ durch Berechnen aller Nullstellen von

$$f(\lambda) = 1 + \varrho \sum_{i=1}^{n} \frac{u_i^2}{d_i - \lambda}.$$

  Berechne den Vektor $\hat{u}$ mit dem Satz von Löwner und den Eigenvektor von $D + \hat{u}\hat{u}^T$:

$$\hat{q}_j = (D - \lambda_j)^{-1}\,\hat{u}$$

  Berechne die Eigenvektoren von A: $Q = \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix} [\hat{q}_1, \dots, \hat{q}_n]$
else
  $Q = 1, \Lambda = A$
end if

Die Kosten im Algorithmus III.7 belaufen sich auf:

$$\begin{aligned}
t(n) &= 2\left(\frac{n}{2}\right) + \overset{\text{Eigenwert}}{\mathcal{O}(n^2)} + \overset{\text{Eigenvektor}}{\mathcal{O}(n^2)} + \overset{\text{Aufdatierung } Q}{cn^3} \\
&\approx 2t\left(\frac{n}{2}\right) + cn^3 \\
&= 2\left(2t\left(\frac{n}{4}\right) + c\left(\frac{n}{2}\right)^3\right) + cn^3 = 4t\left(\frac{n}{4}\right) + c\frac{n^3}{4} + cn^3 \\
&= 2^{\log_2(n)} t(1) + \sum_{j=0}^{\log_2(n)} cn^3 \cdot \frac{1}{4j} \\
&= n \cdot 0 + \sum_{j=0}^{\log_2(n)} cn^3 \cdot \frac{1}{4^j} = cn^3 \sum_{j=0}^{\log_2(n)} \cdot \left(\frac{1}{4}\right)^j \\
&= \frac{1 - \left(\frac{1}{4}\right)^{\log_2(n)+1}}{1 - \frac{1}{4}} cn^3 \approx \frac{4}{3} cn^3
\end{aligned}$$

Literaturverzeichnis

[GE94] Ming Gu and Stanley C. Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM J. Matrix Anal. Appl.*, 15(4):1266–1276, 1994. 7

[GV96] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition, 1996. 2