Übung 1
Computational Physics III

Matthias Plock (552335) Paul Ledwon ()

3. Mai 2018

Inhaltsverzeichnis

1 Aufgabe 1

1.1 Einleitung

Es soll das lineare Gleichungssystem

$$Ax = b$$

gelöst werden. A ist eine quadratische, symmetrische und dünn besetzte $N \times N$ Matrix, x und b sind N-dimensionale Vektoren. Für das Lösen soll die Methode der konjugierten Gradienten (conjugate gradient) verwendet werden. Diese wurde der Vorlesung entsprechend implementiert. Im weiteren Verlauf wird der Fehler $e_k = x - x_k$ benötigt. Der Vorlesung folgend kann man ihn aus dem Residuum r_k berechnen. Es gilt

$$r_k = b - Ax_k = Ax - Ax_k = A(x - x_k) = Ae_k \,.$$

Bei bekanntem Residuum r_k lässt sich der Fehler also durch invertieren der Matrix A bestimmen,

$$r_k = Ae_k \qquad \Longrightarrow \qquad e_k = A^{-1} r_k \,.$$

1.2 Beschreibung des Programms

Die Implementierung wurde in Python durchgeführt. Der Einstiegspunkt des Programms liegt in der Datei `bin/conjugate_gradient.py`, d.h. Ausführung dieser Datei führt das Programm aus. Die Programmdateien liegen im Verzeichnis `bin/`. Dort befinden sich drei Dateien:

- `cg.py` Die Datei enthält die Funktionen `cg(A, b)` (Durchführung des conjugate gradient Verfahrens) und `two_norm(x)` (Berechnung der Zwei-Norm für den Vektor `x`). Auf die Durchführung des Verfahrens wird später genauer eingegangen.
- `laplace.py` Die Datei `laplace.m` wurde nach Python portiert. Sie enthält die Funktion `laplace(N)`. Diese hat als Rückgabewert den diskretisierten Laplace Operator mit Dimensionen $N^2 \times N^2$.
- `conj_grad.py` Diese Datei enthält den Einstiegspunkt für das Programm, d.h. durch Ausführung dieses Programms werden u.A. alle Abbildungen generiert.

1.3 Ablauf des Programms

Durch Aufrufen der Datei `conj_grad.py` wird das Programm gestartet. Im unteren Abschnitt befindet sich eine `if`-Abfrage, (`if __name__ == '__main__'`), welche nach `True` evaluiert, wenn die Datei von der Shell aus aufgerufen wird. Wir rufen die importierte Funktion `laplace(N` auf und weisen `A` einen diskretisierten Laplace Operator der Dimension 64×64 zu. Für Aufgabenteil (b) speichern wir nun einige rechte Seiten in unterschiedlichen Variablen:

- in der Variable `A_smallest_eigvec` speichern den Eigenvektor, der zum kleinsten Eigenwert gehört,
- in `unit_vector` setzen wir alle Einträge auf 0, einen zufälligen Eintrag setzen wir dann auf 1,
- in `ones_vector` speichern wir das Resultat von Au mit $u = (1, 1, \ldots)$,
- und in `sum_of_eigvecs` addieren wir zwei zufällige Eigenvektoren zusammen.

Für jeden der oben definierten rechten Seiten sowie für einen Vektor, der sich aus zufälligen Einträgen zusammensetzt, wird nun der Lösungsvektor x mit Hilfe der Funktion `cg(A, b)` bestimmt. Die Methode gibt außerdem noch das Residuum für jeden Iterationsschritt zurück. Aus diesem Residuum wird nun mit Hilfe der Matrix A der Fehler berechnet, anschließend werden die Normen von Residuum und Fehler zusammen in einer Abbildung geplottet.

1.4 Implementierung der Methode

Es soll hier kurz auf die beiden Abbruchskriterien eingegangen für die conjugate gradient Methode werden.

- Endliche Iterationsschritte Bei einer Problemgröße von $N \times N$ (Lösungsvektor also N-Dimensional) erwarten wir, dass Konvergenz nach maximal N Schritten vorliegt. Das liegt daran, dass der Lösungsvektor in jedem Schritt um eine weitere Dimension aufgespannt wird. Da der Lösungsvektor aber N-dimensional ist, kann man nach N Schritten keine weiteren Dimensionen mehr aufspannen, das Ergebnis sollte konvergiert sein. Erstes Abbruchkriterium ist die endliche Anzahl der Iterationsschritte.
- Abbruch bei Erreichen der Genauigkeit Das zweite Abbruchkriterium ist eine Überprüfung der 2-Norm des Residuums, das in jedem Iterationsschritt berechnet wird. Im k-ten Schritt wird das Residuums r_{k+1} berechnet. Ist $\|r_{k+1}\|_2$ kleiner als eine vorgegebene Toleranz δ, so wird die Iteration (konvergiert) abgebrochen und der bestimmte Lösungsvektor x sowie alle bisher bestimmten Residuuen zurück gegeben. Als Toleranz δ setzen wir hier $\delta = N\varepsilon$ mit Problemgröße N und Maschinengenauigkeit ε.

1.5 Resultate

Bestimmt werden sollten die Normen des Residuums $\|r_k\|$ und des Fehlers $\|e_k\|$, auch in Abhängigkeit der rechten Seite b. Abgesehen von Abb. 4 konvergieren die Methoden für alle rechten Seiten nach etwas mehr als 30 Schritten. Verwendet man $b = Au$ mit $u = (1, 1, \ldots)$, so konvergiert die Methode bereits nach 10 Schritten.

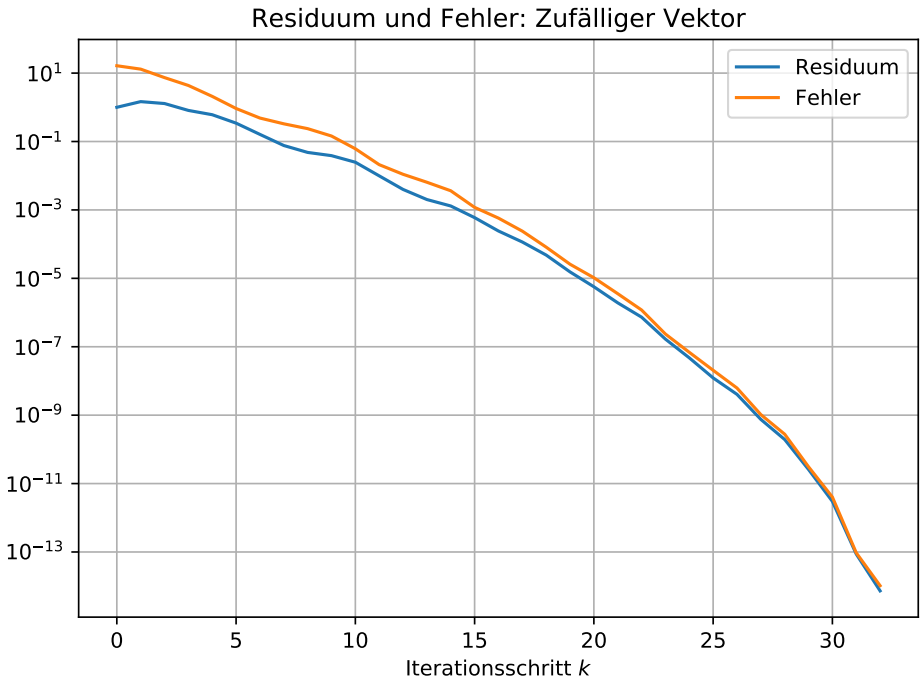

Abbildung 1: Residuum und Fehler für einen zufälligen Vektor. Nach 32 Schritten ist die festgelegte Genauigkeit erreicht.

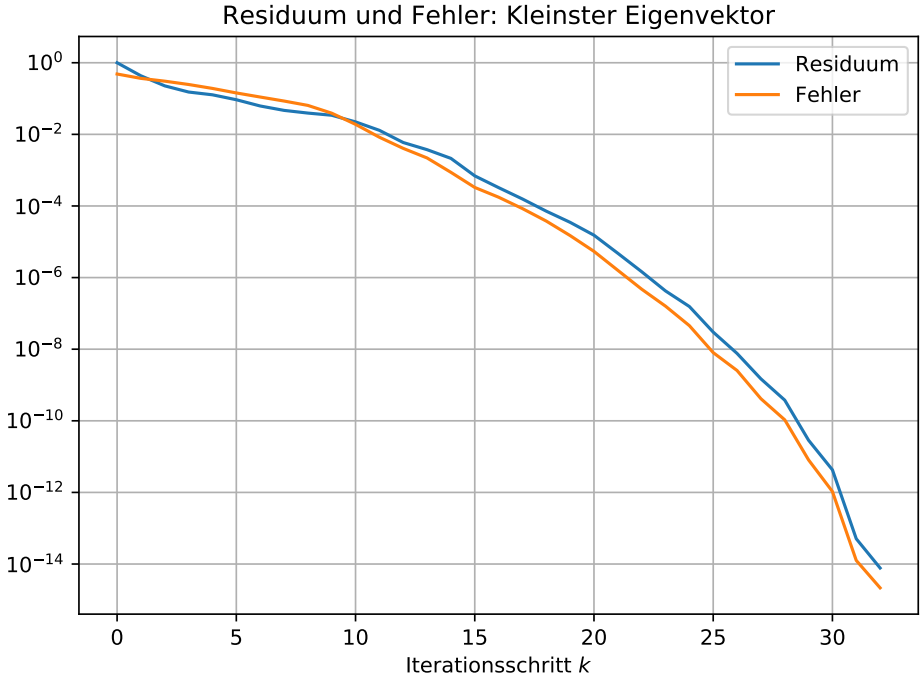

Abbildung 2: Residuum und Fehler für den kleinsten Eigenvektor. Nach 33 Schritten ist die festgelegte Genauigkeit erreicht.

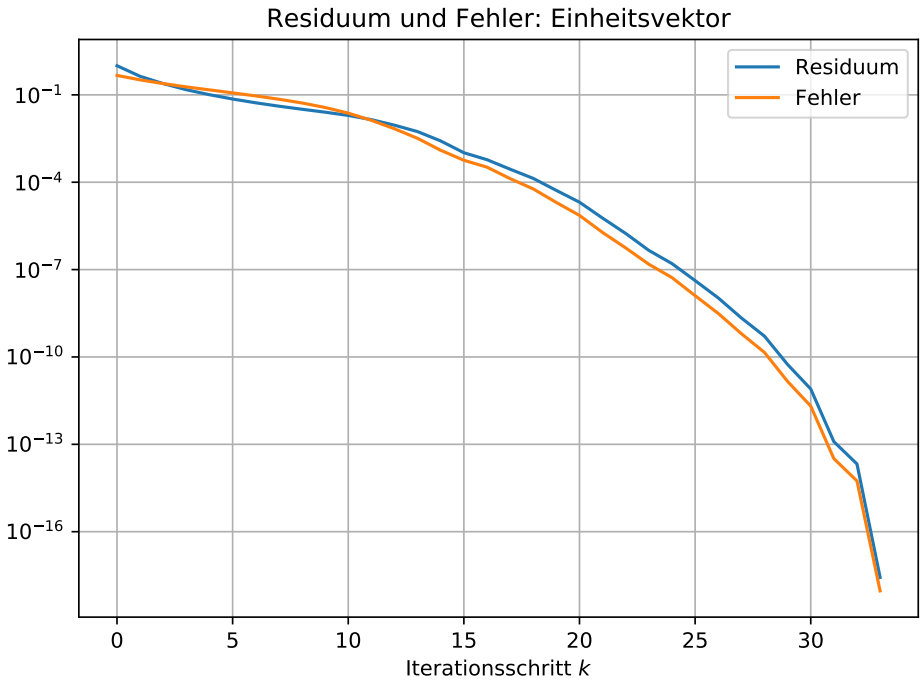

Abbildung 3: Residuum und Fehler für den Einheitsvektor. Nach 33 Schritten ist die festgelegte Genauigkeit erreicht.

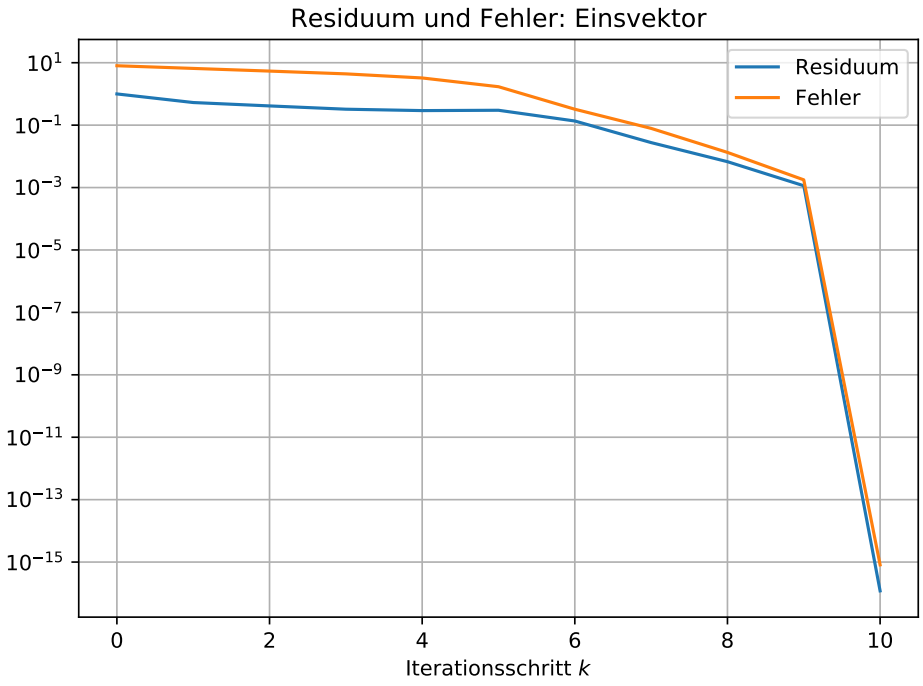

Abbildung 4: Residuum und Fehler für $b = Au$ mit $u = (1, 1, \dots)$. Nach 10 Schritten ist die festgelegte Genauigkeit erreicht. Bis zum 9ten Schritt verhält sich die Konvergenz identisch zu den anderen rechten Seiten, im 10ten Schritt wird die festgelegte Genauigkeit dann abrupt erreicht.

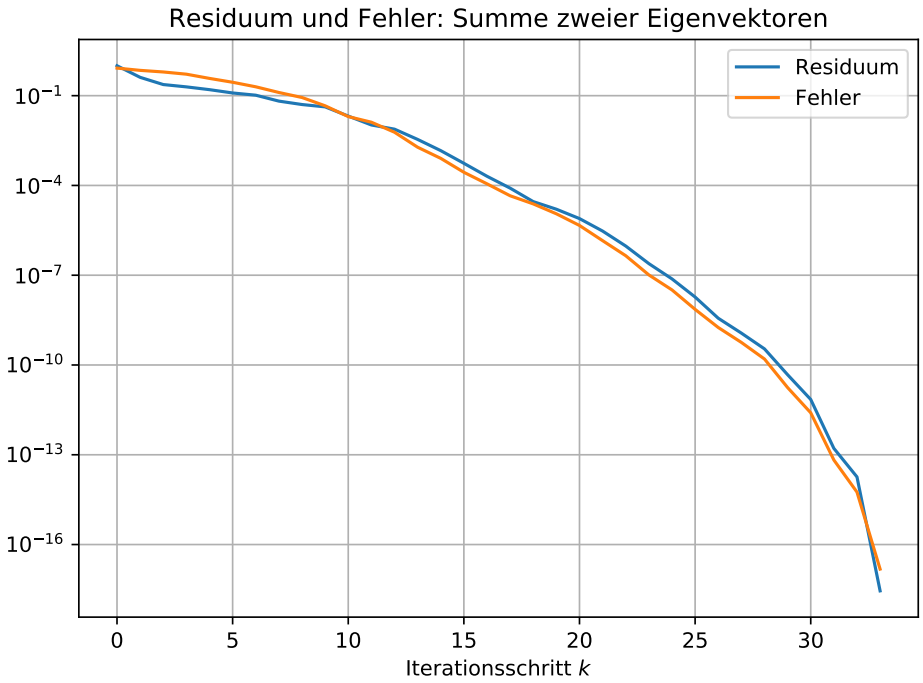

Abbildung 5: Residuum und Fehler für die Summe zweier Eigenvektoren. Nach 33 Schritten ist die festgelegte Genauigkeit erreicht.

2 Aufgabe 2