Prof. Dr. A.K. Hartmann WiSe 2021/22, 5.04.4521 und 5.04.4524
Blatt 1, 11.11.2021

Übungen zur Computerorientierten Physik

1 Eigenwert

Seit $\mathbf{A} = (a_{ij})$ eine invertierbare (hier symmetrische) $n \times n$ Matrix mit Eigenwerten λ_i und Eigenvektoren $\vec{v}_i$, also

$$\mathbf{A}\vec{v}_i = \lambda_i \vec{v}_i \quad (i = 1, \dots, n).$$

Die Eigenwerte seien dem Betrag nach fallend geordnet und als verschieden angenommen, also

$$|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|.$$

Weiterhin wird o.B.d.A. angenommen, dass die Eigenvektoren eine Orthonormalbasis bilden, sich also jeder Vektor als Linearkombination $\sum_i c_i \vec{v}_i$ der Basis schreiben lässt.

Hier sollen Sie einen einfachen iterativen Algorithmus zur Bestimmung des betragsmäßig größten Eigenwertes λ_1 programmieren. Dazu starten Sie mit einem beliebigen (zufälligen) Vektor $\vec{b}^{(0)}$, der normiert sei (also $||\vec{b}^{(0)}|| = \sqrt{\vec{b}^{(0)} \cdot \vec{b}^{(0)}} = \sqrt{\sum_i (b_i^{(0)})^2}$) und berechnen iterativ $(k = 0, 1, 2, \dots)$:

$$\vec{b}^{(k+1)} = \frac{\mathbf{A}\vec{b}^{(k)}}{||\mathbf{A}\vec{b}^{(k)}||} \tag{1}$$

$$\mu_{k+1} = \vec{b}^{(k)} \cdot \mathbf{A}\vec{b}^{(k)} \tag{2}$$

Damit gilt $\vec{b}^{(k)} = \frac{\mathbf{A}^k \vec{b}^{(0)}}{||\mathbf{A}^k \vec{b}^{(0)}||}$. Wir können $\vec{b}^{(0)}$ als Linearkombination der Basis schreiben $\vec{b}^{(0)} = \sum_i c_i \vec{v}_i$. Also gilt:

$$\mathbf{A}^k \vec{b}^{(0)} = \mathbf{A}^k (\sum_i c_i \vec{v}_i) = \sum_i c_i \lambda_i^k \vec{v}_i = \lambda_1^k \left(\sum_i c_i \left(\frac{\lambda_i}{\lambda_1} \right)^k \vec{v}_i \right) .$$

Unter der Annahme $c_1 \neq 0$, wenn also $\vec{b}^{(0)}$ irgendeinen Anteil in Richtung von $\vec{v}_1$ enthält, und da $|\frac{\lambda_i}{\lambda_1}| < 1$ für $i > 1$, konvertiert die Summe gegen $c_1 \vec{v}_1$ und somit $\vec{b}^{(k)}$ gegen $\vec{v}_1$. Daraus folgt, dass $\mu_k \to \vec{c}_1 \cdot \lambda_1 \vec{v}_1 = \lambda_1$ konvergiert.

1. Laden Sie das Programfragment `eigenvalue_fragment.c` vom StuIP. Schauen Sie sich das Programm an. Das soweit vorhandene Programm erzeugt eine symmetrische $n \times n$ Zufallsmatrix `matrix[][]` (Diagonalwerte 1, ander Einträge aus $[-1, 1]$, Hinweis: Die Funktion `drand48()` liefert eine (Pseudo-) Zufallszahl die gleichverteilt in $[0, 1]$ liegt.)

Als nächstes wird ein Zufallsvektor `vec[]` erzeugt (entspricht $\vec{b}^{(0)}$). Es folgt eine (unvollständige) Schleife in der (1) und (2) iteriert werden.

Das Programm kann mit `cc -o eigenvalue eigenvalue_fragment.c -lm -Wall` kompiliert werden (Eingabe in der Shell, in dem Verzeichnis in dem das Programm gespeichert ist).

2. Ergänzen sie die Schleife um

   - die Berechnung der Länge `norm` von `vec[]` und um eine anschließende Nomierung von `vec[]` auf 1.
   - die Multiplikation von `matrix[][]` mit `vec[]`. Speichern Sie das Ergebnis in `vec2[]`.

   (Die Schleife enthält schon die Berechnung von `mu` (entspricht μ_k), eine Kontroll-Ausgabe sowie die Umgkopierung von `vec2[]` nach `vec[]`)

3. Compilieren und testen Sie das Programm geeignet, z.B. indem Sie für kleine Werte von `n` geeignete (einfache) Maxtrizen und Vektoren vorgeben und das Ergebnis verifzieren.

4. Leiten Sie die Ausgabe der Iteration in eine Datei, z.B. mit

   ```
   eigenvalue > konvergenz.dat
   ```

   Plotten Sie die beiden Schätzer `vec2[0]/vec[0]` (3. Spalte in `konvergenz.dat`) und `mu` (4. Spalte) (z.B. mit `gnuplot`) als Funktion des Scheifenzählers `iter` (1. Spalte). Welcher Schätzer konvergiert schneller? Probieren Sie verschiedene Matrizen (also unterschiedliche Werte für `seed`). Wählen Sie eine geeignet Iterationszahl `max_iter`, so dass Sie einigermaßen sicher sein können, dass der größte Eigenwert konvergiert ist.

↑ ——————— Minimalziel ——————— ↑

5. Umgeben Sie den Haupteil des Programms mit einer Schleife, so dass der größte Eigenwert für (z.B.) 10000 unterschiedliche Zufallsmatrizen berechnet wird. Die Werte werden schon im Code in ein Histogramm eingetragen. Geben Sie das Histogramm aus (Code bei `/* print histogram */` ent-kommentieren und die Ausgabe in der Schleife durch einen `/* ... */` Kommentar umgeben).

   Leiten Sie die Ausgabe des Histogramms wieder in eine Datei und plotten Sie das Histogramm? Was erhalten Sie?

6. Ändern Sie die erzeugten Matrizen, so dass auf der Diagonale einer 0 steht. Wie sieht das erzeugte Histogramm jetzt aus. Was passiert wenn auf der Diagonale eine 0.5 steht?

7. (Zusatzaufgabe) Wie könnte der Algorithmus erweitert werden, um auch den zweitgrößten Eigenwert zu bestimmen (daraus folgt dann auch, wie alle Eigenwerte sukzessive bestimmt werden können).