Praktikum Wissenschaftliches Rechnen

Eigenwerte & -vektoren

Sebastian Schöps, Timon Seibel

TECHNISCHE UNIVERSITÄT DARMSTADT

Sommersemester 2025
07. Juli 2025
Übungsblatt 6

Ziel dieser Übung ist es, die Vektoriteration in `Python` zu implementieren. Hierzu wird Ihnen auf `Moodle` das Grundgerüst des abzugebenden `Python`-Projekts zur Verfügung gestellt. Ihre Aufgabe ist es, den Code entsprechend der Aufgabenstellung fertigzustellen.
Sie dürfen weitere (Hilfs-)Methoden hinzufügen, um Ihren Code übersichtlicher zu gestalten. Aus der Sicht des Software Engineerings wird dies sogar empfohlen. Kommentieren Sie Ihren Code, sodass er für Dritte verständlich ist.

Wichtig: Die Übung ist **in Gruppen** zu bearbeiten und die vorgegebenen (noch zu vervollständigenden) Methoden sind zwingend zu verwenden. Insbesondere dürfen weder die Namen der vorgegebenen Methoden noch die Reihenfolge ihrer Argumente abgeändert werden. Ebenso dürfen Sie die Rückgabewerte nicht ändern. Andernfalls könnten die zugehörigen Tests fehlschlagen, was zu Punktverlust führt.
Weiterhin möchten wir Sie daran erinnern, dass Sie, mit Ausnahme von `numpy` und `matplotlib`, keine externen Bibliotheken benutzen dürfen, sofern dies die Aufgabenstellung nicht ausdrücklich erlaubt.

Zur Bearbeitung der Übung stellen wir Ihnen die folgenden Dateien zur Verfügung:

ev.py Grundgerüst der Implementierung der Vektoriteration

main.py Von hieraus sollen die geforderten Plots **automatisiert** erstellt und gespeichert werden

beobachtungen.txt Hier sollen sie Ihre Beobachtungen schriftlich festhalten

Abgabe: Komprimieren Sie alle relevanten und Verzeichnisse in `gruppe_xy_uebung_6.zip` (oder `.tar.gz`), wobei `xy` durch Ihre Gruppennummer zu ersetzen ist, und reichen Sie die Archiv-Datei bis **spätestens 20.07.2025, 23:59 Uhr** auf `Moodle` ein. Spätere Abgaben können nicht berücksichtigt werden. Reichen Sie Ihre Abgabe daher frühzeitig ein und melden Sie sich bei Problemen mit dem Upload vor Ablauf der Deadline, damit wir gegebenenfalls eine Lösung finden können.

Aufgabe 6.1: Evaluation

Im Zuge der sechsten - und damit letzten - Übung möchten wir die Lehrveranstaltungsevaluation durchführen. Da es sich bei dem Praktikum um eine komplett neue Veranstaltung handelt, ist Ihr Feedback für uns sehr wichtig. Insbesondere Ihre Anmerkungen in den Freitextfeldern - egal ob positiv oder negativ - helfen uns, das Praktikum für die Zukunft zu verbessern. Daher möchten wir Sie bitten, in großer Zahl an der Evaluation teilzunehmen.
Die Teilnahme ist selbstverständlich wie immer **freiwillig** und **anonym**. Sie erhalten für die Teilnahme keine Punkte.

https://t1p.de/LVE
TAN/Losung: GQCPJ

Aufgabe 6.2: Vektor-Iteration (6 Punkte)

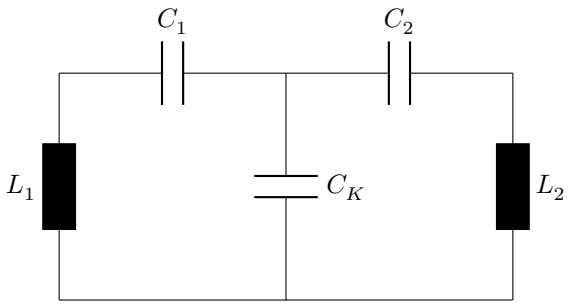
In dieser Aufgabe sollen Sie die Vektoriteration, auch *Potenzmethode* oder *power method* genannt, implementieren. Vervollständigen Sie hierzu die Methode `power_method`, welche eine **komplexwertige**[1] Matrix `A: np.ndarray[complex]` und einen **komplexwertigen** Startvektor `x0: np.ndarray[complex]` als Eingabewerte erwartet. Weiterhin soll die Methode als optionale Inputargumente die Toleranz `tol` mit Standardwert 10^{-6} und und die Höchstzahl an Iterationen `max_iter` mit Standardwert 100 übergeben bekommen können.
Beenden Sie die Berechnung sobald die Höchstzahl an Iterationen `max_iter` erreicht wurde, oder die Änderung des Eigenwertes zweier Iterationen kleiner als die Toleranz wird, sprich, $|\lambda^{(i)} - \lambda^{(i-1)}| <$`tol`.
Als Rückgabewert liefert die Methode ein Array mit den Approximationen $\lambda^{(0)}, \lambda^{(1)}, \dots, \lambda^{(k)}$ des betragsmäßig größten Eigenwerts sowie den zur letzten Approximation $\lambda^{(k)}$ gehörenden approximierten Eigenvektor $\mathbf{x}^{(k)}$. Denken Sie daran, dass auch Ihre Rückgabewerte komplexwertig sind.
Hinweis: Normieren Sie den Vektor in jeder Iteration mit der euklidischen Norm. Nutzen Sie den Rayleigh-Quotienten

$$\lambda^{(k)} = \frac{(\mathbf{x}^{(k)})^H A \mathbf{x}^{(k)}}{(\mathbf{x}^{(k)})^H \mathbf{x}^{(k)}}$$

zur Eigenwertapproximation. Verwenden Sie zur Berechnung des Skalarprodukts `numpy.vdot`[2], da dieses komplexe Zahlen korrekt behandelt (in einem komplexen Skalarprodukt geht das erste Argument komplex kunjugiert ein).

Aufgabe 6.3: Experiment (6 Punkte)

In dieser Aufgabe sollen Sie die von Ihnen implementierte Potenzmethode auf ein Praxisbeispiel anwenden. Wir betrachten hierzu zwei LC-Schwingkreise, welche über eine Kapazität C_K miteinander gekoppelt sind. Dies ergibt folgenden Schaltkreis:



Über die Kirchhoff'sche Maschenregel kommen wir nun zu folgendem gekoppelten Gleichungssystem:

$$L_1 \ddot{Q}_1 + \frac{1}{C_1} Q_1 + \frac{1}{C_K}(Q_1 - Q_2) = 0$$
$$L_2 \ddot{Q}_2 + \frac{1}{C_2} Q_2 + \frac{1}{C_K}(Q_2 - Q_1) = 0$$

Wir können dieses System äquivalent als Matrixgleichung $L\ddot{\mathbf{Q}} + C\mathbf{Q} = 0$ mit Induktivitätsmatrix

$$L = \begin{pmatrix} L_1 & 0 \\ 0 & L_2 \end{pmatrix}$$

[1] `https://python-basics-tutorial.readthedocs.io/de/latest/types/numbers/complex.html`
[2] `https://numpy.org/doc/2.2/reference/generated/numpy.vdot.html`

und Kapazitätsmatrix

$$C = \begin{pmatrix} \frac{1}{C_1} + \frac{1}{C_K} & -\frac{1}{C_K} \\ -\frac{1}{C_K} & \frac{1}{C_2} + \frac{1}{C_K} \end{pmatrix}$$

formulieren. Nun verwenden wir den Ansatz $\mathbf{Q}(t) = \mathbf{x}e^{\mathrm{j}\omega t}$ und erhalten hiermit das Eigenwertproblem

$$(C - \omega^2 L)\mathbf{x} = 0.$$

Damit wir nun die Potenzmethode auf dieses Problem anwenden können, müssen wir es noch in die Form

$$A\mathbf{x} = \omega^2\mathbf{x} = \lambda\mathbf{x}$$

bringen, wobei $A = L^{-1}C$.
Sei nun $L_1 = 10\,\mu\mathrm{H}$, $L_2 = 22\,\mu\mathrm{H}$, $C_1 = 100\,\mathrm{nF}$, $C_2 = 47\,\mathrm{nF}$ und $C_K = 10\,\mathrm{nF}$. Verwenden Sie `power_method` mit Startvektor `x0` $= (1, 0)^\top$ um die Approximationen an den betragsmäßig größten Eigenwert sowie den zugehörigen Eigenvektor zu berechnen. Lassen Sie $\lambda^{(k)}$ und $\mathbf{x}^{(k)}$ auf der Konsole ausgeben und halten Sie die Matrix A sowie das berechnete Eigenwert-Eigenvektor-Paar $(\lambda^{(k)}, \mathbf{x}^{(k)})$ in `beobachtungen.txt` schriftlich fest.
Erläuern Sie schriftlich in ein bis zwei Sätzen in `beobachtungen.txt` wie ω physikalisch zu interpretieren ist. Geben Sie den konkreten Wert an.

Aufgabe 6.4: Plots (5 Punkte)

Betrachten Sie nun die Matrix

$$A = \begin{pmatrix} 5 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 3 \end{pmatrix}.$$

In dieser Aufgabe sollen Sie das Verhalten der Potenzmethode für verschiedene Startvektoren betrachten. Wenden Sie hierfür die Potenzmethode auf A mit den Startwerten `x0` $= (1, -1, 1)^\top$ und `x0` $= (0, 0, 1)^\top$ an. Lassen Sie jeweils den approximierten Eigenwert $\lambda^{(k)}$ mit zugehörigem Eigenvektor auf der Konsole ausgeben und halten Sie diese zusätzlich in `beobachtungen.txt` schriftlich fest.
Berechnen Sie als Referenz λ_{ref} mit Hilfe von `numpy.linalg.eig`, halten Sie diesen schrifltich fest und plotten Sie für beide Startvektoren den Fehler$|\lambda^{(i)} - \lambda_{\text{ref}}|$, $i = 0, 1, \ldots, k$, über der Anzahl an Iterationen. Beide Kurven sollen im gleichen Plot gezeigt werden. Verwenden Sie eine semilogarithmische Darstellung mit logarithmischer y-Achse und lassen Sie den Plot als `Comparison_different_start_vectors.pdf` in den Ordner `Plots` exportieren. Denken Sie wie immer an Achsenbeschriftungen, einen Titel sowie eine Legende.
Was stellen Sie fest? Halten Sie ihre Beobachtungen schriftlich in ein bis zwei Sätzen in `beobachtungen.txt` fest.

Aufgabe 6.5: Komplexwertige Matrix (3 Punkte)

Abschließend sollen Sie das komplexwertige Eigenwertproblem $A\mathbf{x} = \lambda\mathbf{x}$ mit Matrix

$$A = \begin{pmatrix} 3 & 1 \\ 0 & 3 + 1\mathrm{j} \end{pmatrix}$$

betrachten. Berechnen Sie mit der Potenzmethode und Startvektor `x0` $= (0, 1)^\top$ die Approximation $\lambda^{(k)}$ an den betragsmäßig größten Eigenwert sowie den dazugehörigen approximierten Eigenvektor $\mathbf{x}^{(k)}$. Lassen Sie beides auf der Konsole ausgeben und halten Sie das Eigenwert-Eigenvektor-Paar zusätzlich in `beobachtungen.txt` schriftlich fest.

Aufgabe 6.6: Tests

Damit Sie selbst einen Eindruck davon erhalten können wie automatisierte Tests in `Python` aussehen, stellen wir Ihnen die Datei `test.py` zur Verfügung, welche bereits Beispieltests beinhaltet. Diese überprüfen lediglich, dass Ihre Implementierung den korrekten Input entgegennimmt und der Output das richtige Format hat. Gerne dürfen Sie für sich selbst weitere Tests ergänzen, um Ihre Implementierung hiermit zu überprüfen. Das wird aber nicht bewertet.