Numerische Methoden der Elektrotechnik

Ulf Schlichtmann

15. April 2015

Hausaufgabe 1 - Numerik in MATLAB

Abgabetermin: 03.05.2015

Aufgabe 1 - Numerische Phänomene

Die erste Aufgabe zeigt ein kurzes Beispiel für Probleme, die in der Numerik auftreten können.
Bearbeiten Sie die Aufgabe in der MATLAB-Datei `aufgabe1_1.m`.

1. Vervollständigen Sie die Funktion `rundungsfehler()`:

   Eingabewerte sind eine Zahl `x` sowie die Anzahl der Iterationen `n`
   Rückgabewert ist `f`

   Die Funktion besteht aus zwei Schleifen. In der ersten Schleife soll n-mal die Wurzel aus x gezogen, in der zweiten Schleife x wieder n-mal quadriert werden.

2. Vervollständigen Sie das Hauptprogramm:

   Werten Sie die Funktion `rundungsfehler()` mit $x = 100$ und $n = 60$ aus und speichern Sie das Ergebnis in der Variable `result`, die vom Hauptprogramm ausgegeben wird.

   Beschreiben und erklären Sie Ihre Beobachtungen.

   Ermitteln Sie den Rückgabewert der Funktion für n von 1 bis 60, speichern Sie die Werte im Feld `results` und stellen Sie sie graphisch dar.

   Skalieren Sie die Achsen sinnvoll und beschriften Sie diese.

Nützliche Befehle: `sqrt`

Aufgabe 2 - Maschinengenauigkeit

In dieser Aufgabe soll die Maschinengenauigkeit Ihres Rechners bestimmt werden. Unter der Maschinengenauigkeit wird das kleinste ϵ, für das gilt $1 + \epsilon > 1$, verstanden.
Bearbeiten Sie die Aufgabe in der MATLAB-Datei `aufgabe1_2.m`.

1. Implementieren Sie die Funktion `genauigkeit()`:

   Eingabewerte sind der Startwert `start` sowie die Genauigkeit `precision`, in der gerechnet werden soll (`1`: single precision, `2`: double precision)
   Rückgabewert ist die errechnete Maschinengenauigkeit `epsilon`

   Hinweis: Starten Sie mit $\epsilon = 2$ und halbieren Sie ϵ solange, bis $1 + \epsilon \leq 1$ gilt.

2. Speichern Sie die errechneten Genauigkeiten in den Variablen `epsilon_single` bzw. `epsilon_double`. Diese Werte sowie die Genauigkeit `eps`, mit der MATLAB arbeitet, werden vom Hauptprogramm ausgegeben.

Nützliche Befehle: `if`, `while`, `single`, `double`

Aufgabe 3 - Maschinengenauigkeit am Beispiel der Zahl *e*

Die Zahl $\boldsymbol{e}$ ist durch $\lim_{n\to\infty}(1+\frac{1}{n})^n$ definiert. In dieser Aufgabe sollen Sie die Approximation von e untersuchen.
Bearbeiten Sie die Aufgabe in der MATLAB-Datei `aufgabe1_3.m`.

Implementieren Sie das Hauptprogramm:

Approximieren Sie $\boldsymbol{e}$ für die 191 vorgegebenen Werte n zwischen 10 und 10^{20}, jeweils in single sowie double precision.

Berechnen Sie für beide die Abweichung zwischen dem Ergebnis, das MATLAB liefert und dem approximiertem Wert und speichern Sie den Fehler in den vorgegebenen Feldern `err_single` bzw. `err_double`.

Stellen Sie den Verlauf der Abweichung über n in einer Zeichnung dar (verschiedene Farben! Beschriftung!) und kommentieren Sie die Ergebnisse.

Nützliche Befehle: `exp`, `plot`, `hold`

Aufgabe 4 - Die Reihenfolge von Operationen kann das Ergebnis beeinflussen

In der Numerik ist es manchmal vorteilhaft, eine Formel in einen mathematisch äquivalenten Ausdruck umzuformen. Dies soll an folgendem Beispiel demonstriert werden.
Bearbeiten Sie die Aufgabe in der MATLAB-Datei `aufgabe1_4.m`.

$$\sum_{k=1}^{K}\frac{1}{k^2} \quad \text{konvergiert für} \quad \lim_{K\to\infty} \quad \text{zu} \quad \frac{\pi^2}{6}.$$

1. Berechnen Sie die Summe mit einfacher Genauigkeit (single precision) bis $K = 10^8$. Brechen Sie die Schleife ab, wenn sich die Summe nicht mehr verändert und geben Sie das k zu diesem Zeitpunkt aus.

   Speichern Sie die Abweichung zu $\frac{\pi^2}{6}$ in `err_forward`.

2. Führen Sie nun die Summation in umgekehrter Reihenfolge ($k = K, K-1, \ldots 1$) durch und speichern Sie die Abweichung zu $\frac{\pi^2}{6}$ in `err_backward`.

3. Vergleichen und begründen Sie die Ergebnisse.

Aufgabe 5 - Power-Iteration

Die Power-Iteration-Methode zur numerischen Bestimmung des größten Eigenwerts und des dazugehörigen Eigenvektors einer Matrix $\mathbf{M}$ ist wie folgt definiert:

$$\begin{aligned}\mathbf{x} &:= \mathbf{M}\mathbf{x}\\ \mathbf{x} &:= \frac{\mathbf{x}}{||\mathbf{x}||}\\ \lambda &:= \mathbf{x}^T\mathbf{M}\mathbf{x} \text{ bis } \mathbf{x} \text{ konvergiert.}\end{aligned}$$

Wenn $\mathbf{M}$ genau einen betragsmäßig höchsten Eigenwert besitzt, dann konvergiert $\mathbf{x}$ zu einem Vielfachen des korrespondierenden Eigenvektors.

Aufgabenteil I

In diesem Aufgabenteil soll die Methode verstanden und veranschaulicht werden.
Gegeben ist die Matrix $\mathbf{A} = \begin{bmatrix} 0.4 & -0.6 & 0.2 \\ -0.3 & 0.7 & -0.4 \\ -0.1 & -0.4 & 0.5 \end{bmatrix}$
Bearbeiten Sie die Aufgabe in der MATLAB-Datei `aufgabe1_5.m`.

a) Welchen Rang hat die Matrix $\mathbf{A}$? Was bedeutet dies für die Eigenwerte?
Berechnen Sie die Eigenwerte und die dazugehörigen Eigenvektoren mit der `eig`-Funktion in MATLAB. Welches ist der größte Eigenwert λ_{max} und zu welchem Eigenvektor $\mathbf{v_{max}}$ gehört er?

b) Implementieren Sie die oben vorgestellte Methode in der vorgegebenen Funktion `showPowerIteration()`:

> Eingabeparameter: Matrix `M`, Startvektor `x`, Anzahl der Iterationen `N`
>
> Ermitteln Sie in jedem Iterationsschritt die Differenz vom normierten $\mathbf{x}$ zum tatsächlichen $\mathbf{v_{max}}$. Als Fehlermaß dient der Euklidsche Abstand.
>
> Stellen Sie diese Differenz sowie den ermittelten größten Eigenwert für $n = 1, 2 \dots N$ graphisch nebeneinander (`subplot`) dar. Skalieren Sie den Betrag der Differenz logarithmisch (`log10`), wählen Sie eine geeignete Achsenskalierung und beschriften Sie die Zeichnung vollständig.

Rufen Sie die Funktion im Hauptprogramm für $\mathbf{M} = \mathbf{A}$, $N = 50$ und 100 zufällig generierte $\mathbf{x}$ auf. Überlagern Sie die Darstellung für alle $\mathbf{x}$ und beschreiben Sie das Ergebnis.

c) Rufen Sie die Funktion im Hauptprogramm für $\mathbf{M} = \mathbf{A}$, $N = 10$ und $\mathbf{x} = \mathbf{v_{max}}$ auf.
Was geschieht im Gegensatz zu b) und warum?

d) Rufen Sie die Funktion im Hauptprogramm für $\mathbf{M} = \mathbf{A}$, $N = 10$ und $\mathbf{x} = \mathbf{v_2}$ auf, dem Eigenvektor zum betragsmäßig zweitgrößten Eigenwert.
Was geschieht im Gegensatz zu b) und c) und warum?

e) Rufen Sie die Funktion im Hauptprogramm für $\mathbf{M} = \mathbf{A}$, $N = 50$ und $\mathbf{x} = \mathbf{v_{min}} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$, den dritten Eigenvektor, auf.
Warum geschieht hier etwas unterschiedliches als in d)?

Aufgabenteil II

In diesem Aufgabenteil soll die Rechenzeit der Power-Iteration-Methode untersucht werden. Hierzu ist eine Modifikation der bisher implementierten Funktion nötig.

f) Um die Funktionalität von Aufgabenteil I nicht zu beeinträchtigen, implementieren Sie für Aufgabenteil II die Funktion `measurePowerIteration()`:

> Eingabeparameter: Wie `showPowerIteration()`, außerdem Abbruchparameter `epsilon`
> Rückgabewert: Rechenzeit `time`
>
> Berechnen Sie für jede Iteration den geschätzten größten Eigenwert von $\mathbf{M}$.
>
> Ermitteln Sie in jedem Iterationsschritt die Differenz vom normierten $\mathbf{x}$ zum tatsächlichen $\mathbf{v_{max}}$.
>
> Brechen Sie die Methode ab, sobald diese Differenz kleiner als ϵ wird
>
> Bestimmen Sie die benötigte Rechenzeit sowie die Differenz für alle durchlaufenen n und stellen Sie den Zusammenhang graphisch dar. Skalieren Sie die Differenz logarithmisch (`log10`) und beschriften Sie die Zeichnung vollständig.

Bestimmen Sie, wie lange die `eig`-Funktion für die Bestimmung von $\mathbf{v_{max}}$ von $\mathbf{A}$ benötigt und speichern Sie den Wert in `eig_time_A`.
Rufen Sie nun die Funktion `showPowerIteration()` für $\mathbf{A}$, einen zufälligen Vektor $\mathbf{x}$, $N = 1000$ sowie $\epsilon = 10^{-12}$ auf. Speichern Sie die benötigte Rechenzeit in `pi_time_A`, vergleichen und kommentieren Sie die Ergebnisse.

g) Die Matrix $\mathbf{B}$ sei eine quadratische Zufallsmatrix der Dimension 1000. Berechnen Sie wie in f) die Rechenzeiten für beide Methoden und speichern Sie die Ergebnisse in `eig_time_B` sowie `pi_time_B`.
Was beobachten Sie?

h) Nennen Sie einige mögliche Nachteile bzw. Schwachstellen der Power-Iteration.