UNIVERSITÄT DES SAARLANDES

Lehrstuhl für Physikalische Chemie und Didaktik

Kurs „MATLAB für Naturwissenschaftler“
im Sommersemester 2019

Till Biskup, Matthias Marquardt, Johann Seibert, Christopher Kay

— Aufgabenblatt 3 vom 30.04.2019 —

Aufgabe 3—1 (Präzision numerischer Datentypen)

Reelle Zahlen lassen sich aus nachvollziehbaren Gründen numerisch nicht exakt in Computersystemen darstellen. Auch wenn MATLAB darauf optimiert ist, dass diese Tatsache selten zum Problem wird, kann man bewusst und gezielt Fehler durch die endliche Genauigkeit der Darstellung reeller Zahlen hervorrufen.

a) Wie wir wissen, gilt $\sin(\pi) = 0$. Da die Konstante `pi` in MATLAB allerdings nicht exakt der Kreiszahl π entspricht, liefert uns MATLAB ein anderes Ergebnis.

   Geben Sie in MATLAB die beiden folgenden Befehle ein und erklären Sie (sich) das Ergebnis:

   ```
   sin(pi) == 0
   sin(pi) < eps(1)
   ```

b) Die numerische Genauigkeit ϵ hängt von der Größe der Zahl ab, die man gerade betrachtet. Den jeweiligen Wert kann man in MATLAB mit dem Befehl `eps()` erhalten, der als Argument die betreffende Zahl akzeptiert.

   Stellen Sie die numerische Genauigkeit von MATLAB für das Intervall $[0, 1]$ grafisch dar. Experimentieren Sie dabei mit unterschiedlich großen Zwischenintervallen (0,1, 0,01, 0,001) und achten Sie darauf, dass in der Grafik auf der x-Achse immer das angegebene Intervall $[0, 1]$ angegeben wird.

c) Um das Ergebnis des Rundungsfehlers in MATLAB zu demonstrieren, eignet sich ein Polynom siebter Ordnung recht gut. Das Beispiel ist aus dem online auf den Seiten von MathWorks verfügbaren Buch „Numerical Computing with MATLAB“ von Cleve Moler[1] entnommen.

   Definieren Sie sich in MATLAB die beiden folgenden Polynome

$$f(x) = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1$$
$$g(x) = (x - 1)^7$$

   und stellen Sie sie in derselben Abbildung für x im Intervall $[0.988, 1.012]$ mit einer Schrittweite von 0.0001 dar. Was Sie sehen, ist das Resultat von Rundungsfehlern für das erste Polynom, $f(x)$. Der Grund hierfür ist, dass die Werte für $f(x)$ durch Summen und Differenzen von Zahlen mit einer maximalen Größe von $35 \cdot 1.012^4$ berechnet werden.

[1] `https://de.mathworks.com/moler/index_ncm.html`

Aufgabe 3—2 (Formatierte Ausgabe von Zeichenketten)

MATLAB beherrscht die formatierte Ausgabe von Zeichenketten und lehnt sich bei der dazu verwendeten Syntax eng an die Programmiersprache C an. Die beiden Befehle zur formatierten Ausgabe von Zeichenketten sind `fprintf` für die Ausgabe in eine Datei (oder auf die Kommandozeile) und `sprintf` für die Ausgabe in eine Stringvariable.

Diese Möglichkeiten eignen sich u.a. zur formatierten Ausgabe von Datentabellen, sowohl auf der MATLAB-Kommandozeile als auch in Textdateien.

a) Erzeugen Sie sich die Vektoren `x` und `y` und über die Befehle `x = 0:0.2:2` und `y = sin(x)` und stellen Sie die beiden in einer Tabelle dar. Experimentieren Sie mit den unterschiedlichen Formatierungsmöglichkeiten für Gleitkommazahlen, die MATLAB Ihnen bietet.

b) Spielen Sie mit den Möglichkeiten, die Datentabelle mit Spaltenköpfen zu versehen. Ihr Ergebnis sollte so aussehen wie in Listing 1 gezeigt.

   Beachten Sie, dass die Abstände zwischen den Spalten nicht manuell mit Leerzeichen erzeugt wurden, sondern über Tabulatoren.

Listing 1: Formatierte Ausgabe einer Datentabelle mit Spaltenköpfen

```
x       sin(x)
0.0     0.000000000
0.2     0.198669331
0.4     0.389418342
0.6     0.564642473
0.8     0.717356091
1.0     0.841470985
1.2     0.932039086
1.4     0.985449730
1.6     0.999573603
1.8     0.973847631
2.0     0.909297427
```

Aufgabe 3—3 (Schleifen und Entscheidungsstrukturen)

Schleifen (`for`, `while`) und Entscheidungsstrukturen (`if`, `switch`) gehören zu den essentiellen Konstrukten fast jeder Programmiersprache.

a) Schreiben Sie eine Funktion `unityMatrix`, der Sie als Argument die Dimension (als Skalar) übergeben und die Ihnen unter Verwendung von Schleifen und Entscheidungsstrukturen eine quadratische Einheitsmatrix der entsprechenden Dimension zurückgibt.

   Die gleiche Funktionalität erhalten Sie durch den MATLAB-Befehl `eye`.

b) Schreiben Sie (quasi als Erweiterung der obigen Funktion) eine Funktion `diagonalMatrix`, der Sie als Argument einen Vektor übergeben und die Ihnen, wiederum unter Verwendung von Schleifen und Entscheidungsstrukturen, eine quadratische Diagonalmatrix der entsprechenden Dimension des übergebenen Vektors zurückgibt, die auf ihren Diagonalelementen die entsprechenden Elemente des Vektors und ansonsten nur Nullen aufweist.

   Die gleiche Funktionalität erhalten Sie durch den MATLAB-Befehl `diag`.

Hinweis: Die hier aufgeführten Beispiele dienen lediglich dazu, mit Schleifen bekannt zu werden. Die Verwendung von Schleifen für die elementweise Operation auf Matrizen sollte in MATLAB wann immer möglich vermieden werden, da die alternativen, durch spezielle Funktionen bereitgestellten Matrixoperationen teilweise um ein Vielfaches schneller ausgeführt werden als die Variante mit Schleifen direkt in MATLAB. (Das liegt u.a. daran, dass die Matrixoperationen direkt auf in Fortran programmierte LAPACK- und BLAS-Routinen zurückgreifen.)

Aufgabe 3—4 (Entscheidungsstrukturen)

Ein häufiger Einsatzort von Entscheidungsstrukturen mit mehr als zwei Zweigen ist das Ausführen von Code in einer Funktion abhängig von einem vom Nutzer als Argument übergebbaren Schalter.

Grundsätzlich lassen sich solche Abfragen natürlich mit `if`-Strukturen abbilden. Das wird allerdings bei mehr als zwei oder drei Möglichkeiten schnell unübersichtlich. Hier ist die `switch-case`-Struktur eine große Hilfe, auch wenn sie nicht in jedem Fall entsprechende `if`-Strukturen ersetzen kann.

Stellen Sie sich vor, Sie hätten eine Funktion geschrieben, die MATLAB-Abbildungen speichern kann, und der man als optionales Argument eine Größenangabe („small“, „medium“, „large“) übergeben kann. Wir werden hier nicht die gesamte Funktion entwickeln, sondern uns lediglich um die Abfrage des (letztlich optionalen) Parameters kümmern.

a) Schreiben Sie eine Test-Funktion „`saveFigure`“, die als Argument einen String „`size`“ versteht.

   Die Funktionsdeklaration sollte wie folgt aussehen: `function saveFigure(size)`, der Aufruf wäre entsprechend z.B. `saveFigure('large')`.

b) Implementieren Sie in dieser Funktion die entsprechende Entscheidungsstruktur, einmal mit einer `if`- und einmal mit einer `switch`-Anweisung.

   Geben Sie jeweils je nach Parameter zum Test einen kurzen Text auf der Kommandozeile aus. Darüber hinaus soll eine Warnung ausgegeben werden, wenn der Nutzer einen anderen Parameter eingegeben hat als vorgesehen.

c) **Zusatzaufgabe:** Sorgen Sie (mit einem einzelnen Befehl) dafür, dass die Abfrage nicht nach Groß- und Kleinschreibung unterscheidet.

   Das ist gerade bei längeren (textlichen) Argumenten, die als Schalter Verwendung finden und vom Nutzer eingegeben werden, sehr nützlich und verbessert die Nutzbarkeit ganz wesentlich.

d) **Zusatzaufgabe:** Wie können Sie unter Verwendung der `switch`-Struktur elegant erreichen, dass statt des ausgeschriebenen Arguments („large“, ...) auch der (in diesem Fall eindeutige) Anfangsbuchstabe genügt?

e) **Zusatzaufgabe:** Welche Situation können Sie sich vorstellen, in der Sie nicht auf eine `switch`-Struktur zurückgreifen können, sondern tatsächlich mit einer (ggf. auch längeren) `if-elseif-else`-Struktur arbeiten müssen?