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 2 vom 26.04.2019 —

Aufgabe 2—1 (Tastaturbelegung des MATLAB-Editors)

Konfigurieren Sie den MATLAB-Editor so um, daß er mit einer „sinnvollen“ Tastaturbelegung funktioniert. Der voreingestellte Standard der Tastaturbelegung ist das „Emacs-Standardset“, für die meisten Nutzer ist das „Windows-Standardset“ aber vertrauter, insbesondere was die Tastenkürzel für „Kopieren“, „Ausschneiden“ und „Einfügen“ anbelangt.

Aufgabe 2—2 (Funktionen des MATLAB-Editors)

Laden Sie sich den nachfolgend gezeigten Quellcode von der Internetseite des Kurses[1] herunter und öffnen Sie ihn in MATLAB mit dem MATLAB-Editor.

Listing 1: Fehlerhafter Quellcode

```
x=[1:0.1:40*pi];
y=sin(5*x)*sin(0.1*x).*cos(2*x)
plot(x,y)
xlabel('time); ylabel('intensity');
tic; figure(); maxvalue = 100;
set(gca,'XLim',[0 maxvalue]); set(gca,'YLim',[0 1]);
hold on; plot([0 maxvalue],[.5 .5])
for k=1:maxvalue value = rand;
  if value<0.5 disp('Lower half'); else disp('Upper half'); end
plot(k,value,'rx'); pause(0.02);
  end
hold off
toc
tic; t = 0:0.01:2*pi; x = 4 * cos(3*t); y = 4 * cos(4*t+pi/2);
figure; hold on; for k=1:length(x) plot(x(k),y(k),'r.'); pause(0.001); end
```

Zur Beachtung: Dieses Beispiel ist unkommentiert und daher schlechter Quellcode. Des Weiteren geht es dieses eine Mal auch nicht darum, dass Sie im Detail verstehen, was der Quellcode macht. Es dient hier lediglich der Demonstration der Fähigkeiten des MATLAB-Editors. Spätestens nach den nächsten Lektionen wissen Sie, wie Sie in dieser Hinsicht ordentlich(er)en Quellcode schreiben.

Sie bekommen eine Reihe von Fehlern und Warnungen angezeigt, sowohl durch Markierung der fehlerhaften oder bemängelten Codeteile als auch durch Markierungen in Rot und Orange am rechten Rand des Fensters. Beheben Sie der Reihe nach alle Fehler und Warnungen. Nutzen Sie dazu die Hilfe, die Ihnen MATLAB über den eingebauten statischen Code-Analysator (mlint) anbietet.

[1] https://www.till-biskup.de/de/lehre/matlab/ss2019/material/04/

Aufgabe 2—3 (Gauß-Funktion)

Schreiben Sie die Ihnen von Aufgabenblatt 1 bekannte Gauß-Funktion

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

als MATLAB-Funktion `gaussian.m` um. Die Funktion soll dabei die nachfolgend definierte Schnittstelle besitzen:

Listing 2: Schnittstelle der Funktion gaussian.m

```
y = gaussian(x,sigma,mu)
```

Hierbei seien `y` und `x` jeweils Vektoren und `sigma` und `mu` Skalare.

Zugegeben gibt es keinen Unterschied im Aufruf, wenn Sie die Funktion direkt als anonyme Funktion auf der Kommandozeile definieren, wie Sie das bei der Bearbeitung des vorangegangenen Aufgabenblattes getan haben.

Der große Vorteil der Definition als Funktion in einer entsprechenden Datei ist die viel bessere Wiederverwendbarkeit. Dazu muss der Ordner, in dem sich die Datei befindet, lediglich zum MATLAB-Suchpfad hinzugefügt werden. Ein weiterer Vorteil ist die Dokumentierbarkeit der Funktion. Darauf werden wir später noch einmal zurückkommen.

Die nächste Aufgabe widmet sich einer Funktion, die hinreichend komplex ist, so dass man sie nur noch schwer auf der Kommandozeile definieren kann.

Aufgabe 2—4 (Euler-Drehmatrix)

Koordinatentransformationen spielen in der Physikalischen Chemie (und weit darüber hinaus) oftmals eine große Rolle. Jede beliebige Drehung im euklidischen dreidimensionalen Raum kann durch eine Abfolge dreier einzelner Drehungen beschrieben werden, die jeweils durch einen Winkel charakterisiert werden. Man nennt diesen Satz dreier Winkel häufig auch Euler-Winkel und bezeichnet sie mit (α, β, γ) bzw. (ϕ, θ, ψ).

Es gibt eine ganze Reihe äquivalenter Möglichkeiten für die Drehung, und je nach Kontext (Physik/Physikalische Chemie, Ingenieurswesen) werden unterschiedliche Konventionen verwendet. Wir wollen uns hier auf die $zy'z''$-Konvention beschränken, bei der zunächst um die z-Achse, dann um die (neue) y'-Achse und anschließend wieder um die (dann aktuelle) z''-Achse gedreht wird. Die Euler-Drehmatrix $M_{zy'z''}$ ist nachfolgend sowohl für die drei Einzeldrehungen als auch zusammengefasst als eine Matrix gezeigt.

$$M_{zy'z''} = \begin{pmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$M_{zy'z''} = \begin{pmatrix} -\sin\alpha\sin\gamma + \cos\alpha\cos\beta\cos\gamma & \cos\alpha\sin\gamma + \sin\alpha\cos\beta\cos\gamma & -\sin\beta\cos\gamma \\ -\sin\alpha\cos\gamma - \cos\alpha\cos\beta\sin\gamma & \cos\alpha\cos\gamma - \sin\alpha\cos\beta\sin\gamma & \sin\beta\sin\gamma \\ \cos\alpha\sin\beta & \sin\alpha\sin\beta & \cos\beta \end{pmatrix}$$

Schreiben Sie eine MATLAB-Funktion `rotationMatrix.m`, die die Euler-Drehmatrix für die Drehung gemäß der oben ausgeführten $zy'z''$-Konvention durchführt. Die Funktion soll dabei die nachfolgend definierte Schnittstelle besitzen:

Listing 3: Schnittstelle der Funktion rotationMatrix.m

```
M = rotationMatrix(alpha,beta,gamma)
```

Hierbei seien die drei Euler-Winkel `alpha`, `beta` und `gamma` jeweils Skalare, der Rückgabeparameter `M` eine Matrix der Dimension 3×3.

Diese Drehmatrizen haben spezielle Eigenschaften. U.a. ist ihre Transponierte identisch mit ihrer Inversen. Überzeugen Sie sich davon, indem Sie eine solche Matrix für einen Satz an Eulerwinkeln erzeugen und danach die Identität der Transponierten mit der Inversen durch einen geeigneten MATLAB-Befehl auf der Kommandozeile überprüfen.

Zusatzaufgabe: Auch wenn es in Ihrem Fall (erst einmal) nicht auf die Geschwindigkeit der Abarbeitung ankommt: Überlegen Sie, wie Sie die wiederholten Funktionsaufrufe der trigonometrischen Funktionen für denselben Winkel minimieren können.

Zusatzaufgabe: Testen Sie die Korrektheit Ihrer Euler-Drehmatrix. Gehen Sie dazu wie folgt vor:

a) Erzeugen Sie sich einen Einheitsvektor und drehen Sie ihn mittels der Euler-Drehung um nur einen Winkel um 90°.

   Hinweis: Schon durch Anschauen des Ergebnisses sollten Sie erkennen können, dass und um welche Achse Sie den Vektor um 90° gedreht haben.

b) Erzeugen Sie sich fünf zufällige Vektoren im $\mathbb{R}^3$, drehen Sie diese jeweils um *einen* Winkel um 90° und überprüfen Sie die korrekte Drehung mittels des Skalarproduktes des ursprünglichen und des gedrehten Vektors. Permutieren Sie dabei zusätzlich die Winkel durch, um die Sie drehen.

   Beachten Sie, dass die Komponente dieser zufälligen Vektoren entlang der Drehachse jeweils verschwinden muss, um tatsächlich zu einer Drehung um 90° zu führen.

   Hinweis: Diese Aufgabe lässt sich recht bequem in Form eines MATLAB-Skriptes bearbeiten.