OTH Regensburg

Studiengang
IN, IT B. Sc.
Prof. Dr. Martin Weiß
Sommersemester 2019

Computerarithmetik und Rechenverfahren

1. Praktikum: Grundlagen MATLAB

1.1. Funktionen

Das Skalarprodukt dient zur Berechnung von Winkeln zwischen Vektoren, insb. zum Test auf Orthogonalität. Es ist definiert durch

$$\begin{aligned} \langle v,w\rangle &= v_1\cdot w_1 + v_2\cdot w_2, \qquad v,w\in\mathbb{R}^2 \\ \langle v,w\rangle &= v_1\cdot w_1 + v_2\cdot w_2 + v_3\cdot w_3, \qquad v,w\in\mathbb{R}^3 \\ \text{allgemein: } \langle v,w\rangle &= \sum_{i=1}^{n} v_i\cdot w_i, \qquad v,w\in\mathbb{R}^n \end{aligned}$$

Schreiben Sie Funktionen in eigenen `.m`-Files zur Berechnung des Skalarprodukts für die 3 Fälle - natürlich reicht auch der allgemeinste Fall $\mathbb{R}^n$! Die Länge der Vektoren soll *nicht* als Parameter übergeben werden.

Varianten: Sie können die Formeln für $\mathbb{R}^2$ und $\mathbb{R}^3$ verwenden, oder Schleifen, oder in MATLAB besser auf Schleifen verzichten aufgrund der Identität

$$\langle v,w\rangle = \sum_{i=1}^{n} v_i\cdot w_i = v^t w$$

Bestätigen Sie:

$$\langle \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}3\\4\end{bmatrix}\rangle = 11, \qquad \langle \begin{bmatrix}1\\2\\3\end{bmatrix}, \begin{bmatrix}3\\4\\5\end{bmatrix}\rangle = 26$$

Implementieren Sie Ihre Funktionen als *anonyme Funktionen*, für die kein eigenes `.m`-File benötigt wird, nach dem Muster

```
f = @(x) x^2;
g = @(x,a,b,c) a*x^2+b*x+c;

f(4)
g(5,6,7,8)
```

1.2. Mehr zu Funktionen

MATLAB-Funktionen können mehrere Rückgabewerte haben. Als Beispiel betrachten wie die Umwandlung einer komplexen Zahl $x+iy$ (oder eines Vektors $(x,y)\in\mathbb{R}^2$) in Polarkoordinaten und zurück:

```
function [r, phi] = cart2polar(x, y)
r = sqrt(x*x + y*y);    % oder: r = norm([x, y])
phi = atan2(y, x);
end
```

Alternativ können Sie auch mit nur einem Argument als Vektor arbeiten:

```
function [r, phi] = cart2polarVec(v)
r = sqrt(v(1)*v(1) + v(2)*v(2));    % oder: r = norm(v);
phi = atan2(v(2), v(1));
end
```

Schreiben Sie im gleichen Stil zwei Umkehrfunktionen

```
function [x, y] = polar2cart(r, phi)
function v = polar2cartVec(r, phi)
```

Je nach Anzahl der Variablen in Aufrufen

```
[r1, phi1] = cart2polarVec(v1)
[r2] = cart2polarVec(v2)
r3 = cart2polarVec(v3)
```

werden ein oder mehrere Rückgabewerte geliefert. Für mehr als 1 Rückgabewert müssen die Variablen in eckigen Klammern angegeben werden, bei 1 Rückgabewert sind die Klammern optional.

Testen Sie Ihre Funktionen (wie?). Schreiben Sie insb. automatisierte Tests, die bestätigen: Die Funktionen sind tatsächlich invers zueinander. (Das stimmt nicht ganz: es gibt 2 Probleme!)

1.3. Symbolisches Rechnen mit Funktionen: Erklärung

Symbolisches Rechnen ist in MATLAB nur über die Symbolic Math Toolbox verfügbar, ein Zusatzpaket für ca. 1800 €.

Legen Sie symbolische Objekte `x`, `f1`, `f2`, `f3` an, und leiten Sie die Ausdrücke nach der Variablen `x` ab:

```
syms x f1 f2 f3
f1=x^2
diff(f1, x)
f2=x^3+3*x^2-4*x+2
diff(f2,x)
f3=x^2+x.*sin(x)+exp(x)
d3=diff(f3,x)
hd3=matlabFunction(d3) % handle auf anonyme Funktion
fh3(0)          % Auswertung
% d3(0)         % gibt Fehler
subs(d3,x,0)    % "Auswertung" durch Ersetzung
```

Dabei sind `x`, `f1`, `f2`, `f3` "Ausdrücke" und keine Funktionen mit den aus der Mathematik erwarteten Operationen, insb. Auswertung an einer Stelle: Um Ausdrücke auszwerten, muss man Teilausdrücke mit `subs` substituieren: Erklären Sie die Ausgaben von

```
x = 2
f1
subs(f1)
x = 3
subs(f1)
y = subs(f1)
x = 5
y
```

anhand von http://de.mathworks.com/help/symbolic/use-subs-to-evaluate-expressions-and-functions.html?requestedDomain=www.mathworks.com

Löschen Sie die Objekte aus dem Workspace:

```
clear x f1 f2 f3
```

1.4. Symbolisches Rechnen mit Funktionen: Selber machen

Bestimmen Sie für $f(x) = ax^2 + bx + c$, mit a, b, c Parametern, *mit Hilfe der Symbolic Math Toolbox* die erste und zweite Ableitung. Bestimmen Sie über `solve` die Lage des Scheitelpunkts, als Funktion von a, b, c. Berechnen Sie den Funktionswert am Scheitelpunkt.

Verwenden Sie die Funktionen `solve` und `subs` der Symbolic Toolbox. Erarbeiten Sie selbst die Bedeutung der Funktion `solve`.

Mit `matlabFunction` können Sie eine Symbolic Toolbox Funktion in eine MATLAB-Funktion umwandeln, die Sie über einen handle aufrufen können.

1.5. Funktionen schreiben, Probleme mit großen Zahlen vermeiden

a) Machen Sie sich anhand der Funktion `fakultaet` klar, wie sich eine MATLAB-Funktion syntaktisch von ihrem Äquivalent in C/C++, C#, Java unterscheidet.

```
function f = fakultaet( n )
f = 1;
for k = 1:n
    f = f * k;
end
end
```

oder als rekursive Variante:

```
function f = fakultaetRekursiv( n )
if n == 0
    f = 1;
else
    f = n * fakultaetRekursiv(n - 1);
```

```
end
end
```

Wie groß darf n maximal werden?

b) Implementieren Sie eine MATLAB-Funktion

```
function b = binomial(n,k)
```

zur Berechnung des Binomialkoeffizienten

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}, \qquad n,k \in \mathbb{N}_0, 0 \leq k \leq n$$

Beachten Sie insb. die Grenzfälle $n = 0$, $k = 0$. Implementieren Sie zuerst die "naive" Variante entsprechend der mathematischen Definition unter Verwendung von Fakultäten - diese wird für $n,k \geq 171$ scheitern. Implementieren Sie eine bessere Variante, die in einer Schleife den Binomialkoeffizienten aus kleineren Faktoren aufbaut, etwa:

$$\frac{n}{n-k}, \frac{n \cdot (n-1)}{(n-k) \cdot (n-k-1)}, \frac{n \cdot (n-1) \cdot (n-2)}{(n-k) \cdot (n-k-1) \cdot (n-k-2)}, \ldots$$

c) Testen Sie Ihre Funktionen mit

$$\binom{7}{k}, \quad k = 0, \ldots, 7$$
$$\binom{171}{1}, \quad \binom{171}{170}$$

Hinweis: Aus der Vorlesung MA2 wissen Sie (oder leiten schnell wieder aus der Definition her, oder Interpretation "k Objekte aus n Objekten Ziehen ohne Berücksichtigung der Reihenfolge"):

$$\binom{n}{1} = n = \binom{n}{n-1} \qquad \text{für alle } n \in \mathbb{N}.$$

Aus der Mathematik wissen Sie, oder aufgrund der Interpretation "k Objekte aus n Objekten Ziehen ohne Berücksichtigung der Reihenfolge", etwa Lotto: $\binom{n}{k}$ ist immer ganzzahlig. Stimmt das auch bei großen Werten von n und k, sofern das Ergebnis nicht `Inf` ist? Verbessern Sie Ihre Funktion, indem Sie das Ergebnis auf die nächstliegende natürliche Zahl runden. Funktionen: `round`, `floor`, `ceil`.

d) Verwenden Sie das Kommando `disp` zur Ausgabe von Zwischenschritten. Vergleichen Sie dabei den Unterschied der Ausgabe der Kommandos

```
disp('hello, world');
disp('hello, world\n');
disp(x)
% disp('Wert von x: ' x);  % Fehler
% disp('Wert von x: ' num2str(x));  % Fehler
```

```
disp(['Wert von x: ' num2str(x)]);
disp(sprintf('Wert von x: %d\n', x));
disp(sprintf('Wert von x: %g\n', x));
disp(sprintf('Wert von x: %e\n', x));
```

für `x = sqrt(2)` und `x = sqrt(4)`.

e) [***] Werfen Sie einen Blick in die bei MATLAB mitgelieferte Implementierung der Fakultät: `edit factorial.m`

Statt `disp(sprintf(...))` können Sie auch `fprintf` verwenden. Dann können Sie aber nicht so einfach vorne oder hinten Ausgaben in Strings durch Anhängen ergänzen, sparen sich dafür aber die eckigen Klammern.

1.6. Funktionsdefinition und Punkt-Operator

Plotten Sie mit einer selbst programmierten Funktion den Graph der Funktion $x \mapsto x^2$ im Intervall $[-3,5]$; die Funktion soll im Abstand von $\frac{1}{10}$ ohne Verwendung einer Schleife ausgewertet werden ("idiomatische Verwendung von MATLAB ").

Erklären Sie, warum nur eine der Funktionen funktioniert.

```
function y = fQuadrat1(x)
y = x^2;
end
function y = fQuadrat2(x)
y = x.^2;
end
```

1.7. Meßrauschen beim Smartphone

Anzeigen für analoge Sensoren wie Tachometer, Druckmesser usw. zeigen ein Zittern, das aber nicht sehr auffällt, solange die Werte nur optisch abgelesen und nicht weiterverarbeitet werden. Bei digitaler Anzeige des so erfassten Wertes flackern aber die letzten Ziffern, oder müssen geglättet werden, weswegen sich z.B. digitale Tachometer im Auto nicht durchgesetzt haben. Elektronische Sensoren erzeugen fast direkt Spannungswerte, z.B. über die Änderung der Kapazität bei Beschleunigungssensoren, die miniaturisierte Feder-Masse-Systeme darstellen.

Selbst im Ruhezustand zeigen sich kleine Schwankungen, die auch von der weiteren Verarbeitung in AD-Wandlern usw. herrühren können. Wir wollen diesen Effekt am SmartPhone bestätigen und quantifizieren. Ziel ist eine Abschätzung des absoluten Fehlers der Beschleunigungsmessung. Dabei gibt es 2 wesentliche Effekte:

- das beschriebene Meßrauschen, das man üblicherweise Zufallsvariable modelliert (Vorlesung Statistik)
- Fehler in der Kalibrierung: der Mittelwert des Meßrauschens ist nicht 0

Legen Sie Ihr SmartPhone auf den Tisch; optimal auf eine schwingungsdämpfende Unterlage (Jacke ...). Starten Sie mit phyphox eine Messung der Beschleunigungssensoren.

a) Stellen Sie die Messdaten als Histogramm dar, Funktion `histogram`. Experimentieren Sie mit dem Parameter `bins`.

b) Ermitteln ("schätzen") Sie für jede Beschleunigungskomponente den Mittelwert μ (der 0 bzw. in z-Richtung g sein sollte), und die Standardabweichung σ bzw. Varianz σ^2 der Messwerte. Die Formeln für Stichprobenmittel und Stichprobenvarianz lauten bei Messungen $x_1, \ldots, x_n$:

$$\hat{\mu} \equiv \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad \hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

(In der Statistik wird erklärt, warum bei $\hat{\sigma}^2$ der Faktor $\frac{1}{n-1}$ lautet; Stichwort Erwartungstreue).

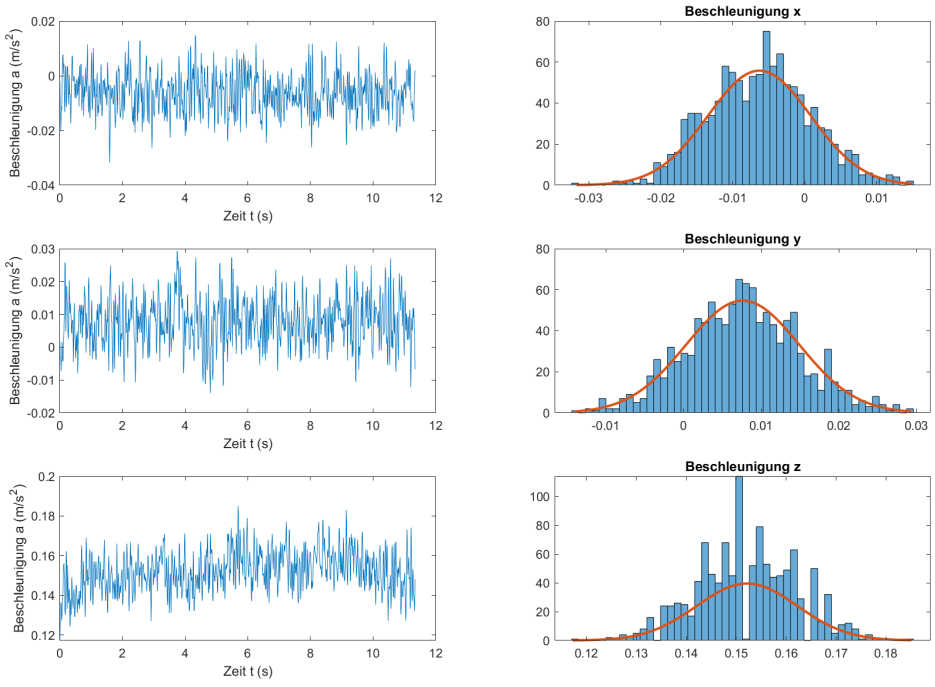
Berechnen Sie $\hat{\mu}$ und $\hat{\sigma}^2$ mit MATLAB, aber *ohne Schleifen*.

c) Oft nimmt man normalverteilte Zufallsvariablen an mit Dichtefunktion

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

Plotten Sie die Dichtefunktion zu Ihren Schätzwerten und vergleichen Sie den quantitativen Verlauf mit dem Histogramm.

d) Verwenden Sie die Funtkion `pd = fitdist(data,'Normal')` aus der Statistics Toolbox. Vergleichen Sie die Ergebnisse mit Ihren Werten.



1.8. ** Umwandlung in Basis b

Implementieren Sie eine MATLAB-Funktion zur Umwandlung nichtnegativer ganzer Zahlen in eine b-adische Darstellung, $2 \leq b \leq 16$ nach der Variante 2 aus der Vorlesung.

Vergleichen Sie als Test mit den MATLAB-Funktionen `dec2bin`, `bin2dec`, `dec2hex`, `hex2dec`, `dec2base`, `base2dec`.