OTH Regensburg

Studiengang
IN, IT B. Sc.
Prof. Dr. Martin Weiß
Sommersemester 2025

Computerarithmetik und Rechenverfahren

0. Praktikum: Erste Schritte in MATLAB

*Die Aufgaben sind in kürzere Päsenzaufgaben und umfangreichere Hausaufgaben unterteilt. Weiterführende Aufgaben sind mit * gekennzeichnet.*

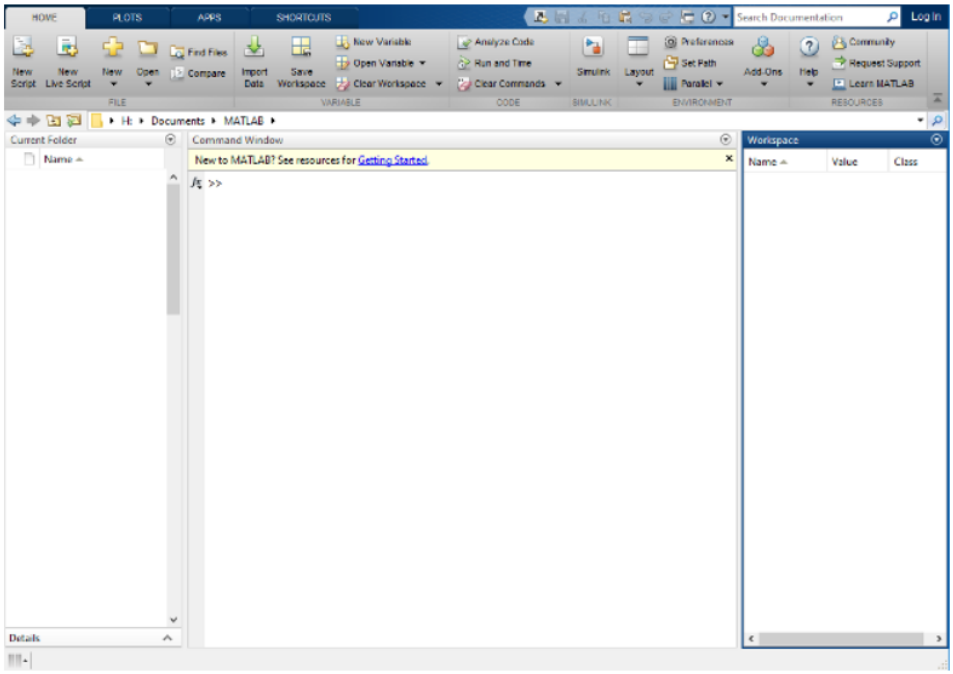
Präsenzaufgaben

Achtung: Einige Sachverhalte sind in MATLAB komplizierter als bei diesen ersten Beispielen - aber die ersten Schritte sollen nicht mit der größten Allgemeinheit befrachtet werden. Fortsetzung folgt. Wenn Sie MATLAB über die Web-Oberfläche `https://matlab.mathworks.com/` nutzen, ist die Bedienung leicht anders.

0.1. MATLAB Desktop: Begriffe aus der Dokumentation

Desktop Basics

When you start MATLAB®, the desktop appears in its default layout.



The desktop includes these panels:

- **Current Folder** — Access your files.
- **Command Window** — Enter commands at the command line, indicated by the prompt (>>).
- **Workspace** — Explore data that you create or import from files.

As you work in MATLAB, you issue commands that create variables and call functions. For example, create a variable

0.2. Wiederholung zur Vorlesung: Vektoren und Matrizen

Eindimensionale Arrays in C, C++, Java entsprechen Vektoren, zweidimensionale Arrays entsprechen Matrizen. Allerdings unterscheidet man in der Mathematik und daher auch in MATLAB Zeilen- und Spaltenvektoren:

```
>> zeile = [1 3 7];
>> spalte = [2
>>           4
>>           9];
```

Der Zugriff erfolgt über runde Klammern: `zeile(i)` bzw. `spalte(i)`. Es gilt aber:

```
>> size(zeile)
ans =
     1     3
>> size(spalte)
ans =
     3     1
```

Eine Addition eines Zeilen-Vektors in $\mathbb{R}^n$ mit einem Zeilenvektor anderer Größe liefert einen Fehler:

```
>> z2 = [1 3 7 8];
>> zeile + z2
Error using  +
Matrix dimensions must agree.
```

Addition mit eines Zeilen- mit einem Spalten-Vektor führt zu keinem Fehler:

```
>> zeile + 5
ans =
     6     8    12
>> zeile + spalte
ans =
     3     5     9
     5     7    11
    10    12    16
```

Es wird dann "komponentenweise" addiert. [1]

Matrizen können entsprechend definiert werden:

```
>> A = [1 2 3
        4 5 6]
>> B = A'
```

Mit dem Apostroph-Operator `'` kann ein Vektor bzw. eine Matrix transponiert werden.

[1] Das Verhalten ist abhängig von der Version: Bis MATLAB 2015 lieferte die Addition von Zeilen- und Spaltenvektoren einen Fehler!

0.3. Vektoren und Matrizen: selber machen

Gegeben seien die Matrizen und Vektoren:

$$A = \begin{bmatrix} -2 & -3 & -4 & -5 \\ 5 & 6 & 7 & 8 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \qquad C = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

$$e_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad e_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}, \qquad v = \begin{bmatrix} 0,0,1,0 \end{bmatrix}, \qquad w = \begin{bmatrix} 0,0,1,0 \end{bmatrix}^t$$

Erzeugen Sie diese in einem MATLAB-Skript, also einer Datei mit Extension `.m`. Legen Sie dazu mit dem Editor der MATLAB-IDE eine Datei `matrizen.m` in Ihrem Arbeitsverzeichnis an. Wechseln Sie mit dem `cd`-Kommando in das Verzeichnis, das `matrizen.m` enthält; alternativ . Führen Sie das Skript aus, indem Sie `matrizen` in der Shell eingeben. Alternativ können Sie im Editor `F5` drücken. (Mit `F9` können Sie nur den markierten Teil eines Skripts ausführen).

Wichtig: Sie müssen das aktuelle Verzeichnis richtig gesetzt haben. Bei `F5` erscheint ggf. ein Dialog, mit dem Sie in das richtige Verzeichnis wechseln können.

Berechnen Sie in der Shell oder in einem Skript:

$$A+B,\ A^t \cdot B,\ e_1 \cdot A,\ e_2 \cdot A,\ A \cdot w,\ w \cdot A,\ A \cdot v,\ w^t \cdot B,\ B \cdot C,\ C \cdot C,\ C^2,\ C^3,\ C^0$$

Erhalten Sie Meldungen in allen Fällen, in denen Sie Fehler erwarten?

Erzeugen Sie die Matrix `A` mit mindestens 3 verschiedenen Syntax-Varianten.

0.4. Vektoren und Matrizen: selber machen

Lösen Sie die folgenden Aufgaben 6.1, 6.4 (aus einem alten Arbeitsblatt zur Mathematik 1, Original im ELO) mit Hilfe von MATLAB! Sie müssen die Aufgaben ggf. aufbereiten und Teile der Arbeit immer noch selbst machen. Definieren Sie Matrizen und Vektoren, und rechnen Sie mit diesen.

Hilfreiche Kommandos: `inv`, Operator \, `rank`, `null`,`rref`.

Eine Online-Hilfe ist jeweils verfügbar über `doc <Kommando>` bzw. `help <Kommando>`.

6.1 Vektorrechnung

a) Vereinfachen Sie den Vektorterm:

$$2\begin{bmatrix} -2 \\ 1 \\ -2 \end{bmatrix} + 5\begin{bmatrix} -1 \\ -3 \\ 0 \end{bmatrix} + 6\begin{bmatrix} 4 \\ 1 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 \\ 1 \\ -2 \end{bmatrix}$$

Lösung

a)

$$\begin{bmatrix} 13 \\ -8 \\ 16 \end{bmatrix}$$

Ein Gleichungssystem $Ax = b$ mit $A \in \mathrm{Mat}_{n,n}(\mathbb{R})$, $b \in \mathbb{R}^n$ kann mit

```
x=A\b
```

gelöst werden. Lösen Sie:

6.4 Gleichungssysteme Lösen Sie die Gleichungssysteme:

a)

$$\begin{array}{rrrcl} 3x_1 & +2x_2 & -6x_3 & = & 1 \\ x_1 & -x_2 & +2x_3 & = & 8 \\ 4x_1 & -x_2 & -3x_3 & = & 3 \end{array}$$

b)

$$\begin{array}{rrrcl} 3x_1 & +2x_2 & +x_3 & = & 1 \\ x_1 & -3x_2 & +2x_3 & = & 2 \\ 2x_1 & +x_2 & -3x_3 & = & 3 \end{array}$$

c) Koeffizientenmatrix und rechte Seite:

$$\begin{bmatrix} 3 & 2 & 1 \\ 1 & -3 & 2 \\ 2 & 1 & -3 \end{bmatrix}, \qquad \begin{bmatrix} 9 \\ 15 \\ -12 \end{bmatrix}$$

Lösung

a)

$$\begin{bmatrix} 5 \\ 5 \\ 4 \end{bmatrix}$$

b)

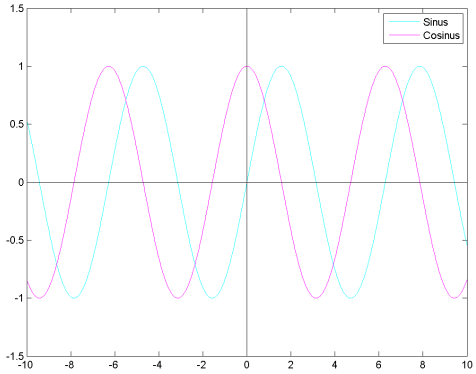
$$\begin{bmatrix} 1 \\ -\frac{5}{7} \\ -\frac{4}{7} \end{bmatrix}$$

c)

$$\begin{bmatrix} 2 \\ -1 \\ 5 \end{bmatrix}$$

Hinweis: Brüche können Sie mit `format rat` oder `rats` ausgeben.

0.5. Erzeugen Sie folgende Graphik:



Legen Sie dazu ein Skript in einem `.m`--File an.

Kommandos: `figure`, `clf`, `shg`, `zoom on|off`, `plot`, `legend`

Bei Vorträgen werden Sie üblicherweise gerügt, wenn Sie Graphiken ohne Beschriftung der Achsen verwenden. Ergänzen Sie diese Beschriftungen und einen Titel! Strings werden in MATLAB in einfachen Anführungszeichen eingeschlossen:

```
title('Mein erster Plot')
```

Kommandos: `xlabel`, `ylabel`, `title`

Erzeugen Sie mit dem `print`-Kommando Graphikformate, die Sie mit Zeichenprogrammen, LaTeXoder Word weiterverwenden können.

Hinweis: Es gibt keinen Befehl, der ein Achsenkreuz einzeichnet (x- und y-Achse). Hier muss man sich mit `line`-Befehlen behelfen, oder eine Funktion schreiben / verwenden wie z.B. unter `http://www.mathworks.com/matlabcentral/fileexchange/22956-axescenter`. Mit `xline(0)` und `yline(0)` kann man mit zwei Komandos ein Koordinatenkreuz zeichnen.

Des Thema ist bei The MathWorks offenbar bekannt, aber seit vielen Releases wurde keine entsprechende Option ergänzt. Das `grid`-Kommando kann teilweise als Ersatz dienen.

Jeder Graphik-Kommando erzeugt ein internes Objekt, und liefert einen *handle* darauf zurück. Mit `get`- und `set`-Kommandos können Eigenschaften von Graphik-Objekten gelesen und gesetzt werden:

```
>> x1 = 1; x2 = 3; y1 = -2; y2 = 5;
>> handle = line([x1 x2], [y1 y2]);
>> shg
>> set(handle, 'LineWidth', 3)
>> get(handle, 'LineWidth')
ans =
     3
>> get(handle)
    AlignVertexCenters: 'off'
            Annotation: [1x1 matlab.graphics.eventdata.Annotation]
          BeingDeleted: 'off'
    ...
```

Hausaufgaben

0.6. phyphox

Installieren Sie die App phyphox aus dem App-Store. Zeichnen Sie die Signale des Beschleunigungssensors auf. Exportieren Sie die Signale als `.csv`-Datei.

Analysieren Sie das Skript `phyphoxEinlesenDarstellen` aus dem ELO zum Einlesen und Darstellen der Daten; eine Beispieldatei `Raw Data.csv` ist ebenfalls abgelegt. Mit `doc` können Sie die MATLAB-Hilfeseiten zu unbekannten Kommandos aufrufen.

Einstellungen zum `.csv`-Export:

Die `.csv`-Datei mit Beschleunigungswerten ist zeilenweise aufgebaut:

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ t_i & a_{x,i} & a_{y,i} & a_{z,i} & a_i \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Zeile i enthält

- den Zeitpunkt i der Abtastung der Beschleunigungssensoren (näherungsweise in Takt 0.01, aber weit von Echtzeitforderungen entfernt. Die Abweichungen lassen sich nicht durch Rundungsfehler erklären.)
- die Werte $a_{x,i}, a_{x,i}, a_{x,i}$ der 3 Beschleunigungssensoren in x-, y-, und z-Richtung zur Zeit t_i
- die Norm $\equiv$ den Betrag a_i des Vektors der Beschleunigungswerte

$$a_i = \sqrt{a_{x,i}^2 + a_{y,i}^2 + a_{z,i}^2}$$

  Dieser Wert wird also nicht gemessen, sondern aus den Vektorkomponenten berechnet.

Die komplizierteste Anweisung im Skript `phyphoxEinlesenDarstellen` lautet
`aCheck = sqrt(sum(a.^2, 2));`, und berechnet ohne explizite Schleife die betraglichen Beschleunigungen.

Schrittweise Erklärung:

- Für eine Matrix `A` berechnet `A.^2` elementweise die Quadrate.

```
A =
     1     2     3
     4     5     6
>> A.^2
ans =
     1     4     9
    16    25    36
>> A^2
Error using  ^  (line 51)
Incorrect dimensions for raising a matrix to a power.
...
```

A^2 berechnet dagegen das Produkt $A \cdot A$, das nur für quadratische Matrizen definiert ist. Das elementweise Produkt von Matrizen wird in der Mathematik selten verwendet und heisst *Hadamard-Produkt*. Selbst für quadratische Matrizen liefern `A^2` und `A.^2` in der Regel verschiedene Ergebnisse:

```
A =
     1     2
     3     4
>> A.^2
ans =
     1     4
     9    16
>> A^2
ans =
     7    10
    15    22
```

- Für einen Zeilen- oder Spaltenvektor `v` berechnet `sum(v)` die Summe der Elemente ohne formale Schleife (die natürlich im MATLAB-Kern steckt; wir haben keinen Vektorrechner).

  Für eine $m \times n$-Matrix berechnet `sum(A,2)` die Summe entlang der "zweiten Dimension", d.h. der Spalten, die mit dem zweiten Index adressiert werden, und liefert einen Vektor mit m Zeilen.

  Für eine $m \times n$-Matrix berechnet `sum(A,1)` die Summe entlang der "ersten Dimension", d.h. der Zeilen, die mit dem ersten Index adressiert werden, und liefert einen Vektor mit n Spalten.

  `sum(a.^2, 2)` liefert also zeilenweise die Summen der quadrierten Beschleunigungswerte in x-, y- und z-Richtung.

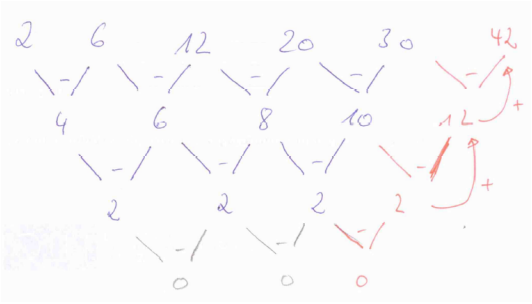
- `sqrt(sum(a.^2, 2))` berechnet elementweise die Wurzel und damit die Beträge der Beschleunigung; hier setzt MATLAB eine Funktion $\sqrt{\cdot} : \mathbb{R} \to \mathbb{R}$ elementweise auf Vektoren $\mathbb{R}^m \to \mathbb{R}^m$ fort.

0.7. Nützliche Funktionen und Anwendungen

Anwendung: In Einstellungstests bei Banken werden z.T. Aufgaben der Art gestellt:

> Setzen Sie die Folge fort!
>
> - 2, 4, 6, 8, 10, ?, ?
> - 3, 5, 7, 9, 11, ?, ?
> - 2, 6, 12, 20, 30, ?

Eine Klasse von Standard-Aufgaben dieser Art kann mit der `diff`-Funktion gelöst werden: Bilde Differenzen der gegebenen Folge, bis die Folge der Differenzen konstant ist. Verlängere dann diese konstante Folge, und erzeuge durch Addition (nach oben in der handschriftlichen Darstellung) die gewünschte Fortsetzung der Folge.



Testen Sie dieses Vorgehen manuell und mit MATLAB!

a) Hilfreiche Funktionen: Die MATLAB-Funktion `diff` berechnet Differenzen aufeinanderfolgender Komponenten von Vektoren, z.B:

```
v = [2     6     12     20     30]
diff(v)
diff(diff(v))
diff(diff(diff(v)))

diff(v,1) % = diff(v)
diff(v,2)
diff(v,3)
```

Geben Sie diese Kommandos in einem Skript ein und bestätigen Sie die Erwartung! Die Funktion `cumsum` ist die Umkehrfunktion; sie liefert die Partialsummen bis zur i-ten Komponente in einem neuen Vektor:

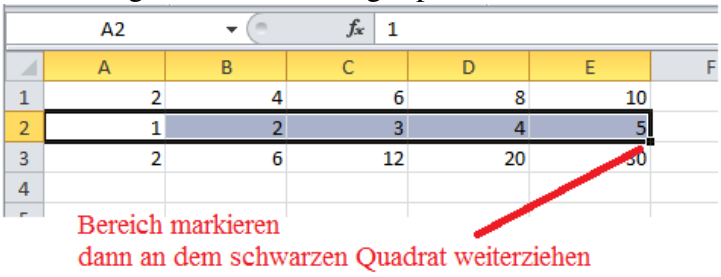
```
v0 = 2*ones(1,7)
v1 = cumsum(v0)
```

b) Bestimmen Sie analog die weiteren Folgenglieder von

- 2, 4, 6, 8, 10, ?, ?
- 1, 4, 9, 16, 25, ? , ?
- 2, 5, 10, 17, 26, ? , ?

c) Konstruieren Sie selbst Aufgaben dieser Art: Schwierigere Aufgaben verwenden Differenzen noch höherer Ordnung. Beeindrucken Sie damit auf der nächsten Party.

d) Versuchen Sie, die Aufgaben mit dem drag-Operator von Excel zu lösen!



e) [***] Warum funktioniert das Vorgehen nicht bei

- 2, 3, 5, 7, 11, 13, 17, ?, ?
- 1, 2, 4, 8, 16, 32, ?, ?
- 1, 3, 7, 15, 31, 63, ?, ?

f) [***] Geben Sie für alle Folgen eine Formel für das n-te Folgenglied an!

0.8. ** "Und glauben Sie mir kein Wort."

In der Vorlesung haben wir ein Beispiel diskutiert, bei dem die Addition von Zahlen nach Standard IEEE 754 im Gegensatz zu $\mathbb{R}$ nicht assoziativ war. Eine Behauptung war: Das liegt nicht an MATLAB, sondern tritt ähnlich in jeder Sprache auf, die Zahlen mit doppelter Genauigkeit verwendet (C, C++, Java, ...).

Ein(e) motivierte(r) Teilnehmer(in) hat diese C-Funktion geschrieben

```
void meinProfHatNichtRecht()
{
    double a, b, c, d, e;
    a = +10^20; b = +17; c = -10; d = 130; e = -10^20;

    printf("a + b + c + d + e = %g\n", a + b + c + d + e);
    printf("a + b + e + c + d = %g\n", a + b + e + c + d);
    printf("a + e + b + c + d = %g\n", a + e + b + c + d);
    printf("a + b + d + e + c = %g\n", a + b + d + e + c);
}
```

und erhält als Ausgabe

```
a + b + c + d + e = 137
a + b + e + c + d = 137
a + e + b + c + d = 137
a + b + d + e + c = 137
```

wie in $\mathbb{R}$. Erklären Sie, warum Ihr Prof in diesem Fall doch recht hat.

Hinweis: Wenn Sie den Source aus dem .pdf in Adobe Acrobat kopieren, machen die Minuszeichen Probleme. Wenn Sie dieselben Probleme haben, und eine Lösung finden: Teilen Sie diese bitte mit.

(***) Das Zitat "Und glauben Sie mir kein Wort." wurde regelmäßig von Erich von Däniken verwendet. Finden Sie heraus, mit welchen Theorien Erich von Däniken bekannt wurde.