

Einführung in die Informatik II

III.2 Dateien

Prof. Dr.-Ing. Marcin Grzegorzek¹
Juniorprofessur für Mustererkennung
Institut für Bildinformatik in der Fakultät IV
Universität Siegen



¹ Die im Rahmen dieser Lehrveranstaltung verwendeten Lernmaterialien wurden uns zum Großteil von Herrn Prof. Dr. Wolfgang Wiechert und Herrn Prof. Dr. Roland Reichardt zur Verfügung gestellt.

Inhaltsverzeichnis

- I. MATLAB-Einführung
- II. Algorithmen
- III. MATLAB-Fortsetzung
 - 1. Internet und Werkzeuge
 - 2. **Dateien**
 - 3. Visualisierung
 - 4. Visualisierung von 3D-Daten
 - 5. Optimierung

Überblick

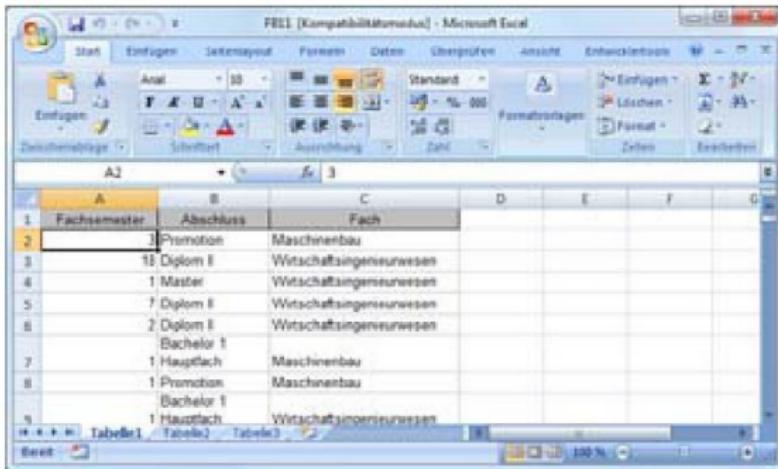
- MATLAB unterstützt mehrere Dateiformate (lesen und schreiben)
- In dieser Vorlesung behandelt werden
 - Excel-Dateiformat
 - MATLAB-Dateiformat
 - Beliebige Textdateien
 - Mit einem Befehl
 - Zeilenweise mit Programmierung
- Alle Dateiformate: `doc fileformats`

Excel-Dateiformat

- Die Dateierweiterung von Excel-Dateien ist i.d.R. „.XLS“ oder „.XLSX“
- In einer Excel-Datei können mehrere Tabellen, gespeichert sein.
- Diese verschiedenen Tabellen heißen *Mappen* (engl.: sheets)
- MATLAB kann nur Zahlen richtig verarbeiten. Spalten mit Text werden (normalerweise) nicht gelesen.
- MATLAB kann Text in Cell-Arrays speichern (→ andere Vorlesung)

Beispiel einer Excel-Tabelle

- In der Beispieltabelle sind Daten über den Fachbereich Maschinenbau gespeichert:
- Spalte A enthält eine Spaltenüberschrift und das Fachsemester des Studierenden



The screenshot shows a Microsoft Excel window with the title 'FEU [Kompatibilitätsmodus] - Microsoft Excel'. The ribbon menu is visible at the top. A table is displayed on the worksheet, with the first row as a header. The columns are labeled 'Fachsemester', 'Abschluss', and 'Fach'. The data rows show various combinations of degree level (Promotion, Diplom II, Master, Diplom I) and degree type (Bachelor, Hauptfach) along with their respective majors (Maschinenbau, Wirtschaftsingeneieurwesen). The table has 10 rows of data, starting from row 2.

| Fachsemester | Abschluss | Fach |
|--------------|-------------|----------------------------|
| 1 | Promotion | Maschinenbau |
| 2 | Diplom II | Wirtschaftsingeneieurwesen |
| 3 | 1 Master | Wirtschaftsingeneieurwesen |
| 4 | Diplom II | Wirtschaftsingeneieurwesen |
| 5 | 2 Diplom II | Wirtschaftsingeneieurwesen |
| 6 | Bachelor I | Maschinenbau |
| 7 | 1 Hauptfach | Maschinenbau |
| 8 | Promotion | Maschinenbau |
| 9 | Bachelor I | Wirtschaftsingeneieurwesen |

Einlesen der Tabelle

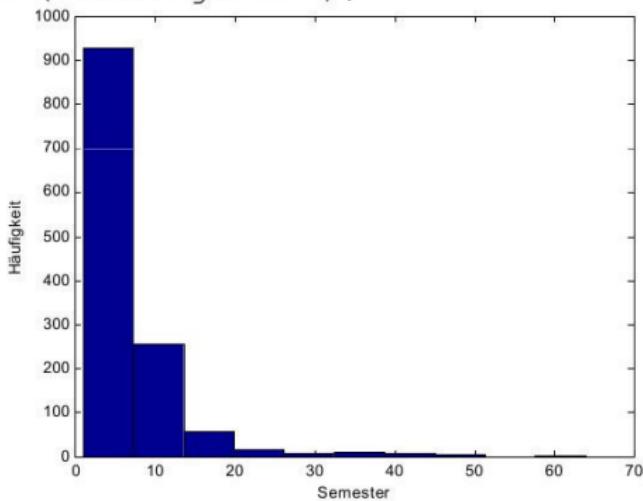
- MATLAB Befehl:
 - $\text{<Variable>} = \text{xlsread}(\text{<Dateiname>})$
- **Beispiel:** SemesterAnzahl=xlsread('FB11.xls');
- Die RückgabevARIABLE ist eine $n \times m$ Matrix
 - n: Anzahl der Zeilen mit numerischen Werten
 - m: Spalten der Excel-Tabelle
- **Beispiel:** size (SemesterAnzahl)
 - [1289, 1]
 - Stehender Vektor (1289 Zeilen und eine Spalte)

Beispiel zur Auswertung

- **Maximales Semester:**
 - `max (SemesterAnzahl)`
 - 64
- **Durchschnittliches Semester:**
 - `mean (SemesterAnzahl)`
 - 6.301784328937161
- **Median Semester (50% der Leute \leq Median)**
 - `median (SemesterAnzahl)`
 - 5

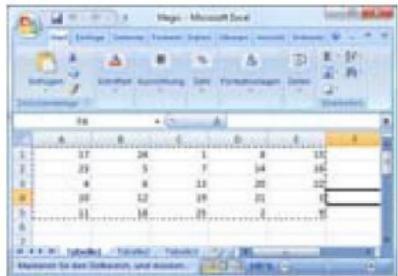
Histogramm

- `hist(SemesterAnzahl)`
- `xlabel('Semester');`
- `ylabel('Häufigkeit');`



Schreiben im Excel-Format

- MATLAB kann in verschiedenen Versionsformaten von Excel schreiben.
 - Dateierweiterung XLS: Excel 97-2003
 - Dateierweiterung XCLxlsx, .xlsb, .xlsm: Excel 2007
- MATLAB Befehl zum Schreiben
 - `xlswrite (<Dateiname>, <Variable>)`
 - Der Dateiname muss die Dateierweiterung enthalten!
- Beispiel:
 - `M=magic(5);`
 - `xlswrite ('Magic.xlsx', M)`



A screenshot of Microsoft Excel showing a 5x5 matrix in cell A1. The matrix is a magic square where the sum of each row, column, and diagonal is 15. The cells are filled with the values 1 through 25 in a specific pattern. The Excel ribbon is visible at the top, and the formula bar shows the formula `=Magic(5)`.

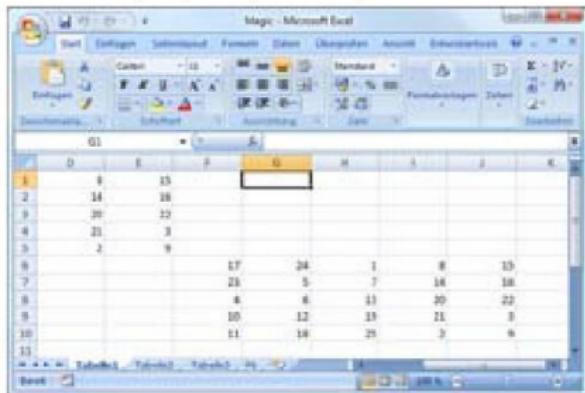
| | 1 | 2 | 3 | 4 | 5 |
|---|----|----|----|----|---|
| 1 | 2 | 7 | 6 | 11 | 9 |
| 2 | 1 | 5 | 9 | 10 | 8 |
| 3 | 8 | 4 | 12 | 13 | 7 |
| 4 | 10 | 12 | 13 | 11 | 5 |
| 5 | 11 | 16 | 17 | 14 | 3 |

xlswrite

- Mit dem Befehl `xlswrite` kann man auch
 - An bestimmte Stellen innerhalb einer Excel-Tabelle schreiben
 - In bestimmte Mappen schreiben, oder diese neu anlegen.
- `xlswrite (<Dateiname>, <Variable>, <Mappe>, <Bereich>)`
- `<Mappe>` ist eine Zeichenkette mit dem Namen
- `<Bereich>` ist die Zellenangabe von den gegenüberliegenden Ecken des Bereichs
 - D2:H4 ist beispielsweise eine 3 x 5 Bereich.
 - Alternativ kann auch nur die linke obere Ecke benannt werden.

Beispiel xlswrite

- In neue Excel-Datei schreiben
 - `M=magic(5);`
 - `xlswrite('Magic.xlsx',M)`
- Diese Datei erweitern: Nur Bereich, erste Arbeitsmappe
 - `xlswrite('Magic.xlsx',M,'','F6')`
- Anwendung:
Ergebnis von Berechnungen aus MATLAB in bestehende Excel-Tabelle schreiben.



The screenshot shows a Microsoft Excel spreadsheet titled 'Magic - Microsoft Excel'. The spreadsheet contains a 5x5 grid of numbers representing a magic square. The numbers are: Row 1: 8, 13, 1, 6, 10; Row 2: 14, 9, 7, 11, 5; Row 3: 2, 12, 15, 3, 10; Row 4: 11, 5, 14, 2, 9; Row 5: 13, 1, 12, 8, 4. The cell at row 5, column 1 (containing the value 13) is highlighted in yellow, indicating it is the active cell. The Excel ribbon is visible at the top, showing tabs for 'Start', 'Einfügen', 'Formatvorlagen', 'Layout', 'Übersicht', 'Ansicht', and 'Erweiterte Formeln'. The 'Layout' tab is currently selected. The status bar at the bottom shows 'Book1' and '100 %'.

| | | | | |
|----|----|----|----|----|
| 1 | 8 | 13 | | |
| 2 | 14 | 9 | | |
| 3 | 2 | 12 | 15 | 3 |
| 4 | 11 | 5 | 14 | 2 |
| 5 | 13 | 1 | 6 | 10 |
| 6 | | 17 | 24 | 1 |
| 7 | | 23 | 5 | 7 |
| 8 | | 4 | 6 | 11 |
| 9 | | 10 | 12 | 13 |
| 10 | | 11 | 18 | 25 |
| 11 | | 19 | 2 | 8 |
| 12 | | 21 | 16 | 10 |
| 13 | | 22 | 13 | 1 |

MATLAB-Dateiformat

- MATLAB hat ein eigenes Dateiformat zum Speichern von Variablen.
- Vorteile:
 - In einer Datei können verschiedenen Variablen gleichzeitig gespeichert werden.
 - Kein Präzisionsverlust beim Speichern von Dezimalzahlen
 - Austausch der Dateien zwischen Betriebssystemen möglich.
 - Komprimiertes Dateiformat
- Nachteil:
 - Kann nur MATLAB lesen.

Speichern im MATLAB-Format

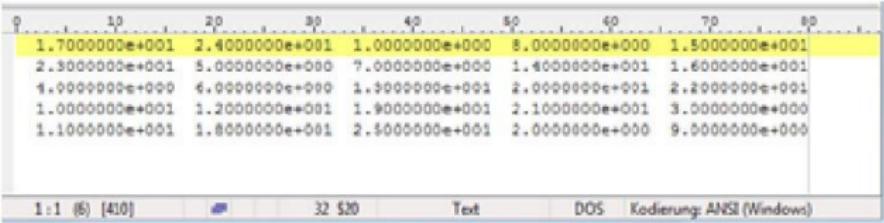
- Befehl zum Speichern im MATLAB Format:
 - `save <Dateiname> <Var1> <Var2> <Var3> ...`
 - `save ('<Dateiname>', '<Var1>', ...)`
 - Im ersten Fall kann Dateiname keine Variable mit dem Dateinamen sein!
- Beispiel:
 - `M=magic(5);`
 - `a=1;`
 - `save Daten M a` oder `save('Daten', 'M', 'a')`
- Datendateien von MATLAB haben *.mat* als Dateierweiterung

Laden von MATLAB-Datendateien

- Es gibt zwei Möglichkeiten, um alle Variablen aus einer MATLAB Datei zu laden:
 - Doppelklick im Fenster *Current Directory* auf den Dateinamen.
 - Den Befehl `load` verwenden.
- Beispiel:
Laden der eben beispielhaft gespeicherten Datei
 - `load Daten` oder `load ('Daten')`
 - Damit werden die Variablen `M` und `a` in den `Workspace` geladen.
- `load Daten a`: Lädt nur die Variable `a`, nicht `M`.

Textdateien

- Die Befehle `save` und `load` können auch Textdateien speichern und lesen, indem der Parameter `-ascii` übergeben wird.
 - Beispiel:
 - `M=magic(5);`
 - `save M.txt M -ascii`
- Ergebnis:



A screenshot of a text editor window showing a 5x5 matrix of numbers in scientific notation. The matrix is as follows:

| | | | | |
|---------------|---------------|---------------|---------------|---------------|
| 1.700000e+001 | 2.400000e+001 | 1.000000e+000 | 8.000000e+000 | 1.500000e+001 |
| 2.300000e+001 | 5.000000e+000 | 7.000000e+000 | 1.400000e+001 | 1.600000e+001 |
| 4.000000e+000 | 4.000000e+000 | 1.300000e+001 | 2.000000e+001 | 2.200000e+001 |
| 1.000000e+001 | 1.200000e+001 | 1.900000e+001 | 2.100000e+001 | 3.000000e+000 |
| 1.100000e+001 | 1.800000e+001 | 2.500000e+001 | 2.000000e+000 | 9.000000e+000 |

The table is displayed in a text editor window with the following status bar information: 1:1 (6) [410], 32 \$20, Text, DOS, Kodierung: ANSI (Windows).

- Laden: `X=load('M.txt', '-ascii');`

Textdateien

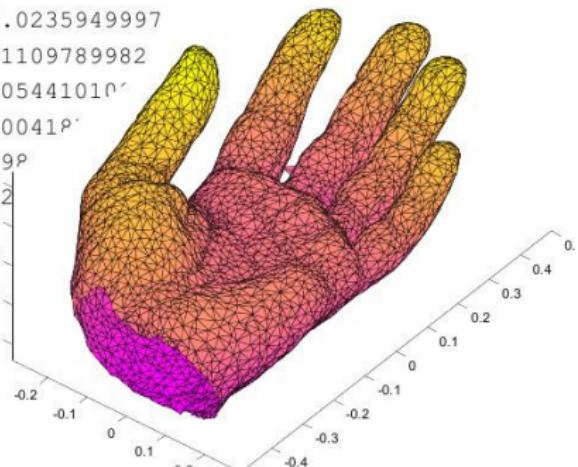
- Es existiert kein allgemeines Format zum Speichern von „Daten“.
- Häufig muss man jedoch exportierte Daten von anderen Programm weiterverarbeiten.
- Weiteres Problem:
 - Vielleicht braucht man nicht alle Daten
 - Viel zu viele Daten (passt nicht in Hauptspeicher)
- Lösung:
 - Zeilenweise die Datei einlesen und weiterverarbeiten.

Beispiel

- Es sei folgende Datei gegeben (Auszug):

```
# 3D-Punktwolke von Laserscanner
# Format: x y z Koordinate
# Stand: 14.05.2009
-0.1784259975 -0.2726149857 0.1257269979
-0.1590040028 -0.0372560993 -0.0235949997
-0.1081809998 -0.2712480128 0.1109789982
0.0074431300 -0.2806249857 0.005441010
0.0521670990 -0.2792350054 -0.00418
-0.0311114993 0.4988600016 0.09
-0.2406609952 -0.0318467990 0.2
```

- Die Datei kann nicht mit `load` gelesen werden, weil Kommentarzeilen vorhanden sind.



Öffnen einer Datei

- Mit dem Befehl `fopen (<Dateiname>)` wird eine Datei zum Lesen geöffnet.
 - Der Befehl gibt ein so genanntes *File-Handle* zurück. Das ist eine eindeutige Kennung für diese Datei.
 - Es können beliebig viele Dateien gleichzeitig geöffnet werden. Jede hat ein eigenes *File-Handle*.
 - Wenn die Datei nicht geöffnet werden kann, dann wird der Wert -1 zurückgegeben.
- Beispiel:

```
filename= 'Hand.txt';
fid=fopen (filename);
```

Schließen einer Datei

- Nach dem letzten Zugriff muss die Datei wieder geschlossen werden.
- Wenn die Datei nicht geschlossen wird, ist sie vom Betriebssystem blockiert.
- Befehl: `fclose (<File-Handle>)`
- Beispiel: `fclose (fid); %fid aus Beispiel`
- ACHTUNG: Wenn eine Datei nicht geöffnet wurde, dann kann sie auch nicht geschlossen werden.
 - Besser: `if (fid ~= -1) ...`

Lesen einer Zeile aus einer Datei

- Die Daten einer Zeile können mit dem Befehl `fgetl (<File-Handle>)` gelesen werden.
 - Der Befehl gibt eine Zeichenkette zurück, wenn die Zeile erfolgreich gelesen werden konnte.
 - Es wird immer die nächste Zeile gelesen.
 - Es wird eine -1 zurückgegeben, wenn das Ende der Datei erreicht wurde.
- Beispiel: `Zeile=fgetl(fid);`

Beispielprogramm: Zeilen zählen

```
filename= 'Hand.txt';
n=0;                                % Zähler für Zeilen
fid=fopen(filename);      % Datei öffnen

tline = fgetl(fid);      % Erste Zeile lesen

% Schleife, solange keine -1 zurückgegeben wird
while tline ~= -1
    n=n+1;                      % Zeilenzähler +1
    tline = fgetl(fid);      % Noch eine Zeile lesen
end

fclose(fid);                      % Datei wieder schließen
disp(n);                          % Zeilenzahl ausgeben
```

Beispielprogramm: Zeilen zählen

```
filename= 'Hand.txt';
n=0;                                % Zähler für Zeilen
fid=fopen(filename);      % Datei öffnen

if fid ~= -1                      % Nur wenn Datei offen
    tline = fgetl(fid);
    while tline ~= -1
        ...
    end
    fclose(fid);
else                                % Fehlermeldung
    disp('Datei nicht vorhanden');
end
```

Beispiel: Kommentare überspringen

```
while tline ~= -1 %ischar(tline)
    % Nur zählen, wenn erstes Zeichen keine
    % Raute ist.
    if tline(1) ~= '#'
        n=n+1;
    end
    tline = fgetl(fid);
end

% Ausgabe
disp(sprintf('Die Datei %s hat %i
Koordinaten',filename,n));
```

Umwandeln in Zahl

- Der Befehl `sscanf(<Variable>,<Format>)` wandelt Daten aus einer Zeichenkette um.
 - Beide Parameter sind Zeichenketten:
 - `<Variable>`: Hier stehen die Daten drin
 - `<Format>`: Schablone für die Erkennung.
- Für einfache Fälle von Zahlen genügt `%g` als Platzhalter für Zahlen.
- Mit `%g*` können Zahlen ignoriert werden.
- Beispiel:
 - `xyz=sscanf(tline,'%g %g %g');`
 - `xz=sscanf(tline,'%g %g* %g');`