

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



¹Die 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

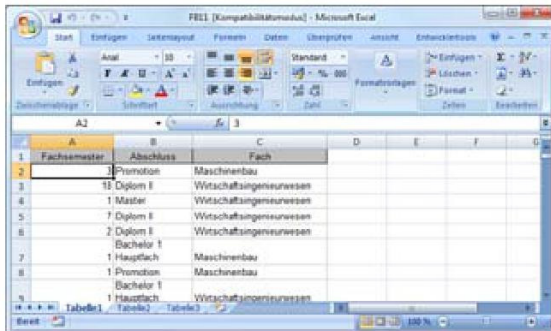
- MATLAB unterstützt mehrere Dateiformate (lesen und schreiben)
- In dieser Vorlesung behandelt werden
 - Excel-Dateiformat
 - MATLAB-Dateiformat
 - Beliebige Textdateien
 - Mit einem Befehl
 - Zeilenweise mit Programmierung
- Alle Dateiformate: `doc fileformats`

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



	A	B	C	D	E	F	G	H
1	Fachsemester	Abschluss	Fach					
2		1 Promotion	Maschinenbau					
3		15 Diplom II	Wirtschaftsingenieurwesen					
4		1 Master	Wirtschaftsingenieurwesen					
5		7 Diplom II	Wirtschaftsingenieurwesen					
6		2 Diplom II	Wirtschaftsingenieurwesen					
7		Bachelor I						
8		1 Hauptfach	Maschinenbau					
9		1 Promotion	Maschinenbau					
10		Bachelor I						
11		1 Hauptfach	Wirtschaftsingenieurwesen					

Einlesen der Tabelle

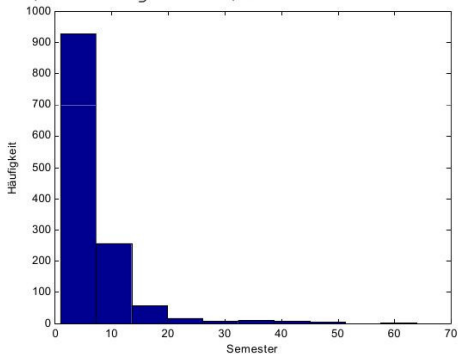
- MATLAB Befehl:
 - `<Variable> = xlsread(<Dateiname>)`
- Beispiel: `SemesterAnzahl=xlsread('FB11.xls');`
- Die Rückgabeveriable ist eine $n \times m$ Matrix
 - n : Anzahl der Zeilen mit numerischen Werten
 - m : Spalten der Excel-Tabelle
- Beispiel: `size(SemesterAnzahl)`
 - `[1289, 1]`
 - Stehender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

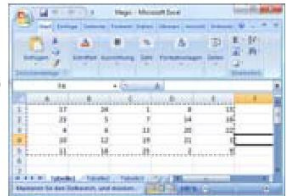
- `hist(SemesterAnzahl)`
- `xlabel('Semester');`
- `ylabel('Häufigkeit');`



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)`

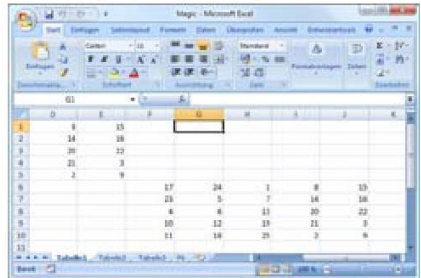


xlswrite

- 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.



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

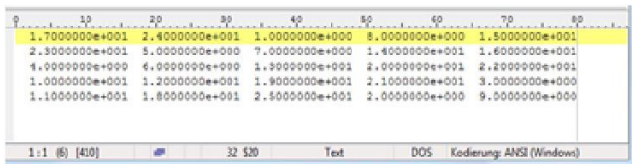
- Befehl zum Speichern im MATLAB Format:
 - `save <Dateiname> <Var1> <Var2> <Var3> ...`
 - `save('<Dateiname>', '<Var1>', ...)`
 - Im ersten Fall kann Dateiname keine Variable mit dem Dateinamen sein!
- Beispiel:
 - `M=magic(5);`
 - `a=1;`
 - `save Daten M a` oder `save('Daten', 'M', 'a')`
- Datendateien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



The image shows a MATLAB editor window displaying a 5x5 matrix. The matrix is a magic square, where each row and column contains the numbers 1 through 5 exactly once. The values are displayed in scientific notation (e.g., 1.0000000e+001). The first row is highlighted in yellow. The status bar at the bottom indicates the file is named '1:1 (6) [410]', is 32 lines long, and is saved as a text file in DOS format with ANSI encoding.

1.0000000e+001	2.4000000e+001	1.0000000e+000	5.0000000e+000	1.5000000e+001
2.3000000e+001	5.0000000e+000	7.0000000e+000	1.4000000e+001	1.6000000e+001
4.0000000e+000	6.0000000e+000	1.3000000e+001	2.0000000e+001	2.2000000e+001
1.0000000e+001	1.2000000e+001	1.9000000e+001	2.1000000e+001	3.0000000e+000
1.1000000e+001	1.8000000e+001	2.5000000e+001	2.0000000e+000	9.0000000e+000

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

Textdateien

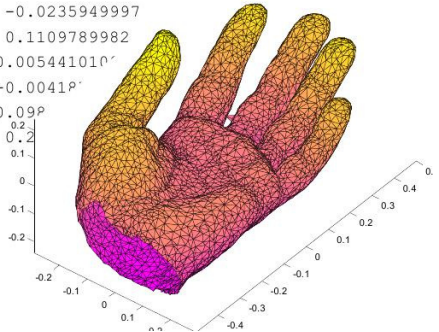
- Es existiert kein allgemeines Format zum Speichern von „Daten“.
- Häufig muss man jedoch exportierte Daten von anderen Programm weiterverarbeiten.
- Weiteres Problem:
 - Vielleicht braucht man nicht alle Daten
 - Viel zu viele Daten (passt nicht in Hauptspeicher)
- Lösung:
 - Zeilenweise die Datei 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 0.0054410100  
0.0521670990 -0.2792350054 -0.0041800000  
-0.0311114993 0.4988600016 0.0990000000  
-0.2406609952 -0.0318467990 0.2000000000
```

- 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');`