10 Lesen und Schreiben von Dateien

10.1 Mit `load` und `save`

Binäre Dateien

Mit `save` können Variableninhalte binär im Matlab-Format abgespeichert werden.

Syntax: `save` *Dateiname Variable1 Variable2 ...*

Mit `save` *Dateiname* werden alle Variablen abgespeichert. Der Dateiname sollte die Endung `.mat` haben (für binäre Dateien).

Beispiel: `save koordinaten.mat x y`

Mit `load` können die obigen Variablen wieder geladen werden:

`load koordinaten.mat`.

Textdateien

Mit

`save` *Dateiname Variable* `-ascii`

kann der Inhalt einer einzigen Variablen (nur Matrizen und Vektoren) als Textdatei abgespeichert werden.

Beispiel: `save xkoordinaten.txt x -ascii`

Hier sollte die Endung `.mat` nicht verwendet werden, da Matlab die Datei sonst als binäre Datei interpretiert.

Mit `load xkoordinaten.txt` kann der Inhalt der Datei xkoordinaten.txt wieder geladen werden. Allerdings wird hier eine Variable `xkoordinaten` erzeugt, da in der Textdatei keine Information über den Variablennamen existiert.

Bessere Variante: `x = load('xkoordinaten.txt')`. Dann darf das Argument von `load` auch eine String-Variable sein, also

```
file = 'xkoordinaten.txt';
x = load(file);
```

Bemerkung:

Befehl Arg1 Arg2 entspricht *Befehl*`('`*Arg1*`','`*Arg2*`')`.

Format der Textdatei:

Selbstgeschriebene oder von anderen Programmen erzeugte Textdateien dürfen nur Zahlen in Form eines Vektors oder einer Matrix enthalten. Für `load` müssen die Werte mit Leerzeichen getrennt sein. Falls andere Trennzeichen verwendet werden, muss `dlmread` verwendet werden, z.B. `x = dlmread('xkoordinaten.txt',',')`, falls die Zahlen durch Beistriche getrennt sind. Siehe auch `dlmwrite`.

10.2 Mit `fscanf` und `fprintf`

Öffnen von Dateien, `fopen`

fid = `fopen(`*file*`,'r')` öffnet die Datei *file* zum Lesen.

fid = `fopen(`*file*`,'w')` öffnet die Datei *file* zum Schreiben (neu).

fid = `fopen(`*file*`,'a')` öffnet die Datei *file* zum Schreiben (anhängen).

fid ist ein Datei-Zeiger und wird für das Lesen und Schreiben benötigt.

fid = `1` Datei-Zeiger für Bildschirmausgabe.

Falls *fid* gleich `-1` ist, konnte die Datei nicht geöffnet werden.

Weitere Möglichkeiten: Siehe `doc fopen`.

Textmodus unter Windows:

Damit die Zeilenumbrüche richtig sind, ist zusätzlich noch ein `t` anzugeben, also `wt`.

Schreiben von Dateien, `fprintf`

Syntax: `fprintf(`*fid*`,`*format*`,`*Variablen*`)`

```
fprintf(1,'log10(%3.1f)=%8.4f\n',[.1:.1:1;log10(.1:.1:1)])
```

ergibt

```
log10(0.1)= -1.0000
log10(0.2)= -0.6990
log10(0.3)= -0.5229
log10(0.4)= -0.3979
log10(0.5)= -0.3010
log10(0.6)= -0.2218
log10(0.7)= -0.1549
log10(0.8)= -0.0969
log10(0.9)= -0.0458
log10(1.0)=  0.0000
```

Bedeutung der Argumente:

`1`	Ausgabe am Bildschirm
`%3.1f`	Fixpunktdarstellung, mindestens 3 Zeichen insgesamt, 1 Nachkommastelle
`%8.4f`	Fixpunktdarstellung, mindestens 8 Zeichen insgesamt, 4 Nachkommastellen
`\n`	Zeilenumbruch

`fprintf` gibt jeweils zwei Werte pro Zeile aus bis der Inhalt des Feldes aufgebraucht ist. Spaltenweises Durchzählen!

Zwei weitere wichtige Format-Bezeichner:

`%d` ganze Zahlen, `%4d` ganze Zahlen (mind. 4 Zeichen),

`%s` String.

Genaueres über Format-Bezeichner: Siehe `doc fprintf`.

Anführungszeichen ' in einem String:

```
'Das ist ein Anführungszeichen '' in einem String'
```

Es sind zwei Hochkommas notwendig.

Backslash \ in einem String:

```
'Das ist ein Backslash \\ in einem String'
```

Bei Verwendung mehrerer Variablen ist eine Schleife notwendig:

```
x = .1:.1:1;
y = log10(x);
for i=1:length(x)
  fprintf(1,'log10(%3.1f)=%8.4f\n',x(i),y(i))
end
```

Das Ergebnis ist dann wie oben.

Schließen von Dateien, `fclose`

Syntax: `fclose(`*fid*`)`

Dateien werden erst nach dem Schließen fertig geschrieben!

Lesen von Dateien, `fscanf`

Syntax:

A `= fscanf(`*fid*`,`*format*`)`

A `= fscanf(`*fid*`,`*format*`,`n`)`

A `= fscanf(`*fid*`,`*format*`,[`m`,`n`])`

`fscanf` liest Daten aus einer Datei mit Datei-Zeiger *fid* entsprechend dem Format *format*. In `[`m`,`n`]` kann n auch `inf` sein, falls die Zahl der Zeilen (!) in der Datei nicht bekannt ist.

Für ein Beispiel verwenden wir eine Datei `matrix.txt` mit dem Inhalt

```
     1     2     3
     4     5     6
     7     8     9
    10    11    12
```

```
>> fin=fopen('matrix.txt','r');
>> A=fscanf(fin,'%f',[3,inf])
A =
     1     4     7    10
     2     5     8    11
     3     6     9    12
>> A = A'    % in der Matrix spaltenweises Durchzaehlen,
A =          % daher transponieren
     1     2     3
     4     5     6
     7     8     9
    10    11    12
>> fclose(fin)
```

sscanf und sprintf

Der Befehl sscanf liest aus einem String, der Befehl sprintf schreibt in einen String.

Syntax:

A = sscanf(*str*,*format*) u.s.w.

str = sprintf(*format*, *Variablen*)

```
>> str = sprintf('Die Zahl pi = %10.6f',pi)
str =
Die Zahl pi =   3.141593
```

```
>> zahlen = sscanf('1 2 3 4','%f')'
zahlen =
     1     2     3     4
```

Zeilenweises Einlesen mit fgetl

```
fin = fopen('matrix.txt','r');

c = 1;
while (~feof(fin))
  L = fgetl(fin)
  A(c,:) = sscanf(L,'%f')';
  c = c+1;
end
A
```

feof(fid) liefert 1, falls das Dateiende erreicht ist, und 0 sonst.

Lesen mit textread

Mit textread kann etwas Tabellenförmiges eingelesen werden. Die einzelnen Spalten dürfen unterschiedliche Datentypen enthalten. In der ersten Outputvariable steht der Inhalt der ersten Spalte u.s.w.

Strings werden in Cell-Arrays abgespeichert.

Syntax:

```
[A,B,C,...] = textread(filename,format)
[A,B,C,...] = textread(filename,format,N)
[...] = textread(...,'param','value',...)
```

`N` ist die Anzahl der Zeilen, die gelesen werden sollen.

Wir lesen den Inhalt

```
as 1 2
wer 10 20
ac 100 200.13
```

der Datei `test.txt`:

```
[str,x,y]=textread('test.txt','%s %f %f') % String Zahl Zahl
str =
    'as'
    'wer'
    'ac'
x =
     1
    10
   100
y =
    2.0000
   20.0000
  200.1300
```

Wie kann so ein Inhalt gelesen werden?

```
as 1 2
wer xxx 10 20
a b c 100 200.13
```

Wir sind nur am Text und an der letzten Spalte interessiert.

```
>> [str,y]=textread('test.txt','%[^0123456789] %*f %f')
                    % keine Zahlen, ignoriere Spalte mit Zahl, Zahl
str =
    'as '
    'wer xxx '
    'a b c '
y =
    2.0000
   20.0000
  200.1300
>> str = deblank(str)
str =
    'as'
    'wer xxx'
    'a b c'
```

Der * in %*f bewirkt, dass die Werte in der ersten Spalte mit Zahlen gelesen, aber in keine Variable geschrieben werden.

Der Befehl `deblank` entfernt Leerzeichen am Ende eines Strings.

Alle Zeilen eines Textes in ein Cell-Array einlesen:

```
L = textread('test.txt','%s','delimiter','\n','whitespace','');
```

Oder nur die ersten zwei Zeilen:

```
>> L = textread('test.txt','%s',2,'delimiter','\n','whitespace','')
L =
    'as 1 2'
    'wer xxx 10 20'
```