Prof. Dr. Angela Kunoth
Max Glembock, Laslo Hunhold

Matlab Präsenzaufgaben — WS 2017/2018

Übungsblatt 8

Ausgabe: 06.12.2017

Speichern und Laden von Variablen

Oft ist es wünschenswert, die aktuell im Workspace erstellten Variablen zu speichern. Gerade bei aufwendigen numerischen Simulationen, deren Ergebnisermittlung einiges an Rechenzeit erfordert, ist dies sinnvoll.
Mit der Funktion `save` werden Variablen aus dem Workspace von Matlab in einer Datei gespeichert. Wird der Befehl `save` ohne weitere Angaben verwendet, so wird eine Datei `matlab.mat` im aktuellen Verzeichnis erzeugt, in der alle Variablen des Workspace gespeichert werden. Mit dem Befehl `load` ohne weiteren Zusatz wird nach der Datei `matlab.mat` gesucht und – sofern gefunden – werden alle darin gespeicherten Variablen wieder in den Workspace geladen. Meist wird man aber einen eigenen Namen für die Datei angeben wollen, in der man die Variablen speichert. Manchmal möchte man sogar gar nicht alle, sondern nur eine konkrete Auswahl an Variablen speichern. Die allgemeine Syntax von `save` lautet daher:

```
save(filename, variable1, ..., variableN);
```

Die Angabe der Variablen muss hier in Strings erfolgen, ist jedoch optional. Eine alternative Syntax bietet `save filename variable1 ... variableN`. Für diese Syntax ist es nicht notwendig, die Argumente als String anzugeben. Möchte man die gespeicherten Variablen laden, so geht dies durch:

```
load(filename, variable1, ..., variableN);
```

Diese Syntax sowie die Alternative ist analog zu der von `save`. Der Dateiname `filename` kann für `save` und `load` auch eine Pfadbezeichnung sein, wenn nicht in das aktuelle Verzeichnis geschrieben werden soll bzw. wenn die zu ladende Datei in einem anderen Verzeichnis liegt.

Aufgabe 31 (Speichern und Laden von Variablen)
Initialisieren Sie die Variablen `x1=3`, `x2=4`, `d=[0 2 3; 7 8 -2]`, `I=speye(5)` und `str='Hallo Welt'`.

a) Speichern Sie alle Variablen unter einem Dateinamen Ihrer Wahl, löschen Sie Ihren Workspace über `clear all` und laden Sie anschließend alle Variablen wieder.
b) Speichern Sie die Variablen `x1`, `I` und `str` unter einem Dateinamen Ihrer Wahl. Löschen Sie Ihren Workspace und laden Sie anschließend nur die Variable `I`.

Testen Sie in beiden Aufgabenteilen die verschiedenen Syntaxen.

Wichtig: Im Workspace bereits definierte Variablen werden beim Laden gleichnamiger Variablen ersetzt.

Speichern und Laden von Excel-Dateien

Für die Datenverarbeitung ist das Tabellenformat von Microsoft Excel in vielen Bereichen unumgänglich, da sehr viele Programme Daten in diesem Format import- und exportieren können. Aus diesem Grund bietet auch Matlab Schnittstellen für den Umgang mit Excel. Wir möchten zunächst das Schreiben von Matrizen in eine (neue) Excel-Datei erläutern. Matlab bietet dazu den Befehl `xlswrite`, der wie folgt aussieht:

```
xlswrite(filename, matrix, sheet, xlRange);
```

Hierbei steht `filename` wieder für den Namen oder Pfad der neu anzulegenden bzw. zu bearbeiteten Datei und muss als String eingegeben werden. `matrix` steht für eine Zahlenmatrix oder für ein Cell-Array. In einem Cell-Array können ebenfalls Strings eingetragen sein. `sheet` ist ein optionaler positiver Integer-Wert, der die zu bearbeitende Tabelle im Excel-Dokument auswählt. Alternativ kann auch ein String angegeben werden, falls die Tabellen eigene Namen haben. Standardmäßig wird immer die erste Tabelle gewählt. Ist die Tabelle unter der angegebenen Nummer bzw. dem angegeben String nicht zu finden, so erstellt Matlab eine neue Tabelle mit entsprechender Benennung. `xlRange` ist ein String und steht für den Zellenblock, in die die Matrix eingefügt werden soll, also bpsw. `'A4:C8'`. Standardmäßig wird immer die Zelle `'A1'` zum Einfügen der Tabelle ausgewählt. Ist die Matrix größer als die Menge der Felder in der `xlRange`, so werden auch nur die Matrixeinträge übernommen, die in die `xlRange` passen. Umgekehrt können Excel-Dateien mit dem Befehl

```
num = xlsread(filename, sheet, xlRange);
```

ausgelesen werden. Die Argumente sind hier entsprechend zu `xlswrite` zu interpretieren. Bei obigem Lesezugriff werden die Zahlenwerte von Matlab in der Matrix `num` gespeichert. Sind in der zu lesenden Datei

jedoch auch Strings enthalten, die in Matlab übertragen werden sollen, so muss der Befehl

```
[num, txt, raw] = xlsread(filename, sheet, xlRange);
```

verwendet werden. Hier werden in der Matrix `num` wieder die Zahlenwerte, im Cell-Array `txt` die String-Werte und im Cell-Array `raw` sowohl Zahlen- als auch String-Werte abgespeichert.

Aufgabe 32 (Speichern, Laden und Bearbeiten von Excel-Dateien)
Initialisieren Sie in Matlab das Cell-Array `A = {1 3, 'Hallo Welt'; 'Excel' -7.5 pi }`.

a) Erstellen Sie eine Excel-Datei `MeineExcelDatei`, in der Sie die neue Tabelle `MeineTabelle` anlegen und dort an Zelle `'D5'` das Cell-Array `A` einfügen. Schauen Sie sich anschließend die Tabelle an.
b) Löschen Sie Ihren Workspace und laden Sie aus Ihrer erstellten Excel-Datei die Zelleinträge `'D5:E6'`. Laden Sie sowohl die Zahlenwerte als auch die String-Werte.

Öffnen und Lesen von weiteren Datentypen

Matlab kann nicht nur eigene Dateitypen sowie Excel-Dateien öffnen und bearbeiten, sondern bietet auch eine ganze Reihe an I/O-Funktionen für den Umgang mit weiteren Datentypen. Einfache I/O-Funktionen werden von Matlab durch die `f`-Befehle bereit gestellt, die wir im Folgenden kurz vorstellen möchten.
Durch den Befehl

```
fileID = fopen(filename, permission);
```

kann Matlab intern eine Datei öffnen bzw. neu anlegen, um diese zu lesen oder zu bearbeiten. Unter `filename` kann wieder der Name der zu öffnenden Datei (mit Dateiendung) oder ein Pfad eingetragen werden. Das Argument `permission` ist ein String durch den man festlegt, mit welchen Absichten man die entsprechende Datei öffnet. Die gebräuchlichsten Argumente für `permission` sind:

`'r'`	Öffnet eine Datei, nur Leserechte
`'w'`	Öffnet oder erstellt eine Datei, nur Schreibrechte (bestehende Inhalte werden gelöscht, falls vorhanden)
`'a'`	Öffnet oder erstellt eine Datei, nur Schreibrechte (neue Daten werden am Ende der Datei hinzugefügt)

Standardmäßig ist `'r'` als `permission` festgelegt. In der `fileID` wird zur Identifikation der geöffneten Datei eine eindeutige Nummer (≥ 3) abgespeichert. Das Gegenstück zu `fopen` ist `fclose(fileID)`, womit die entsprechende Datei wieder geschlossen werden kann. Mit `fclose('all')` werden alle geöffneten Dateien geschlossen. Unter den Permissions `'w'` und `'a'` kann man mit folgendem Befehl in eine Datei schreiben:

```
fwrite(fileID, A, precision);
```

Hierbei steht `A` für eine Matrix bzw. einen String, welcher in die Datei geschrieben werden soll. `precision` steht für die Genauigkeit bzw. Speicherart mit der die Werte aus `A` in der Datei gespeichert werden sollen. Für Zahlenwerte ist standardmäßig ein vorzeichenloser 8-Bit-Integer (`'uint8'`) als Speicherart ausgewählt. Strings werden direkt als solche gespeichert. Die Werte für `precision` können in der Matlab-Dokumentation von `fwrite` eingesehen werden. Das Pendant zu `fwrite` ist `fread` mit der Syntax:

```
A = fread(fileID, sizeA, precision);
```

Hierbei werden die Werte aus der betrachteten Datei in einem Vektor `A` gespeichert. Optional kann über `sizeA` auch die Größe des Vektors eingestellt werden, so dass nur entsprechend viele Elemente in Matlab übertragen werden. Die `precision` gibt hierbei an, wie die Werte der betrachteten Datei zu interpretieren sind. Im Gegensatz zu der `precision` von `fwrite` gibt es hier mehr Einstellungen, da zusätzlich eine `precision` ausgewählt werden kann, mit der die Werte in den Vektor `A` geschrieben werden sollen. Standardmäßig ist `'uint8=>double'` eingestellt, was bedeutet, dass vorzeichenlose 8-Bit-Integer in das Double-Format umformatiert werden. Auch hier können die Werte für `precision` in der Matlab-Dokumentation von `fread` eingesehen werden.
Sowohl `fread` als auch `fwrite` nutzen interne Zähler zum Erhalt der aktuellen Zeichenposition innerhalb der geöffneten Datei. Sollten Sie also aus einer Datei bereits 8 Zeichen ausgelesen haben, so befindet sich dieser Zähler an Position 9. Der Zähler kann mit der Funktion `ftell` ausgelesen und mit `fseek` verändert werden.

Wichtig: Wenn eine Datei nur mit Leserechten geöffnet ist, können keine neuen Werte in ihr gespeichert werden. Schließen Sie die Datei daher zunächst und öffnen Sie sie anschließend mit Schreibrechten erneut.

Aufgabe 33 (Speichern, Laden und Bearbeiten von *.txt-Dateien)
Erstellen Sie eine `txt`-Datei mit Namen `MeinText.txt`.

a) Öffnen Sie Ihre `txt`-Datei und schreiben Sie den Vektor `C=[-100 100]` (zweimal) in diese mit der `precision` `'*int8'`. Lesen Sie nun die Datei ohne `precision`. Lesen Sie anschließend die Datei erneut mit der `precision` `'*int8'`.
b) Überschreiben Sie in Ihrer `txt`-Datei alles bisher Eingetragene mit dem String `'Hallo Welt'` (ebenfalls zweimal). Lesen Sie die Datei nun ohne `precision`. Lesen Sie anschließend die Datei erneut mit der `precision` `'*char'`.

Wie sind die Ergebnisse in beiden Aufgabenteilen zu erklären?