

Mathematik am Computer

7. Übung: Matlab, Teil IV

Marcus Grote und Helmut Harbrecht

Universität Basel

8.–12. November 2021

1 Matlab als Programmiersprache

- Die while-Schleife
- Beispiel zur Vorbereitung: Magische Quadrate
- Skripte und Funktionen
- Speicherverwaltung

2 Matlab-Programmierung

- Globale und Lokale Variablen

Die while-Schleife

```
while t < 1  
    Befehle  
end
```

- 1 Falls zu Beginn der while-Schleife $t < 1$ gilt, so werden alle Befehle zwischen `while` und `end` ausgeführt.
- 2 Es wird nun wieder geprüft, ob immer noch $t < 1$ gilt. Falls ja, so werden wieder alle Befehle zwischen `while` und `end` ausgeführt.
- 3 Dies wiederholt sich solange bis $t < 1$ nicht erfüllt ist.

Die while-Schleife

Anmerkungen:

- Es muss t vor `while` $t < 1$ ein Wert zugewiesen sein.
- Gilt $t < 1$ am Anfang nicht, so werden alle Befehle übersprungen.
- Genauer: $t < 1$ liefert als Ergebnis entweder 0 (`false`) oder 1 (`true`). Die Befehle in der Schleife werden solange wiederholt ausgeführt, bis der Wert der logischen Abfrage $t < 1$ gleich 0 (`false`) ist.
- Gilt $t < 1$ immer, so bricht der Programmablauf niemals ab. Deswegen wird der Wert von t normalerweise innerhalb der Schleife geändert.
- Jede logische Abfrage ist erlaubt, also auch z.B.
`while t >= 0 && t < 1` oder `while t == 15`.

Die while-Schleife

Beispiel

INPUT

```
Sum = 0;
cnt = 0;
while Sum < 10
    if rand(1) > 0.5
        Sum = Sum + 1;
    else
        Sum = Sum - 1;
    end
    cnt = cnt + 1;
end
disp(['S: ' num2str(Sum)])
disp(['C: ' num2str(cnt)])
```

OUTPUT

```
S: 10
C: 292
```

```
S: 10
C: 5238
```

```
S: 10
C: 366
```

```
S: 10
C: 82
```

⋮

Magische Quadrate

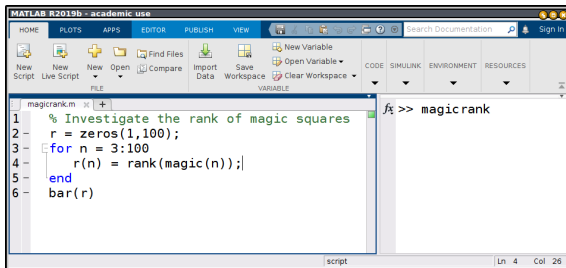
Eine $(n \times n)$ Matrix A ist ein **magisches Quadrat**, falls die Zeilen- bzw. Spaltensummen konstant sind, z.B.

$$A = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$$

`magic(n)` bestimmt ein $(n \times n)$ magisches Quadrat.

Skripte

Magische Quadrate



The screenshot shows the MATLAB R2019b interface. The main window is titled "MATLAB R2019b - academic use". The top menu bar includes HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. Below the menu bar is a toolbar with icons for New Script, New Live Script, New, Open, Compare, Import Data, Save Workspace, and Clear Workspace. The main editor area is split into two panes. The left pane shows a script named "magicrank.m" with the following code:

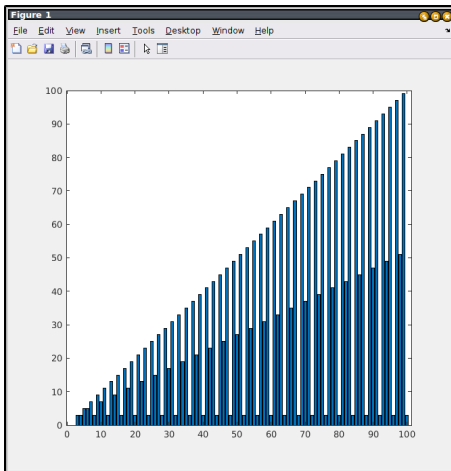
```
1 % Investigate the rank of magic squares
2 r = zeros(1,100);
3 for n = 3:100
4     r(n) = rank(magic(n));
5 end
6 bar(r)
```

The right pane is the Command Window, showing the command `magicrank` entered at the prompt `>>`. The status bar at the bottom indicates "script" and "Ln 4 Col 26".

Eingabe von `magicrank` im Command Window führt das Skript `magicrank.m` aus.

Skripte

Magische Quadrate: Ausgabe



Skripte und Funktionen

Einschränkungen

Skripte führen nach Aufruf die darin enthaltenen Befehle sequenziell aus. Dabei ist zu beachten:

Skripte haben keinen abgekapselten Workspace.

Die Variablen im Haupt-Workspace können durch ein Skript initialisiert, verändert und gelöscht werden.

Angenommen wir wollen rechnen, ohne die Werte im Haupt-Workspace zu verändern:

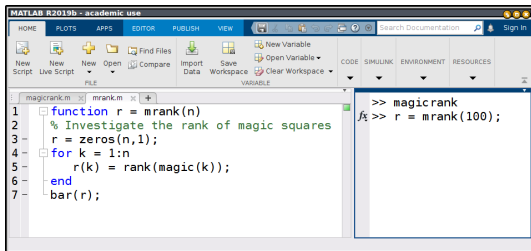
Wir brauchen also einen temporären Workspace

→ in den ausgewählte Parameter übergeben werden, und

← aus dem Werte und Ergebnisse zurückgenommen werden

Lösung: Funktionen.

Funktionen



```
MATLAB R2019b - academic use
HOME PLOTS APPS EDITOR PUBLISH VIEW Search Documentation Sign In
New Script New Live Script New Open Find Files Compare Import Data Save Workspace Clear Workspace
FILE VARIABLE
magicrank.m mrank.m
1 function r = mrank(n)
2 % Investigate the rank of magic squares
3 r = zeros(n,1);
4 for k = 1:n
5     r(k) = rank(magic(k));
6 end
7 bar(r);
>> magic rank
fx >> r = mrank(100);
```

Die Eingabe von `mrank(100)` liefert dasselbe Ergebnis.

Funktionen

- Dateiname und Funktionsname sollen übereinstimmen.
- Die Dateierweiterung ist `.m`.
- Alle Variablen die innerhalb einer Funktion bearbeitet werden, sind **lokal**, d.h. in einem temporären, abgekapselten Workspace.
- Die Funktion `function r = mrank(n)` bedarf eines Eingabeparameters und gibt eine Variable als Ergebnis zurück.
- Eingabe mehrerer Parameter und Rückgabe mehrerer Variablen ist möglich.

Beispiel:

```
function [V1 , V2, V3] = nFunktion(P1, P2)
```

Aufruf: `X=1; Y=2;`
`[A, B, C] = nFunktion(X, Y);`

Funktionen

Aufruf:

- 1 `mrnk(100)` führt die Funktion aus und gibt den Rückgabewert aus.
- 2 `mrnk(100) ;` führt die Funktion aus und gibt den Rückgabewert nicht aus.
- 3 `z = mrnk(100) ;` führt die Funktion aus und speichert den Rückgabewert in `z`.
- 4 `z = mrnk(100)` führt die Funktion aus, speichert den Rückgabewert in `z` und gibt ihn aus.

Globale Variablen

Variablen, die direkt im „Command Window“ oder durch eine Befehlsabfolge in einem Skript definiert werden, sind **global**.

Auf diese Variablen kann vom „Command Window“ und von jedem Skript aus zugegriffen werden.

Die Variablen und ihre Werte existieren so lange, bis diese gelöscht oder geändert werden.

Lokale Variablen

- Variablen in Funktionen sind **lokal**.

Sie sind nur innerhalb dieser Funktion bekannt.

Sobald der Funktionsaufruf endet, werden die Variablen und ihre Werte gelöscht.

- Funktionen kennen keine globalen Variablen, deswegen:
 - können Funktionen keine globalen Variablen modifizieren,
 - müssen alle benötigten Werte als **Parameter** übergeben werden,
 - müssen alle nach Ende des Funktionsaufruf weiter benötigten Werte als **Rückgabewerte** zurückgegeben werden.