Mathematik am Computer

10. Übung: Matlab, Teil VII

Marcus Grote und Helmut Harbrecht

Universität Basel

29. November – 3. Dezember 2021

Übersicht

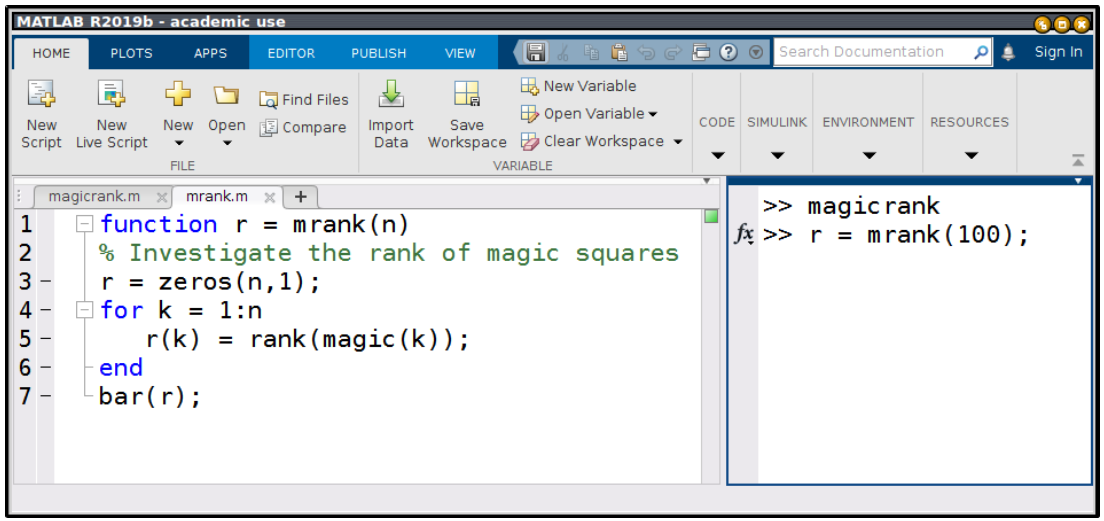
1. Wiederholung
   - Skripte und Funktionen
   - Steuerung
2. Matlab-Programmierung
   - Funktionen als Parameter
3. Matlab-Abbildungen speichern
   - Plot erstellen, speichern & in LaTeXanzeigen
   - Plot anpassen

Skript

magicrank.m

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

Funktionen



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

Verzweigung: if-elseif-else-end

```
if n == 1
      Befehle1
elseif n == 2
      Befehle2
elseif n < 1
      Befehle3
      ⋮
elseif n == 'A'
      BefehleN
else
      BefehleAlt
end
```

elseif und else mit den darauf folgenden Befehlen sind optional.

Die for-Schleife

Der Vektor `vec` hat die Länge `n`.

```
for k = vec
      Befehle
end
```

1. Zunächst ist $k = vec(1)$, d.h. der erste Wert des Vektors `vec`, und es werden alle Befehle zwischen `for` und `end` mit dem Wert $k = vec(1)$ ausgeführt.
2. Es wird $k = vec(2)$ gesetzt und alle Befehle zwischen `for` und `end` mit dem Wert $k = vec(2)$ ausgeführt, usw.
3. Es werden alle Werte von k durchlaufen, bis einschliesslich $k = vec(n)$.

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.

Mathematische Funktionen definieren

Wir haben eine Funktion f und wollen sie an den Stellen x in Matlab auswerten und die Ergebnisse plotten.

Damit die Funktion f gezeichnet wird, muss die Matlab-Implementierung `f` auf den Vektor `x` der x-Werte angewandt werden. Damit werden die Funktionswerte $y = f(x)$ erhalten.

Also muss `f` so implementiert sein, dass `x` in die Funktion eingesetzt werden kann.

Mathematische Funktionen definieren

Beispiel: Die Funktion

$$f(x) = \frac{1}{1+5x^2}, \qquad x \in \mathbb{R}$$

soll als Matlab-Funktion `f` implementiert werden:

`f.m`

```
function y = f(x)

eins = ones(size(x));
y = eins ./ (eins + 5 * x.^2);
```

Mathematische Funktionen definieren

Beispiel: Die Funktion

$$f(x) = \frac{1}{1+5x^2}, \qquad x \in \mathbb{R}$$

soll als Matlab-Funktion `f` implementiert werden:

`f.m`

Alternativ:

```
function y = f(x)
y = 1 ./ (1 + 5 * x.^2);
```

Funktionen als Parameter

Ein Verfahren hängt oft von einer Funktion f ab, auf die das Verfahren angewandt werden soll, z.B.

- Verfahren zur Bestimmung einer Nullstelle x^* mit $f(x^*) = 0$
- Verfahren zur Berechnung eines Integrals

$$\int_a^b f(x)\,\mathrm{d}x$$

- Spezielle graphische Methoden zur Visualisierung von f

Problem: Es können nur Variablen an Funktionen übergeben werden, nicht Funktionen selbst!

Funktionen als Parameter

Lösung: Mache aus der *Funktion* eine *Variable*:

Ist `f` eine Matlab-Funktion, so ist `@f` ein Verweis auf die Funktion `f`. Er heißt Handle (Griff) der Funktion `f`

Dieser Handle kann in einer Variablen abgespeichert werden und/oder als Wert an eine weitere Funktion übergeben werden.

Funktionen als Parameter

```
function y=g(x)
y = 1./(x.^2);
```

```
function S = p_summe(f,n,N)
S=0;
for i = n:N
    S = S + f(i);
end
```

Der Aufruf `p_summe(@g,1,100)` berechnet

$$\sum_{i=1}^{100} \frac{1}{i^2}.$$

Funktionen als Parameter

- Handles können in Variablen abgespeichert werden:

  `s = @sin` oder `handle_g = @g`

- Es können auch Handles direkt definiert und abgespeichert werden:

  ```
  s = @(x) 2*x+1;
  ```

  speichert die Funktion $x \mapsto 2x + 1$ in der Variablen `s` ab. Der Aufruf `s(3)` ergibt 7.

Plots erstellen & speichern

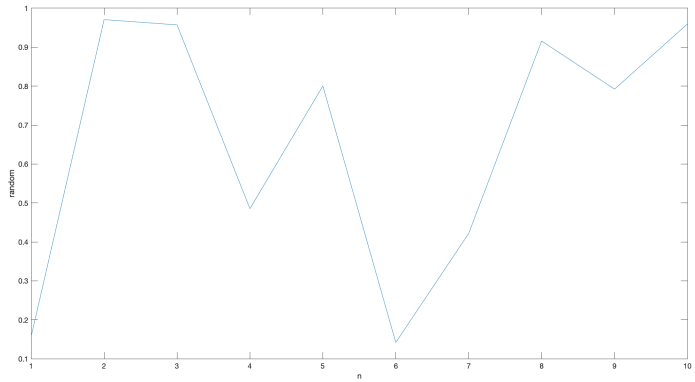
Mit dem Befehl `saveas` kann eine Matlab-`figure` abgespeichert werden.
Beispiel einer `figure` die als `.png`-Datei abgespeichert wird:

In Matlab:

```
figure(1)
plot(1:10,rand(1,10))
xlabel('n')
ylabel('random')
saveas(figure(1),'bild1.png')
```

In LaTeX:

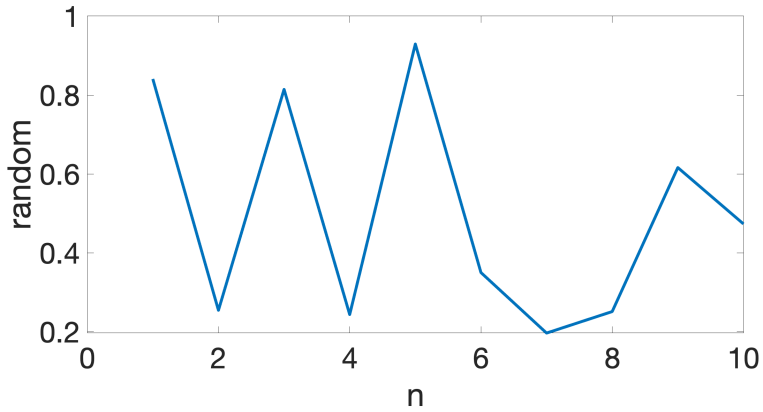
```
\begin{figure}[h]
\centering
\includegraphics
    [width=0.9\textwidth]
    {figures/bild1.png}
\end{figure}
```



Plot Optionen anpassen

Damit der Plot besser lesbar ist, können Plot-Optionen in Matlab angepasst werden.

```
figure(2)
plot(1:10,rand(1,10),
        'LineWidth',4.0)
set(gca,'FontSize',40)
xlabel('n')
ylabel('random')
saveas(figure(2),'bild2.png')
```



Wenn diese Optionen für mehrere Plots benötigt werden, dann können die Default-Werte angepasst werden:

```
set(0,'defaultLineLineWidth',2)
set(0,'defaultAxesFontSize',20)
```

Mit `get` statt `set` und ohne Angabe eines Zahlenwerts können diese Default-Werte abgerufen werden.