Einführung in das Programmierung mit Matlab, Teil 1/4

Thomas Dunst

Mathematisches Institut
Universität Tübingen

(e–mail: progtutor@na.uni-tuebingen.de)

09. Oktober 2012

Gliederung

Wer, Was, Wofür,...

Infos unter:
`http://na.uni-tuebingen.de`

- Zielgruppe: Programmieranfänger
- Erstellung kleiner gut lesbarer Programme in MATLAB
- Vorbereitung für Numerik - Programmieraufgaben

- kein vollständiger oder ausführlicher Programmierkurs
- kein Ersatz für Benutzung von Literatur bzw. Hilfefunktionen

..., Womit, Wie, Wann

- Programmiersprache/-umgebung: MATLAB
- Vormittags: Vermittlung der Theoretischen Grundlagen
  Zeit: 10-12 Uhr, N1
- Nachmittags: Theorie in Übungen anwenden
  Zeit: 13-15 Uhr, D02A38 (CIP-Pool, Computerraum)
  Übungsaufgaben lösen unter Anleitung

..., Womit, Wie, Wann

- Tag 1:
  - Grundlegender Syntax
  - Kontrollstrukturen
  - Einfache Operatoren
- Tag 2:
  - Besprechung Übungsblatt 1
  - Vektoren und Matrizen
  - Funktionen und Skripte
- Tag 3:
  - Besprechung Übungsblatt 2
  - Visualisierung
- Tag 4:
  - Besprechung Übungsblatt 3
  - Was tun bei Fehlern?
  - Allgemeines zum Programmierübungsbetrieb

Was bedeutet Programmieren?

Übersetzung von Menschensprache in Computersprache

Analog zu Übersetzungen:

- Der Sinn der Anweisungen in Menschensprache muss dazu nicht notwendigerweise verstanden werden!
- Aber: Das Wissen darum schadet nicht.
- Hingegen: (Minimale) Programmierkenntnisse sind notwendig!

Unterschiede:
Beim Programmieren

- muss man sich vorher überlegen, was alles passieren kann (z.B. Div. durch Null, sonst bricht Programm ab).
- muss man genau ausdrücken, was man will. Ungenaue Umgangssprache reicht nicht.

Ein Beispiel I

Berechnung von $S = \sum_{i=0}^{n} \frac{(-1)^i}{(2i+1)}$ für $n = 5$.

```
1  n = 5;
2  S = 0;
3  for i = 0 : n
4    S = S + (-1) ^ i / (2 * i + 1)
5  end
6
7  disp('Ergebnis: S = '); disp(S);
```

```
 S =  1
 S =  0.66667
 S =  0.86667
 S =  0.72381
 S =  0.83492
 S =  0.74401
```

```
Ergebnis: S =
 0.74401
```

Ein Beispiel I (Mit Kommentar)

Berechnung von $S = \sum_{i=0}^{n} \frac{(-1)^i}{(2i+1)}$ für $n = 5$.

```
1  n = 5;
2  S = 0; % Startwert fuer Summation
3  for i = 0 : n
4    S = S + (-1) ^ i / (2 * i + 1)
5  end
6  % Ausgabe des Ergebnisses:
7  disp('Ergebnis: S = '); disp(S);
```

```
S =  1
S =  0.66667
S =  0.86667
S =  0.72381
S =  0.83492
S =  0.74401
```

```
Ergebnis: S =
 0.74401
```

MATLAB ist:

- ein Software-Paket zur numerischen Lösung mathematischer Probleme.
- eine Programmiersprache, in der der Lösungsalgorithmus für das Problem formuliert wird.
- Vorbild für das weitgehend kompatibele Open Source Software-Paket Octave.

Alle in diesem Kurs vorgestellten Programme und Sprachelemente liegen in der Schnittmenge von MATLAB und Octave.
Aufruf von MATLAB:

- interaktiv: Eingabe von `matlab` in einem Terminalfenster.
- nicht-interaktiv: Eingabe von `matlab skript` in einem Terminalfenster, wobei `skript` der Name der Skriptdatei ist.

Das Softwarepaket MATLAB umfasst

- eine (grafische) Oberfläche, welche wiederum:
- einen Text-Editor (zur Bearbeitung von MATLAB-Skripten)
- ein Kommando-Fenster (zur Ein- und Ausgabe von Text)
- eine umfangreiche Hilfefunktion (...)
- einen Dateninspector (zum Ansehen der aktuellen Daten)
- einen Debugger (zur Fehlersuche im Skript)

Die Programmiersprache MATLAB umfasst:

- Daten (Skalare, Vektoren, Matrizen)
- Anweisungen
  - Operatoren (arithmetische Op. z.B. +,-, relationale Op. <,>)
  - Funktionen (sin, Ausgabe von Variablen, eigene Funktionen)
  - Kontrollanweisungen (Schleifen, Verzweigungen)
- Kommentare (zur besseren Lesbarkeit)

Erste Hilfe:

- Hilfe in Menuleiste wählen (nur MATLAB)
- Eingabe von `help` bzw. `doc` im Kommando-Fenster führt auf die Hilfeseiten von MATLAB bzw. Octave (info-Hilfe).
- MATLAB Anleitung:
  `http://www.mathworks.com/help/techdoc/`

$\Rightarrow$ Tipp: Bei Problemen im Internet suchen!
(Stichwort matlab + gesuchter Begriff)

Grundlegende Syntax von MATLAB I

- Variablen: sind Zeichenfolgen ohne `+ - * / %`, die keine Syntax-Elemente sind, z.B.

```
funktionswert = 2 * andererwert;
a = b + c;
```

- Kommentar ist alles hinter einem Prozentzeichen (%)

```
% Das ist ein Kommentar
a = 5; % Das ist auch ein Kommentar
```

- Vordefinierte Konstanten, z.B. pi, i, eps
- Strings (Zeichenketten) werden immer mit einfachen Anführungszeichen (') umschlossen, z.B.

```
string = 'Dies ist ein String';
```

Grundlegende Syntax von MATLAB II

- Abschließen eines Ausdrucks und Unterdrücken von Ausgaben mittels Semikolon (;)

```
a = 1 + 1  b = 2 * a  % Fehler, ein Ausdruck
a = 1 + 1; b = 2 * a; % Richtig, zwei Ausdrücke
a = 1 + 1 % Ausgabe: a = 2
b = 3 * 3; % keine Ausgabe
```

- Fortgeschritten: Code-Zeilen auf mehrere Zeilen aufteilen mit dem 3-Punkt-Operator (...)

```
f = 1 + 2 + 3 + 4 + ...
5 + 6 + 7 + 8;
```

Operatoren in MATLAB

- Zuweisungsoperator: `=`

  `a = 3;  b = c;`

- Arithmetische Operatoren: `+ - * / ^`

  `3 + 2;  a - b;  x = 2^3;`

- relationale Operatoren: `== ~= > >= < <=`

  `3 == 3` Ausgabe: 1

  `x = 3; x ~= 3` Ausgabe: 0

- logische Operatoren: `& | ~` (and, or, not)

  `x = 3`

  `(x > 1) & (x < 5)` Ausgabe: 1

(Das Ergebnis einer relationalen und logischen Operation ist entweder: wahr (=1) oder falsch: (=0). Solche Operationen werden oft in if-Abfragen verwendet

if-Abfrage

if *logischer Ausdruck*
    Anweisung(en)
elseif *logischer Ausdruck*
    Anweisung(en)
⋮
else
    Anweisung(en)
end

Bsp. 1

```
1  if x ~= 0
2    a = 5 / x
3  end
```

Bsp. 2

```
1  % x in Intervall [0,20]?
2  x = 17;
3  if (x<=20) & (x>=0)
4       disp('In [0,20]')
5       if x>=10
6            disp('Zweistellig');
7       else
8            disp('Einstellig');
9       end
10 elseif x>20
11      disp('Zu gross');
12 else
13      disp('Zu klein');
14 end
```

while-Schleife

while *logischer Ausdruck*
    Anweisung(en)
end

Bsp:

```
 1  % Was macht das Programm?
 2  x = 1;
 3  n = 0;
 4  while x > eps
 5     x = x / 2;
 6     n = n + 1;
 7  end
 8  x
 9  n
10  % eps ist Maschinen-
11  % genauigkeit
12  eps
```

Ausgabe:

```
x =  2.2204e-16
n =  52
ans =  2.2204e-16
```

for-Schleife

for *Variable* = *Startwert* : *Inkrement* : *Endwert*
    Anweisung(en)
end

Bsp:

```
1  % berechnet n!
2  x = 1;
3  n = 5;
4  for i = n:-1:1
5     x = x * i
6  end
```

Ausgabe:

```
x =  5
x =  20
x =  60
x =  120
x =  120
```

for-Schleife

for *Variable* = *Startwert* : *Inkrement* : *Endwert*
    Anweisung(en)
end

Bsp:

```
1  % berechnet n!
2  % Effizienterer Code
3  n = 5;
4  x = n;
5  for i = (n-1):-1:2
6     x = x * i
7  end
```

Ausgabe:

```
x =  20
x =  60
x =  120
```

Ähnlichkeiten zwischen while- und for-Schleife

Eine while-Schleife ist allgemeiner: Es gilt

for-Schleife $\Rightarrow$ while-Schleife

aber nicht immer

for-Schleife $\Leftarrow$ while-Schleife

Äquivalente Schleifen:

```
1  % for-Schleife: einfacher, schneller
2  for n = 4 : 2 : 10
3    n
4  end
5
6  % while-Schleife: flexibler, aufwendiger
7  n = 4 % vorher setzen
8  while n < 10
9    n = n + 2  % Selbst imkrementieren
10 end
```

Verwende jedoch for-Schleifen, wenn davor klar ist, wieoft iteriert wird

break und continue in Schleifen

Die beiden Schlüsselwörter veranlassen MATLAB die Bearbeitung des Schleifeninneren an der Stelle ihres Vorkommens abzubrechen.

- **break**
  MATLAB setzt die Programmabarbeitung unmittelbar hinter der Schleife fort.
- **continue**
  MATLAB springt an den Schleifenanfang und setzt die Programmabarbeitung mit der Auswertung der Schleifenbedingung fortgesetzt.

Bsp:

```
1  for i=1:10
2      if i==4
3          continue
4      end
5      if i==8
6          break
7      end
8      i
9  end
```

Ausgabe:

```
i=1
i=2
i=3
i=5
i=6
i=7
```

Bemerkungen I

- Wo Whitespaces, also Leerzeichen, Tabulatoren erlaubt oder erforderlich sind, können beliebig viele davon zusammenhängend verwendet werden. (Also ein Leerzeichen oder z.B. 10 für schöne Einrückung)
- In Terminal-Fenstern oder Command Windows muss nach der Texteingabe immer die Entertaste gedrückt werden, damit der Text bearbeitet (ausgeführt) wird.
- Vor dem Ausführen des eigenen Programms in MATLAB das Speichern im Texteditor nicht vergessen.
- Vorsicht bei der Benutzung von i und j: beide können in Matlab für die komplexe Zahl i stehen.

Bemerkungen II

- Der Sinn von Variablen besteht darin, Daten (eine Zahl, einen Vektor, ...) für eine spätere Verarbeitung aufzubewahren (z.B. für nächsten Schleifendurchgang.)
- Nicht vergessen, einer Variable einen Wert zuzuweisen, bevor sie verwendet wird.
- Um innerhalb eines Programms den Wert einer (existierenden) Variable zu kontrollieren, fügt man an der interessierenden Stelle eine Zeile mit nur dem Namen der Variable ein
- Das Malzeichen zwischen zwei Faktoren darf nicht weggelassen werden.

Bemerkungen III

- Operatoren haben immer einen oder mehrere Operanden. Z.B. Der + - Operator erwartet links und rechts von ihm einen Operanden: `a + b`. Sind mehrere Additionen verkettet, etwa `a + b + c + d` klammert Matlab implizit und berechnet folgenden Ausdruck: `((a + b) + c) + d`.
  Die i-ten Partialsummen S_i in den Übungen berechnen sich mit $q_i = q(i)$ als

$$S_i = S_{i-1} + q_i = ((((q_1 + q_2) + q_3) + \cdots + q_{i-1}) + q_i).$$