



Einführung in Matlab

4. Schleifen

Prof. Dr. Christiane Zarfl, Dipl.-Inf. Willi Kappler, Prof. Dr. Olaf
Cirpka



- wie Sie durch Skripte Befehlsfolgen wiederverwendbar machen.

Wie kann ich häufig vorkommende Berechnungen/Abläufe automatisieren?



- können Sie Berechnungen “umgangssprachlich” als Algorithmus formulieren.
- können Sie in Matlab Algorithmen implementieren und verwenden dabei sicher die Hilfsmittel der
 - logischen Operatoren
 - IF-THEN-ELSE-Anweisungen
 - While- und For-Schleifen

- **Ziel:** Lösen einer (beliebig) komplizierten Berechnungsaufgabe
- **Vorgehen:**
 - 1 Genaue Beschreibung der Aufgabe
 - 2 Formulierung eines **Algorithmus** (Schritt-für-Schritt Berechnungsanweisung) in Form eines Ablaufplans/“Kochrezepts” oder Pseudocode
 - 3 Umsetzung in Programmiersprache
 - 4 Testen des Programms, evtl. zurück zu Punkt 2, oder 3
 - 5 Wartung und Pflege des Programms während der Nutzung
- Setzt voraus, dass Möglichkeiten der Programmiersprache bekannt sind
- In diesem Kurs geht es hauptsächlich um Punkt 3



- Sprachliche Mischung aus natürlicher Sprache, mathematischer Notation und einer höheren Programmiersprache
- Dient genauer Beschreibung des Algorithmus
- Ist für Menschen leicht verständlich
 - kann aber vom Computer (noch) nicht ausgeführt werden
 - ist in den meisten Fällen keine Programmiersprache
 - orientiert sich aber oft an “echten” Programmiersprachen
- Soll Algorithmen verständlich und klar ausdrücken, ohne auf die Eigenheiten einer Programmiersprache Rücksicht nehmen zu müssen
- Umgangssprache hilft, Verfahrensschritte zu verdeutlichen
 - `“durchlaufe das Feld a mit Index i”`
 - `“vertausche die Inhalte der Variablen x und y”`

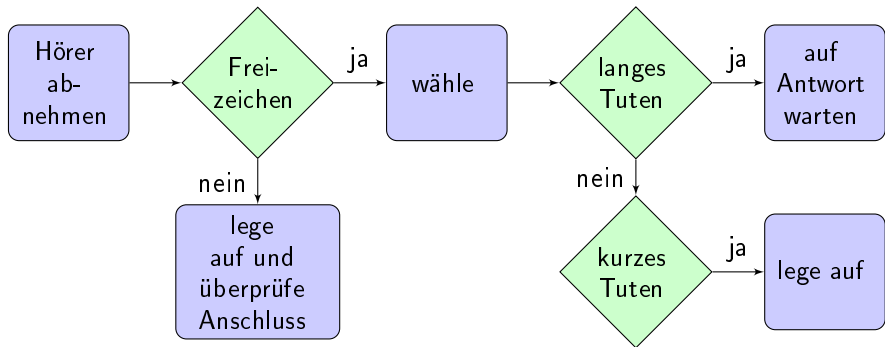


- Telefoniervorgang als Pseudocode:

```
1 Hoerer abnehmen
2 IF Freizeichen
3     THEN waehle
4         IF langes Tuten
5             THEN auf Antwort warten
6         ELSE IF kurzes Tuten
7             THEN lege auf
8 ELSE lege auf und ueberpruefe Anschluss
```



- Telefoniertvorgang als Flussdiagramm:





Schreiben Sie einen Pseudocode für die Berechnung des Mittelwerts m eines Zahlenvektors x

$$m = \frac{1}{n} \sum_{i=1}^n x_i$$



- Vergleich zwischen zwei **Zahlen**:

- $a = (3 > 1) \Rightarrow$ erzeugt logische Variable a mit Wert `true (1)`
- $2 \geq 4 \Rightarrow$ `false (0)`
- $1 < \pi \Rightarrow$ `true (1)`
- $3 == \pi \Rightarrow$ `false (0)`
- vergleichendes "ist gleich" durch "=", Zuweisung durch "="

- Vergleich zwischen zwei **Vektoren**:

- $[1:4] < [5:-2:-1] \Rightarrow$ `[true,true,false,false]`

- Logisches "und" durch "&":

- $(x > 1) \& (x < 5) \Rightarrow$ wahr für alle Werte von x zwischen 1 und 5

- Logisches "oder" durch "|":

- $(x > 1) | (x < 5) \Rightarrow$ ist immer wahr

- Logische Negation durch "~":

- $a \sim 5 \Rightarrow$ wahr, wenn $a \neq 5$



- Die Benutzung logischer Operatoren als Auswahlindizes (Filter) ist eine angenehme Besonderheit von Matlab.
- Erzeuge einen x-Vektor zwischen 0 und 10. Wähle alle x-Werte für $x^2 \geq 4$ und $x^2 \leq 64$ aus.
- In Matlab sehr einfach zu realisieren:
 - `x = [0:1:10]`
 - `x (x.^2>=4 & x.^2<=64)`
 - Gibt `[2 3 4 5 6 7 8]` zurück



- Beispiel: Ratengesetz nur dann auswerten, wenn die Konzentration größer null ist:

```
1  if c > 0                % Bedingung
2      rate = k*c/(c+K); % falls erfuehlt
3  else
4      rate = 0;           % falls nicht erfuehlt
5  end
```

- Allgemein formuliert erlaubt Matlab (siehe [doc if](#)):

```
1  if expression1
2      statements1
3  elseif expression2 % weitere Bedingungen
4      statements2
5  else
6      statements3
7  end
```



- Studieren als Pseudocode:

```
1 Waehle Studiengang
2 Einschreiben
3 ECTS = 0

5 WHILE (ECTS < 168)
6     waehle neue Veranstaltung
7     beende Veranstaltung erfolgreich
8     addiere ECTS
9 END

11 Schreibe Bachelorarbeit
```



- Wird wiederholt, solange Bedingung erfüllt ist:

```
1  x=1;           % Initialisierung
2                  % Beginn der Schleife
3  while x<10     % Fortsetzungsbedingung
4                  % diverse Operationen
5      x=x*1.2;    % oder irgendetwas anderes,
6                  % was Einfluss auf die
7                  % Bedingung hat
8  end            % Ende der Schleife
```



- Vorlesungsbesuch als Pseudocode:

```
1  Registriere unter Ilias
3  FOR woche=1:1:10
4      besuche Vorlesung
5      lade Hausaufgaben unter Ilias herunter
6      bearbeite Hausaufgaben
7      lade eigene Ergebnisse unter Ilias hoch
8      erhalte Rueckmeldung
9      IF Rueckmeldung unklar
10         stelle Rueckfragen
11     ELSE merke richtiges Vorgehen
12 END
14 Schreibe Klausur
```



- Schleife mit vorgegebener Anzahl an Wiederholungen:

```
1 for i=1:10 % 10 Durchgaenge
2     % Befehle, die eventuell von i abhaengen
3 end
```

- Bei jedem Schleifendurchgang wird dem Index i der nächste Wert des Vektors $[1:10]$ zugewiesen
- Muss nicht ein einfacher Zähler sein:

```
1 tvec=[2 5 8 12 33];
2 for t=tvec
3     % auszufuehrende Befehle
4 end
```



While Schleife

- Gut, wenn vorher nicht genau bekannt ist, wann der Abbruch erfolgen muss.
- Erlaubt kompliziertes Fortführungskriterium.
- **Gefahren:**
 - verpasster Einstieg (Typ `while 0==1`)
 - Endlosschleife, weil Kriterium immer erfüllt (Typ `while 1==1`)

For Schleife

- Gut, wenn Anzahl der Durchgänge vorher festliegt.
- Ein/Ausstieg gewährleistet.
- Zusätzliche Verlängerung nicht möglich.
- Vorzeitiger Ausstieg mit `break`-Befehl möglich:

```
1 for i=1:10
2     % Berechnung
3     if (Bedingung)
4         break
5     end
6 end
```




- Schreiben Sie ein Matlab-Programm zur Berechnung des Mittelwerts eines Zufallsvektors unter Verwendung von Schleifen
- Schreiben Sie denselben Code vektorisiert (`sum(x)` berechnet die Summe des Vektors `x`).
- Vergleichen Sie Ihr Ergebnis mit dem Matlab-Befehl `mean`!
- **Zusatz:** Die Befehle `tic; ... toc`; messen die Zeit. Vergleichen Sie die Laufzeit ihres Programms und die des Matlab-Befehls `mean` (für große Vektoren)!



- **Ziele der Hausaufgaben zur Vertiefung**

- Sie können umgangssprachlich formulierte Anweisungen in ein Matlab-Programm “übersetzen”.
- Sie implementieren eine komplexe Berechnung in einem Skript und stellen diese grafisch in einem Oberflächenplot dar.

- **Nächste Einheit - noch mehr “Recycling”:**

- Implementieren von eigenen Funktionen in Matlab