Mathematik am Computer

6. Übung: Matlab, Teil III

Marcus Grote und Helmut Harbrecht

Universität Basel

1. – 5. November 2021

Übersicht

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.

Verzweigung: if-elseif-else-end

```
if n == 1
      Befehle
end
```

Falls $n = 1$ gilt, so werden *Befehle* ausgeführt, sonst keine.

```
if n == 1
      Befehle
else
      BefehleAlt
end
```

Falls $n = 1$ gilt, so werden *Befehle* ausgeführt, sonst *BefehleAlt*.

Verzweigung: if-elseif-else-end

```
if n == 1
      Befehle1
elseif n == 2
      Befehle2
elseif n == 15
      Befehle3
else
      BefehleAlt
end
```

- Falls $n = 1$ ist, so werden *Befehle1* ausgeführt.
- Falls $n = 1$ falsch ist und $n = 2$ gilt, so werden *Befehle2* ausgeführt.
- Falls sowohl $n = 1$ und $n = 2$ falsch sind und $n = 15$ gilt, werden *Befehle3* ausgeführt.
- In allen anderen Fällen werden *BefehleAlt* ausgeführt.

Logische Ausdrücke

- Aussagen wie "*A* ist grösser als 0" (`A>0`) oder "*B* ist nicht gleich 5" (`B~=5`).
- Werden in Matlab mit einem Vektor oder Matrix dargestellt vom Typ `logical`, deren Einträge `1` oder `0` bzw. `true` oder `false` sind.
- Man verbindet Zahlen und Vektoren oder Matrizen mit == (gleich), ~= (nicht gleich (äussere Negation)), <, >, <=, >=
- Logische Ausdrücke lassen sich mit sog. Junktoren zu komplexen Aussagen verbinden, diese sind: `&` für *und*, `|` für *oder* und `xor` für *entweder-oder*.
- Eine logische Matrix lässt sich mit ~ (nicht) verneinen, d.h. aus *wahr* wird *falsch* und umgekehrt.

Logische Ausdrücke

- Vergleichen wir zwei Matrizen oder Vektoren **A** und **B**, müssen sie gleich gross sein, da die jeweils korrespondierenden Einträge a_{ij} und b_{ij} miteinander verglichen werden. Das Resultat ist dann eine gleich grosse Matrix mit `1` oder `0` als Einträgen. Es können aber auch Matrizen mit einer einzigen Zahl verglichen werden. Matlab vergleicht dann jeden Eintrag der Matrix einzeln mit dieser einen Zahl.
- Beispiel:

```
>> x = 1:9; x >= 5

ans =

     0     0     0     0     1     1     1     1     1
```

Logische Ausdrücke

Quadrieren aller negativen Zahlen in einem Vektor:

```
>> x = -4:4;
>> my_indices = x < 0

my_indices =

1x9 logical array

1   1   1   1   0   0   0   0   0

>> x(my_indices) = x(my_indices).^2
x =

16     9     4     1     0     1     2     3     4
```

Speicherverwaltung

⟩ Wenn auf einen Matrixeintrag zugegriffen wird, etwa durch `A(2,3:6)`, so muss die Matrix mindestens die Größe 2×6 besitzen und entsprechend **vorher** definiert werden, etwa z.B. als `A = zeros(2,10)`.

⟩ Wenn `A(2,3:6)` ein Wert zugewiesen wird, ohne die Matrix vorher zu definieren oder falls sie zu klein ist, so wird sie automatisch als Nullmatrix der notwendigen Größe angelegt oder entsprechend vergrößert. Das kann zu Fehlern führen und die Laufzeit der Programme verlängern.

⟩⟩⟩⟩ **Lösung**: Initialisiere alle Variablen die gebraucht werden!

- Speicher nicht mehr benötigter Variablen freigeben durch:

  `clear` oder `clear`+Variablename
- `clc` löscht alle vorherigen Eingaben und Ausgaben im Command Window.

Matrix-Vektor-Schreibweise

Es gibt (mindestens) zwei Möglichkeiten, den Vektor

$$v = \begin{pmatrix} 1 & -2 & 3 & -4 & \cdots & \cdots & 999 & -1000 \end{pmatrix}$$

zu erzeugen:

```
v = 1:1000;
for i = 1:500
    v(2*i) = -v(2*i);
end
```

```
v = 1:1000;
v(2:2:1000) = -v(2:2:1000);
```

Die zweite Möglichkeit verwendet statt Schleifen die vektorielle Rechnung. Das ist viel schneller!

Man sollte daher alle Matrix- bzw. Vektor-Operationen möglichst ohne for-Schleifen umsetzen.