# Einführung in Octave

## Theorie & Aufgaben

18. Juni 2018

# 1 Einführung

### Worum geht es?

*GNU Octave* ist eine freies und quelloffenes Softwarepaket zur numerischen Lösung mathematischer Probleme wie:

- Matrizenrechnung
- Gleichungssystemen
- Differenzialgleichungen
- Integration

Octave ist weitgehend kompatibel zur kommerziellen Software *Matlab*.

Im Gegensatz zu Computeralgebrasystemen wie Maple, Mathematica oder Maxima kann Octave nur numerische und keine symbolischen Berechnungen wie $2a+3a \Rightarrow 5a$ durchführen.

Die Software kann für verschiedene Betriebssysteme kostenlos von der folgenden Webseite heruntergeladen werden.

`http://www.gnu.org/software/octave`

Ab der Version 4.0.0 wird Octave mit einer grafischen Benutzerschnittstelle ausgeliefert.

### Die Benutzeroberfläche

- Die *Menüleiste* enthält Befehle für das Arbeiten mit Octave-Skripten. Darunter befindet sich eine Auswahl entsprechender Icons.
- Der *File-Browser* (oben links) ermöglicht das Durchsuchen und Auswählen von Ordnern und Dateien.
- Das *Workspace*-Fenster (mitte links) Zeigt die Werte und den Speicherverbrauch der aktuell definierten Variablen an.
- Die *Command History* (unten links) enthält die zuletzt verwendeten Befehle. Durch einen Doppelklick können sie erneut ausgeführt werden.
- Der rechte Fensterbereich enthält drei Tabs:
  - Das *Command Window* zur interaktiven Befehlseingabe
  - Das *Editor*-Fenster zum Schreiben von Octave-Skripten. Das sind Dateien, die in der Regel mehrere Octave-Befehle enthalten und vom Octave-Interpreter abgearbeitet werden, wenn der „Run“-Befehl gegeben wird.
  - Das *Documentation*-Fenster für die Offline-Hilfe.

### Elementare arithmetische Operationen

Die Eingaben (in blauer Farbe) sind nach dem Symbol >> einzutippen.

```
5*(3+8)
```

```
(17.35+9.8)/(5.3-2.71)
```

```
2^3 + 3**2
```

### Kommentare

Kommentare werden mit dem Prozentzeichen eingeleitet.

```
3**2 % alternative Potenzdarstellung
```

### Konstanten

```
pi
```

```
e
```

```
i  % Alternative: 1i, 5.2i, -0.5i, ...
```

```
(1+2i)*(4-5i)
```

### Logische Operationen

- Die Zahl 0 steht für den Wahrheitswert *falsch/false*.
- Jede Zahl $x \neq 0$ steht für den Wahrheitswert *wahr/true*.

```
not(0)
```

```
and(1,0)
```

```
or(1,0)
```

```
xor(1, 0)
```

### Vergleichsoperatoren

```
5 > 3
```

```
5 < 3
```

```
5 <= 4
```

```
5 >= 4
```

```
5 == 3
```

```
5 ~= 3
```

### Variablen

```
a = 5
```

*Achtung:* Ein Semikolon unterdrückt die Ausgabe.

```
a = 5;
```

Auch Zeichenketten können gespeichert werden:

```
txt = "Hello"
```

Die letzte Ausgabe wird in der Standardvariablen `ans` gespeichert.

```
5 + 3;
ans + 10
```

`who` zeigt die aktuell definierten Variablen an.

```
x = 5.214;
y = "Hello";
who
```

`whos` zeigt die aktuell definierten Variablen ihre Grösse, ihren Speicherbedarf und den Datentyp an.

```
x = 5.214;
y = "Hello";
whos
```

`clear` löscht eine einzelne Variable:

```
a = 25;
clear a
who
```

`clear all` löscht alle Variablen:

```
a = 1234;
b = "Hello";
clear all
who
```

### Ausgabe

Gibt man eine Variable (ohne Semikolon) ein, so wird ihr Wert in Standardrundung ausgegeben:

```
a = pi
```

Mit `disp(...)` lässt sich die Ausgabeunterdrückung umgehen:

```
disp(42);
```

Für die meisten Zwecke genügen aber die beiden vordefinierten Formateinstellungen:

```
format long
pi

format short
e
```

Mit `sprintf(<string>, <arg1>, <arg2>, ...)` können in `<string>` benutzerdefinierter Ausgabeformate für die Argumente `<arg1>`, `<arg2>`, ....

| | |
|---|---|
| `%<b>s` | Zeichenkette |
| `%<b>d` | ganze Zahl |
| `%<b>x` | ganze Zahl in hexadezimaler Form |
| `%<b>.<n>f` | Gleitkommazahl `<n>` Nachkommastellen |
| `%<b>.<n>e` | wissenschaftl. Darst. mit `<n>` sig. Stellen |
| `%<b>.<n>g` | wählt `f` oder `g` abhängig von der Grösse der Zahl |

Mit `<b>` kann jeweils die minimale Breite der Ausgabe festgelegt werden. Wird kein Wert oder der Wert `0` angegeben, so wird so viel Platz verwendet, wie für den die Ausgabe nötig ist.

**Hinweis:** Diese Darstellung ist unvollständig. Mehr Informationen zu `sprintf()` findet man in der Dokumentation zu Octave.

- Die Zahl 12 in hexadezimaler Form:

  ```
  sprintf("%x", 12)
  ```

- Zwei Zeichenketten, die jeweils 10 Zeichen ausfüllen:

  ```
  sprintf("%10s%10s", "Hello", "World")
  ```

- Die Kreiszahl auf 9 Nachkommastellen mit zusätzlichem Text:

  ```
  sprintf("pi=%0.9f", pi)
  ```

- Die Faraday-Konstante auf 3 signifikante Stellen:

  ```
  sprintf("%0.3e", 96485.33289)
  ```

### Zufallszahlen

`rand` erzeugt eine uniform (gleichförmig) verteilte Zufallszahl im Intervall $[0, 1)$.

```
rand
```

### Mathematische Funktionen

| | |
|---|---|
| `exp(x)` | $\mathrm{e}^x$ |
| `log(x)` | $\ln(x)$ |
| `log10(x)` | $\log_{10}(x)$ |
| `abs(x)` | Absolutwert von $x$ |
| `sign(x)` | Vorzeichen von $x$ ($-1$, 0, $+1$) |
| `round(x)` | $x$ auf die nächste ganze Zahl runden. |
| `floor(x)` | $x$ auf die nächste ganze Zahl abrunden ($\searrow -\infty$) |
| `ceil(x)` | $x$ auf die nächste ganze Zahl aufrunden ($\nearrow +\infty$) |
| `fix(x)` | $x$ auf die nächste ganze Zahl zur Null runden |
| `rem(n,d)` | Divisionsrest von $n/d$ |

| | |
|---|---|
| `cos(x)` | Cosinus von $x$ (in rad) |
| `sin(x)` | Sinus von $x$ (in rad) |
| `tan(x)` | Tangens von $x$ (in rad) |
| `sinh(x)` | Sinus hyperbolicus von $x$ |
| `cosh(x)` | Cosinus hyperbolicus von $x$ |
| `tanh(x)` | Tangens hyperbolicux von $x$ |
| `acos(x)` | Arkustangens von $x$ |
| `acosh(x)` | Arkuscosinus hyperbolicus |
| `asin(x)` | Sinus hyperbolicus |
| `asinh(x)` | Arkuscosinus hyperbolicus |
| `atan(x)` | Arkustangens von $x$ |
| `atan2(y,x)` | Arkustangens für zwei Argumente |
| `atanh(x)` | Arkustangens hyperbolicus |

### Hilfe

Mit `help <befehl>` wird eine Hilfe über `<befehl>` angezeigt.

In der grafischen Benutzeroberfläche öffnet Mit `doc <befehl>` ein Fenster mit einem einfachen Dokumentationsbrowser.

# 2 Matrizen und Vektoren

## 2.1 Eingabe von Matrizen

- Vektoren und Matrizen werden durch eckige Klammern `[...]` begrenzt.
- Elemente innerhalb einer Zeile werden durch Leerzeichen oder Kommas getrennt.
- Zeilen werden durch Semikolons getrennt

Eine $2 \times 3$-Matrix:

```
A = [1 2 3; 4 5 6]
```

oder so:

```
A = [1, 2, 3; 4, 5, 6]
```

Die zusätzliche Leerzeile wird hier aus Platzgründen weggelassen.

Ein Zeilenvektor ($1 \times 3$-Matrix):

```
P = [3 4 5]
```

Ein Spaltenvektor ($2 \times 1$-Matrix):

```
v = [7; 8]
```

Der `rand`-Befehl kann auch zum Erzeugen von Matrizen mit Zufallselementen verwendet werden.

Gibt man zwei natürliche Zahlen als Argumente an, wird eine Matrix dieser Grösse mit gleichförmig verteilten Zufallszahlen in $[0, 1)$ erzeugt.

```
rand(1, 3)
```

### Magie?

Die Funktion `magic(n)` mit $n \geq 3$ gibt besondere Matrizen aus.

```
magic(3) % es sind auch magic(4), magic(5), ... möglich
```

Was ist das besondere an diesen Matrizen?

### Matrizen verbinden

Sofern die Dimensionen es erlauben, können Matrizen mit einem Komma (oder einem Leerzeichen) horizontal verbreitert werden:

```
A = [1 2; 3 4; 5 6]
B = [7; 8; 9]
[A B]  % oder: [A, B]
```

Sofern die Dimensionen es erlauben, können Matrizen mit einem Semikolon vertikal gestapelt werden:

```
C = [1 2 3; 4 5 6]
D = [7 8 9]
[C; D]
```

### Arithmetische Folgen

Arithmetische Folge mit $d = 1$:

```
4 : 7 % Achtung: das ist nicht 4/7
```

Arithmetische Folge mit $d = 2$:

```
1 : 2 : 9
```

mit $d = 0.3$:

```
0 : 0.3 : 1
```

### Spezielle Matrizen

```
ones(2, 4)
```

```
zeros(1, 3)
```

```
eye(3, 3)
```

### Die Dimension einer Matrix ändern

```
A = rand(2, 3)
reshape(A, 3, 2)
```

## 2.2 Zugriff auf die Elemente

```
A = [1 2 3 4; 5 6 7 8; 9 10 11 12]
A(2,3)
```

Mit einem Doppelpunkt an der Zeilen- oder Spaltenposition wählt man jeweils alle Elemente aus.

```
A = rand(3, 4)
A(3,:)
A(:,2)
```

Es lassen sich auch mehrere unzusammenhängede Spalten- oder Zeilenindizes auswählen:

```
A = [10 11 12 13; 14 15 16 17; 18 19 20 21]
A([2 3], [1 4])
```

### Dimension einer Matrix

Die Funktion `size(A)` gibt die Anzahl Zeilen und die Anzahl Spalten der Matrix `A` (in dieser Reihenfolge) aus:

```
A = [1 2 3; 4, 5, 6]
size(A)
```

## 2.3 Matrixoperationen

### Addition und Subtraktion

Matrizen gleicher Grösse werden wie üblich mit `+` addiert bzw. mit `-` subtrahiert.

```
A = [1, 2; 4, 5]
B = [6, 7; 8, 9]
A + B
```

### Multiplikation

Multiplikation von Zahlen und Matrizen:

```
A = [1, 2; 3, 4]
10*A
```

Die übliche Matrizenmultiplikation:

```
A = [1 2; 3 4]
B = [5 6; 7 8]
A * B
B * A
```

Die elementweise Matrizenmultiplikation (wird selten benötigt):

```
A = [1 2; 3 4]
B = [5 6; 7 8]
A .* B
```

### Die Inverse

Die Inverse einer Matrix `A` wird mit `inv(A)` berechnet.

```
A = [1 2; 3 4]
B = inv(A)
A * B
B * A
```

### Die Transponierte

Mit `A'` wird die Transponierte einer Matrix `A` ausgegeben.

```
A = reshape(1:6, 2, 3)
A'
```

Transponieren einer Matrix bedeutet, ihre Zeilen und Spalten zu vertauschen.

### Die Spur

Die Spur einer quadratischen $n \times n$-Matrix $A$ ist die Summe ihrer Diagonalelemente $a_{i,i}$ mit $i = 1, 2, \ldots, n$.

```
A = [1 2; 3 7]
trace(A)
```

### Gauss-Jordan

Mit `rref` (reduced row echelon form) bringt man eine Matrix auf reduzierte Zeilenstufenform.

```
A = reshape(1:20, 4, 5)
rref(A)
```

### Die Determinante

Die Determinante ist eine Funktion, die jeder quadratischen Matrix $A$ eine Zahl zuordnet.

Diese Zahl enthält Informationen über bestimmte Eigenschaften der Matrix. Beispielsweise hat eine quadratische Matrix keine Inverse, wenn ihre Determinante null ist.

```
A = [1 2; 3 4]
det(A)
```