Kurze Einführung in Octave

Numerische Mathematik I

Wintersemester 2009/2010, Universität Tübingen

Starten von Octave

in einer Konsole

```
octave
```

eintippen (unter Linux)

Octave als Taschenrechner

Beispiele:

```
>> 7*3
ans =  21
>> 135+4*(3-5)
ans =  127
>> 2^10
ans =  1024
>> sin(pi/3)
ans =  0.86603
>> i*i
ans =  -1
```

Variablen

- `variablenname = wert;` weist der Variablen "variablenname" den Wert "wert" zu
- `variablenname = wert` weist der Variablen "variablenname" den Wert "wert" zu und gibt ihn aus

Beispiel:

```
>> a = 3;
>> b = 4
b =  4
>> b = b+10;
>> b
b =  14
```

Mehrere Befehle können in einer Zeile zusammengefasst stehen:

- ; trennt die Befehle (ohne Ausgabe)
- , trennt die Befehle und gibt Werte aus

Beispiel:

```
>> a = 3; a = a*4, a = a+5;
a =  12
>> a
a =  17
```

Vektoren

Zeilenvektor:

```
>> v = [1 2 3];
```
für $v = (1, 2, 3)$

Spaltenvektor:

```
>> v = [1; 2; 3];
```
für $v = (1, 2, 3)^T$

Einfache Erzeugung von Vektoren:

- `start [: inkrement] : stop`

Beispiel:

```
>> x = 3:6
x =
    3 4 5 6
>> y = 0:0.15:0.7
y =
    0 0.1500 0.3000 0.4500 0.6000
```

Matrizen

Matrix:

`>> A = [1 2; 3 4];` für $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

Spezielle $m \times n$-Matrizen:

- `eye(m,n)` erzeugt eine Matrix mit Einsen auf der Hauptdiagonalen und Nullen sonst
- `zeros(m,n)` erzeugt die Nullmatrix
- `ones(m,n)` erzeugt Matrix aus lauter Einsen
- `rand(m,n)` erzeugt Zufallsmatrix mit gleichverteilten Einträgen aus $[0, 1]$

Die Dimension:

- `length(v)` gibt die Länge des Vektors v zurück
- `size(A)` gibt die Anzahl von Zeilen und Spalten der Matrix A zurück (als Vektor)

Operationen:

- `+, -, *` Addition, Subtraktion, Multiplikation von Matrizen / Vektoren
- `A'` transponiert die Matrix A und konjugiert sie komplex
- `A.'` transponiert die Matrix A

Komponentenweise Operationen:

- `.+, .-, .*, ./, .^` komponentenweise Operationen

Beispiel:

```
>> A = [1 2; 3 4]; B = 2*ones(2,2);
>> x = [1; 5]; A*B*x
ans =
      36
      84
>> A.*B
ans =
      2 4
      6 8
```

Manipulationen:

- `v(j)` j-te Komponente v_j des Vektors v
- `A(j,k)` Eintrag $A_{j,k}$ der Matrix A
- `A(j,:)` j-te Zeile der Matrix A
- `A(:,k)` k-te Spalte der Matrix A

Beispiel:

```
>> A = [1 2 3; 4 5 6]; v = [7;8];
>> v(2)
ans =
      8
>> A(2,2:3) = [0 0]
ans =
      1 2 3
      4 0 0
```

Mit Octave kann man leicht

- lineare Gleichungssysteme lösen (`A\b`)
- LR-Zerlegungen berechnen (`lu(A)`)
- QR-Zerlegungen berechnen (`qr(A)`)
- Choleski-Zerlegungen berechnen (`chol(A)`)
- Hessenberg-Transformationen berechnen (`hess(A)`)
- Eigenwerte berechenen (`eig(A)`)
- Normen berechnen (`norm(v)`)
- Konditionszahlen berechnen (`cond(A)`)
- mit dünnbesetzten Matrizen rechnen
- ...

Funktionen und Skripte

Neben den eingebauten Funktionen in Octave (z.B. `sin`, `lu`) werden wir eigene **Funktionen** schreiben. Dies geschieht in einer Textdatei mit Endung .m (*m-file*).

Mathematik:

$$f : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \times \mathbb{R},$$
$$(x_1, x_2) \mapsto (y_1, y_2) = (x_1 + x_2, x_1 \cdot x_2)$$

Octave:

```
% Diese Funktion berechnet ...
function [y1,y2] = name(x1,x2)
y1 = x1 + x2;
y2 = x1 * x2;
```

- `[y1,y2]` bezeichnet die Rückgabe der Funktion ($\widehat{=}\ (y_1, y_2)$)
- `x1` und `x2` sind die Eingabeparamter ($\widehat{=}\ (x_1, x_2)$)
- `name` ist der Name der Funktion ($\widehat{=}\ f$); das m-file muss dann name.m heißen
- $\mathbb{R} \times \mathbb{R}$ hat in Octave keine Entsprechung

Grundgerüst:

- % leitet einen *Kommentar* ein, der das Verhalten der Funktion beschreibt
- `function ...` bestimmt *Eingabe*, *Rück-/ Ausgabe* und *Name* der Funktion
- dann kommen Befehle, die beschreiben, wie man die Rückgabe berechnet (evtl. sehr viele Befehle, evtl. neue Variablen, ...)

Beispielaufruf:

```
>> [out1,out2] = name(3,4)
out1 =  7
out2 =  12
>> x1
error: 'x1' undefined ...
```

Die innerhalb einer Funktion belegten und verwendeten Variablen sind außerhalb der Funktion nicht sichtbar.

Ein **Skript** ist ebenfalls in einem m-file gespeichert. Die einzelnen Zeilen werden so behandelt, als würden sie direkt in Octave eingegeben. Insbesondere sind die dort definierten Variablen auch außerhalb des Skripts sichtbar. Es gibt keine Eingabe- und Ausgabeparameter.

Grundgerüst:

```
% Diese Skript berechnet ...
var1 = 5+8;
var2 = 7*var3;
```

Beispielaufruf von meinskript.m:

```
>> var3 = 9; meinskript
>> var1
var1 =  13
>> var2
var2 =  63
```

Schleifen und Verzweigungen

- `for name = ausdruck`

  `...`

  `end`

  Beispiel:

  ```
  for n = 1:10
     x(n) = sin(n);
  end
  ```

- `while condition`

  `...`

  `end`

  Beispiel:

  ```
  while t < T
     t = t+h;
  end
  ```

- `if condition`

  `...`

  `[else`

  `...]`

  `end`

  Beispiel:

  ```
  if x < 0
     x = -x;
  end
  ```

Logische Operatoren

`<`	kleiner	`<=`	kleiner oder gleich
`>`	größer	`>=`	größer oder gleich
`==`	gleich	`~=`	ungleich
`&`	und	`|`	oder
`~`	nicht		

Ergebnis: `1` falls wahr, `0` sonst.

Es gibt in Octave Wahrheitswerte `true` und `false` mit `true == 1` und `false == 0`.

Graphische Ausgabe (2D)

Beispiel:

```
>> x = 0:2*pi/1000:2*pi;
>> y = zeros(1,1001);
>> for n = 1:1001, y(n) = sin(x(n)); end
>> plot(x,y)
```

Kürzer (Vektorisieren!):

```
>> y = sin(x)
```

Weitere Befehle: `semilogx`, `semilogy`, `loglog`

Graphische Ausgabe (3D)

Beispiel:

```
>> x = -1:0.01:1;
>> [xx,yy] = meshgrid(x,x);
>> mesh(x,x,xx.^2 + yy.^2)
```