Crashkurs in Octave

Allgemeines

- freier Clone von Matlab, d.h. Tool für wissenschaftliches Rechnen und Visualisierung (nicht zu verwechseln mit Algebrasystemen wie Mathematica und Maple!)
- programmierbar, aber schon mit zahlreichen vordefinierten Funktionen bestückt
- "matrix-basiert" – die Welt besteht nur aus Matrizen...
- Programmstart mit `'octave'`
  $\Rightarrow$ liefert einen Eingabeprompt
- wichtigstes Kommando: `'help'`
- benutzbar wie ein Taschenrechner

Arbeitsumgebung und Bedienung

- Variablenzuweisung: `x = ...`
- falls kein Name explizit angegeben wird, landet das Ergebnis in `'ans'`
- ein Semikolon am Ende unterdrückt die Ausgabe von Ergebnissen (Abbruch mit Ctrl-c)
- wichtige Kommandos:
  - `'who'` listet alle aktuell definierten Variablen auf
  - `'clear'` löscht alle Variablen
  - `'clear x'` löscht Variable x
  - Variablendefinitionen gelten nur, solange der Octave-Prozess läuft
    - `'save'` speichert die Variablen der Arbeitsumgebung
    - `'load'` stellt den gespeicherten Zustand wieder her
- Ausgabe von Textmeldungen: `disp(''Hallo!'');`
- formatierte Ausgaben: `printf(''%f'',x)` wie in C
- Eingaben: `c = input(''Gib was ein!'');`

Datentypen

- einziger Datentyp in Octave sind Matrizen, d.h. ein Skalar ist eine 1×1-Matrix, ein String ist eine Matrix mit char-Einträgen

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

- Matrizen können direkt definiert werden, mit speziellen Funktionen erzeugt oder über for-Schleifen (Achtung!) angelegt werden
  - `'rand(m)'` $\Rightarrow$ erzeugt quadratische Zufallsmatrix
  - `'ones(m,n)'` $\Rightarrow$ erzeugt $m \times n$-Matrix mit Einsen
  - `'zeros(m,n)'` $\Rightarrow$ erzeugt $m \times n$-Matrix mit Nullen
  - per for-Schleife:

```
for i=1:5
   A(1,i)= i;
end
```

- Elementzugriffe:
  - Element (i,j): `A(i,j)`
  - 3.Spalte: `A(:,3)`
  - 5.Zeile: `A(5,:)`
  - Submatrizen: `A([1,3],[2,3])`
- Matrixoperationen: $+, -, *, \hat{}, \ldots$

  WICHTIG: Operatoren arbeiten entweder element- (z.B. `'sin'`) oder matrizen- bzw. vektorweise (z.B. `'inv'`), die Konvertierung der Basisoperatoren erfolgt über einen vorangestellten Punkt:

  `A * B` versus `A .* B`
- Matrixtransponierung: '
- Gleichungssysteme:
  - `A \ b` löst $A \cdot x = B$
  - `b / A` löst $x \cdot A = B$

Programmierung

- Octave bietet alle klassischen Konstrukte einer Hochsprache wie `for, if, while,` etc.

  ABER: Octave ist optimiert auf die Berechnung von Matrizen, d.h. Operationen sollten wenn möglich immer durch Matrizenoperationen realisiert werden!

  Beispiel: Lege eine 3×10-Matrix an, die in jeder Zeile die Zahlen von 1 bis 10 enthält.

  a) intuitiv:
  ```
  for i=1:10
    A(1,i) = i;
    A(2,i) = i;
    A(3,i) = i;
  end
  ```
  b) schnell: `ones(1,3)' * [1:10]`

Grafik

- 2 wichtige Kommandos:
  - `'plot(x,y)'` nimmt zwei Vektoren gleicher Länge und plottet punktweise die resultierenden Paare

    ```
    x = [-1.5:0.01:1.5] ; y= exp(-x.^2)
    ```
  - `'hist(x,15)'` berechnet ein Histogramm mit 15 Bins über die Daten des Vektors x (hilfreich, um sich Verteilungen zu veranschaulichen...)
  - Ausgabe in Datei (zumindest unter Linux/Unix):

    ```
    gset term postscript;
    gset output ''plot.ps'';
    replot;
    gset term x11;
    ```

Statistik

- diverse vordefinierte Verteilungen und Tests...
  - Binomialverteilung `'binomial_...'`
  - Normalverteilung `'normal_...'`
  - Poissonverteilung `'poisson_...'`
  - Exponentialverteilung `'exponential_...'`
  - ...
- am einfachsten zu finden durch Blick in Verzeichnisse unterhalb von

  /usr/share/octave/2.1.x/m/statistics/...

Skript- und Funktionsdateien

- Endung in der Regel ".m"
- enthalten Octave-Kommandos, die durch Aufruf des Dateinamens oder über `load` ausgeführt werden können
- Funktionsdateien enthalten Funktionsdefinitionen:

```
% SUMUP Sums up all entries of the vector
%    s = sumUp(m,n)
%    m is a vector of dimension n,
%    s is the result
%    (function uses a for-loop thus is quite slow!)
function s = sumUp(m,n)
  s = 0;
  for i=1:n
    s = s + m(i);
  end
endfunction
```