Informationen zum Rechnerpraktikum "Einführung in die Optimierung"

TECHNISCHE UNIVERSITÄT DARMSTADT

Dr. Nicole Megow

Dipl. Math. Konstantin Pertschik

Wintersemester 11/12

In den Rechnerübungen sollen einige der in der Vorlesung vorgestellten Algorithmen implementiert werden. Dazu wird octave verwendet, dessen Syntax im Wesentlichen der von matlab entspricht. [1]

Die Rechnerübungen finden in den Rechnerräumen K309 und K313 statt.

1 Hilfen zu octave

Eine ausführliche Dokumentation zu octave gibt es im Netz unter

`www.octave.org` bzw. `www.gnu.org/software/octave/doc/interpreter/`.

Dort wird allerdings die aktuellste Version beschrieben, sodass nicht alle dort erwähnten Funktionen auch in der Version auf den Poolrechnern vorhanden sind.

Im Folgenden wird eine kleine Auswahl von Befehlen aufgeführt, die bei der Implementierung hilfreich sein könnten.

2 Definieren eigener Funktionen

Eigene Funktionen definiert man sich am Besten in sogenannten m-Files. Deren Name sollte mit dem Namen der Funktion übereinstimmen und sie sollten die Endung `m` haben. Das heißt, die Funktion `simplex` sollte in einer Datei mit dem Namen `simplex.m` definiert werden. Die erste Zeile der Datei `simplex.m` könnte beispielsweise folgendermaßen aussehen:

```
function [xOpt, message] = simplex(A, b, c, B);
```

Damit die neudefinierte Funktion von octave auch gefunden wird, muss das zugehörige m-File entweder im aktuellen Verzeichnis liegen oder das Verzeichnis, in dem das m-File liegt, muss in der Liste der Verzeichnisse, die octave durchsucht, enthalten sein. Diese Liste kann man sich mit dem Befehl `path` ausgeben lassen. Mit demselben Befehl kann diese Liste auch verändert werden. Zum Beispiel erweitert der Befehl

```
path(LOADPATH, "~/OptimierungI/octave")
```

oder in Matlab

```
addpath '~/OptimierungI/octave'
```

die Liste um das Verzeichnis `~/OptimierungI/octave`. Diese Änderung geht allerdings verloren, sobald octave beendet wird. Um nicht bei jedem Start von octave die Liste wieder verändern zu müssen, kann der entsprechende Befehl in die Datei `.octaverc` geschrieben werden. Alles, was in dieser Datei steht, wird beim Starten von octave ausgeführt. In Matlab kann der Pfad gespeichert werden, indem er im aktuellen Verzeichnis in der Datei 'pathdef.m' hinzugefügt wird.

[1] Es stehen am Fachbereich zwar auch einige matlab-Lizenzen zur Verfügung, aber nicht in ausreichender Zahl.

3 Ausgabe

matlab und octave: Mit dem Befehl `disp` können die Werte von Variablen ausgegeben werden. Stattdessen kann auch der Befehl `fprintf` benutzt werden, der sowohl unter matlab als auch unter octave funktioniert. Ein Syntaxbeispiel ist:

```
octave:29> i = 42
i = 42
octave:30> x = 1/3
x = 0.33333
octave:31> disp(x)
0.33333
octave:32>  fprintf('Der Wert von i: %i\nDer Wert von x: %f\n', i, x)
Der Wert von i: 42
Der Wert von x: 0.333333
```

octave: Mit dem Befehl `printf` ist in eine formatierte Ausgabe von einer oder mehreren Variablen möglich. Die Syntax ist analog zum entsprechenden C-Befehl. Beispiel:

```
octave:32> printf("Der Wert von i: %i\nDer Wert von x: %f\n", i, x)
Der Wert von i: 42
Der Wert von x: 0.333333
```

Soll die Ausgabe nach einem `printf` sofort erscheinen, muss nach dem `printf` der Befehl `fflush(stdout)` aufgerufen werden.

4 Eingabe

Für die Eingabe von Werten durch den Benutzer kann der Befehl `input` verwendet werden. Die Eingabe wird als Ausdruck behandelt und dementsprechend ausgewertet. Beispiel:

```
octave:9> i = input("Wert von i=")
Wert von i=23+24
i = 47
```

bzw. mit einfachen Anführungszeichen in Matlab:

```
matlab: i = input('Wert von i='')
Wert von i=23+24
i = 47
```

5 Lösen linearer Glechungssysteme

Ist `A` eine invertierbare Matrix und b ein Vektor, dann liefert der Ausdruck `A\b` die Lösung des Gleichungssystem $Ax = b$.

6 Teilmatrizen

Sollen aus einer Matrix `A` bestimmte Spalten ausgewählt werden, kann dies folgendermaßen realisiert werden: Sei `B` der Vektor, in dem die auszuwählenden Indizes stehen. Dann ergibt der Ausdruck `A(:, B)` die gewünschte Matrix. Beispiel:

```
octave:13> A = [1 2 1 0; 3 4 0 1]
A =
```

```
  1  2  1  0
  3  4  0  1

octave:14> B = [2 4]
B =

  2  4

octave:15> A(:, B)
ans =

  2  0
  4  1
```

7 Suchen von Einträgen

Sind in einem Vektor alle Einträge gesucht, die einer bestimmten Bedingung genügen, ist dies einfach mit der Funktion `find` zu realisieren. Zum Beispiel ergibt `find(v)` die Indizes aller Einträge von `v`, die nicht Null sind, und `find(v==0)` die aller Einträge, die gleich Null sind, usw.:

```
octave:16> v = [-1 0 1 0 2 3]
v =

  -1   0   1   0   2   3

octave:17> find(v)
ans =   1  3  5  6

octave:18> find(v==0)
ans =

  2  4

octave:19> find(v>0)
ans =   3  5  6

octave:20> find(v<0)
ans = 1
```

Der Ausdruck `v(find(v>0))` erzeugt einen Vektor, der nur die Einträge von `v` enthält, die größer als Null sind.

```
octave:21> v(find(v>0))
ans =   1  2  3
```