Fachbereich Mathematik
JProf. Dr. Christian Meyer
Lucia Panizzi, Cedric Sehrt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

WS 2008/2009
28. und 30.10.2009

1. Aufgabenblatt des Rechnerpraktikums zur „Nichtlinearen Optimierung“

Aufgabe P1 (Gradientenverfahren)

Implementieren Sie das Gradientenverfahren aus der Vorlesung in MATLAB. Verwenden Sie zur Bestimmung der Schrittweite die Schrittweitenregel von Armijo. Implementieren Sie hierbei die Schrittweitensuche unabhängig vom Hauptalgorithmus des Gradientenverfahrens, so dass Sie die Schrittweitensuche später in anderen Verfahren wiederverwenden können. Beachten Sie, dass das Programm insbesondere auch für Zielfunktionen $f : \mathbf{R}^n \to \mathbf{R}$ mit $n > 1$ funktionieren soll.

Verwenden Sie die folgende Anleitung:

Zum Gradientenverfahren:

Verwenden Sie den Funktionskopf:

```
function[xn] = grad(x0,fg,tol,maxit)
```

mit folgenden Parametern:

x0: Startpunkt,

fg: Eine MATLAB-Funktion, die beim Aufruf [f,g] = fg(x) den Funktionswert $f = f(x)$ der Zielfunktion und deren Gradient $g = \nabla f(x)$ zurückgibt, beim Aufruf [f] = fg(x) nur den Funktionswert $f(x)$.
Hinweis: Verwenden Sie hierfür die MATLAB-Funktion `nargout`.

tol: Toleranz für die Abbruchbedingung: $\|g(x_k)\| \leq tol \cdot \min(1, \|g(x_0)\|)$,

maxit: Maximale Anzahl durchzuführender Iterationen.

Zur Armijo-Regel:

Verwenden Sie den Funktionskopf:

```
function [sig, xn, fn] = armijo(xk,sk,stg,fg,fk,gamma),
```

mit den Eingabedaten:

xk: aktueller Punkt , **sk**: aktuelle Suchrichtung, **stg**: $s_k^T g(x_k)$, **fg**: Zielfunktion (siehe oben), **fk**: aktueller Funktionswert, **gamma**: Parameter γ der Armijo-Regel (siehe VL).

und den Ausgabedaten **sig**: Schrittweite, **xn**: ermittelter Punkt, **fn**: Zielfunktionswert am ermittelten Punkt.

Wählen Sie den Parameter $\beta = 0.5$ der Armijo-Regel fest.

Testen Sie ihr Verfahren für folgende Funktionen und lassen Sie sich die benötigten Iterationen ausgeben:

- $f_1(x_1, x_2) = x_1^2 + \alpha x_2^2$ mit verschiedenen $\alpha \geq 1$,
- Rosenbrock: $f_2(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ (mit globalem Minimum bei $(1, 1)$),
- Dixon: $f_3(x_1, \ldots, x_{10}) = (1 - x_1)^2 + (1 - x_{10})^2 + \sum_{i=1}^{9}(x_i^2 - x_{i+1})^2$ (globales Minimum ist der Vektor $(1, 1 \ldots 1)$).

Aufgabe P2 (Veranschaulichung des Gradientenverfahrens)

Ziel der Aufgabe ist es den Verlauf der Iterationspunkte zu visualisieren. Gehen Sie dazu wie folgt vor:

1. Erweitern Sie zuerst ihr Programm zur Implementierung des Gradientenverfahrens: In jeder Iteration soll nun die Verbindungsgerade zwischen altem und neuem Iterationspunkt – mit dem Befehl `plot`– geplottet werden. In der k-ten Iteration entspricht dies genau der Richtung des steilsten Abstiegs im Punkt x_k mit der Länge der gewählten Schrittweite.
2. Legen Sie nun – zunächst für die Funktion f_1, mit $\alpha = 10$ – eine Matlab-Datei *runf1.m* an, in der Sie alle benötigten Schritte zum Aufruf und der Visualisierung Ihres Programms für die Funktion f_1 zusammenfassen:
   Erzeugen Sie zunächst ein Höhenliniendiagramm zur Funktion f_1 – mit dem Befehl `contour`.
   Rufen Sie nun ihr erweitertes Programm für das Gradientenverfahren auf, welches den Pfad der Iterationspunkte dort einzeichnet.

Hinweis: Damit die Richtungen des Gradientenverfahrens in Ihr Höhenliniendiagramm eingezeichnet werden (und nicht für jede Iteration eine neue Graphik erstellt wird) benötigen Sie den Befehl `hold on`.

Testen Sie Ihr erweitertes Programm für die Funktion f_1 und erstellen Sie dann analog eine Datei runf2.m, um die Funktion f_2 zu testen.

Links zu Matlabdokumentationen im Internet:

http://www.rrz.uni-hamburg.de/RRZ/W.Wiedl/Skripte/Matlab/

http://homepages.fh-regensburg.de/ wah39067/Matlab/MTut2-0.pdf

http://www-m3.mathematik.tu-muenchen.de/m3/ftp/matlab.ps