



Übungen zur Numerischen Optimierung - Blatt 1

Besprechung am 27.10.16, 10:15 Uhr im Raum 220 (HeHo18)

Hinweise

- Die Vorleistung besteht aus der erfolgreichen Bearbeitung (70%) der Aufgaben. Die Erfassung erfolgt durch ein Votiersystem.
- Die Übungen finden wöchentlich statt und haben ein Programmieranteil, der in Zweiergruppen vorgestellt wird.
- Matlabaufgaben sind **zu zweit** im Moodle einzureichen (den Code unbedingt mit dem Namen der Studenten der Zweiergruppe kommentieren).

1. (Aufstellen eines LPs)

Eine Horde von Steinzeitmenschen ist durch eine Naturkatastrophe von der Außenwelt total abgeschnitten und muss sich für die nächsten fünf Jahre völlig von ihrer Ernte und ihren Vorräten versorgen. Glücklicherweise ist gerade Herbst und es kann mit der Ernte begonnen werden (denken Sie an Kartoffelernte). Dies ergibt einen Kartoffelvorrat von K_0 (Zentnern Kartoffeln). Der Planungszeitraum umfasst die Ernten K_1, \dots, K_5 (in Zentnern Kartoffeln) in den folgenden fünf Jahren. Im i -ten Jahr stehen genau K_{i-1} Zentner Kartoffeln zur Verfügung, da wegen Verderblichkeit nichts länger als ein Jahr aufgehoben werden kann. Diese Kartoffeln können als Saatgut (S_i) verwendet oder verzehrt werden (V_i). Die aus dem Saatgut erzielte Ernte ist jeweils $\Phi_i \cdot S_i$, $i = 1, \dots, 5$.

Es ist vorauszusehen, dass in der Horde in den Jahren $i = 1, \dots, 5$ jeweils B_i Menschen leben werden. Die im Jahr i zum Verzehr bestimmten Kartoffeln werden auf diese B_i Personen gleichmäßig aufgeteilt. Jeder Hordenmensch hat einen jährlichen Mindestbedarf von m Zentnern Kartoffeln (damit er überlebt) und einen Maximalbedarf von M Zentnern Kartoffeln (mehr kann er nicht essen).

Die Planung soll garantieren, dass in jeder Periode i jeder Hordenmensch zwischen m und M Zentner verzehren kann, wofür der Ernteertrag aus der Vorperiode K_{i-1} zur Verfügung steht. Ferner soll die letzte Ernte einen Ertrag K_5 von mindestens $2K_0$ erbringen. Als Zielgröße betrachte man den Pro-Kopf-Verzehr in den jeweiligen Jahren. Diesen wollen wir möglichst groß werden lassen. Dazu werden zwei verschiedene Vorgehensweisen vorgeschlagen. Setzen Sie jede in eine Formulierung als lineares Optimierungsproblem um.

- (a) Hier soll die Summe aus den Pro-Kopf-Verzehren der fünf Jahre maximiert werden.
- (b) Hier soll der geringste Pro-Kopf-Verzehr in den fünf Jahren größtmöglich gestaltet werden (die Qualität bemisst sich also am schlechtesten Jahr).

2. (Gradientenverfahren)

a) Implementieren Sie das Gradientenverfahren in MATLAB in einer Datei `gradientMethod.m` unter Verwendung der Armijo-Regel zur Schrittweitensteuerung (siehe Homepage - wählen Sie `s = 1.0`) mit folgendem Funktionsaufruf

```
[X,fx,nit] = gradientMethod(f,gradf,x,sigma,beta,tol)
```

mit den Parametern

- `f` - die Funktion $f: \mathbb{R}^n \rightarrow \mathbb{R}$ als *function handle*
- `gradf` - der Gradient von f als *function handle*
- `x` - der Startpunkt $x \in \mathbb{R}^n$
- `sigma,beta` - die Konstanten $\sigma \in (0,1)$ und $\beta \in (0,1)$ für die Armijo-Regel

- `tol` - die Toleranz für das Abbruchkriterium

Zurückgegeben werden soll das Minimum $\mathbf{x} = x^*$, der optimale Funktionswert $\mathbf{fx} = f(x^*)$ und die Anzahl der Iterationsschritte `nit`.

- b) Schreiben Sie ein Skript `test_gradient.m`, um diese Methode an der Rosenbrockfunktion

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

zu testen. Wählen Sie $\mathbf{x} = [-1.9, 2]$, `tol = 1e-8`, `sigma = 0.5` und `beta = 0.5`.

3. (Newtonverfahren)

- a) Implementieren Sie das Newtonverfahren in MATLAB in einer Datei `newtonMethod.m` mit und ohne Schrittweitensteuerung (Armijo-Regel) mit folgendem Funktionsaufruf

$$[\mathbf{X}, \mathbf{fx}, \text{nit}] = \text{newtonMethod}(f, \text{gradf}, \text{hessf}, \mathbf{x}, \text{tol}, \text{sigma}, \text{beta})$$

mit den Parametern

- `f` - die Funktion $f: \mathbb{R}^n \rightarrow \mathbb{R}$ als *function handle*
- `gradf` - der Gradient von f als *function handle*
- `hessf` - die Hesse-Matrix von f als *function handle*
- `x` - der Startpunkt $x \in \mathbb{R}^n$
- `tol` - die Toleranz für das Abbruchkriterium
- **optional:** `sigma, beta` - die Konstanten $\sigma \in (0, 1)$ und $\beta \in (0, 1)$ für die Armijo-Regel

Sofern `sigma` und `beta` übergeben wird, soll die Schrittweite gesteuert werden, ansonsten nicht. Zurückgegeben werden soll der gesamte Pfad, d.h. $\mathbf{X} = [x_0, x_1, x_2, \dots, x^*]$, der optimale Funktionswert $\mathbf{fx} = f(x^*)$ und die Anzahl der Iterationsschritte `nit`.

- b) Schreiben Sie ein Skript `test_newton.m`, um diese Methode **mit** Schrittweitensteuerung an der Rosenbrockfunktion

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

zu testen. Wählen Sie $\mathbf{x} = [-1.9, 2]$, `tol = 1e-8`, `sigma = 0.5` und `beta = 0.5`.

- c) Vergleichen Sie die Konvergenzrate des Gradientenverfahren mit der Rate des Newtonverfahren für dieses Beispiel. Was fällt auf?

- d) Schreiben Sie ein Skript `test_newton2.m`, welches das Optimierungsproblem

$$\min_{x \in \mathbb{R}} f(x) := x \arctan x - \frac{1}{2} \log(x^2 + 1)$$

mittels Newtonverfahren **ohne** Schrittweitensteuerung löst. Verwenden Sie dazu `tol = 10-12` und die Startwerte $x = 0.5$ und $x = 4$. Diskutieren Sie die Ergebnisse.

- e) Schreiben Sie ein Skript `test_newton_ls.m`, dass das Optimierungsproblem aus d) mittels Newtonverfahren **mit** Schrittweitensteuerung löst und verwenden Sie $x_0 = 4$ als Startwert und `tol = 10-12`. Diskutieren Sie die numerischen Ergebnisse. Was fällt auf?

4. (Quasi-Newton-Verfahren)

Die Wahl der Suchrichtung d im Punkt x_k ist charakteristisch für numerische Optimierungsalgorithmen. Dabei haben Sie die folgenden zwei Verfahren bereits kennen gelernt:

- Gradientenverfahren: $I_{n \times n} d = -\nabla f(x_k)$
- Newtonverfahren: $\nabla^2 f(x_k) d = -\nabla f(x_k)$.

Ein Quasi-Newton-Verfahren verzichtet explizit auf die Hessematrix und approximiert diese in jedem Schritt. Demnach wird das Gleichungssystem

$$W_k d = -\nabla f(x_k)$$

gelöst, eine zulässige Schrittweite t bestimmt und $x_{k+1} = x_k + td$ berechnet. Danach wird eine Aktualisierung der Approximation der Hessematrix vorgenommen, durch folgende Rechenvorschrift nach Broyden-Fletcher-Goldfarb-Shanno

$$W_{k+1} = W_k + \frac{y_k y_k^T}{y_k^T s} - \frac{W_k (s s^T) W_k}{s^T W_k s} \quad (\text{BFGS})$$

mit $y_k := \nabla f(x_{k+1}) - \nabla f(x_k)$ und $s = td$. Die Wahl von W_0 ist dabei beliebig. Wählt man $W_0 = I_{n \times n}$, so entspricht der erste Schritt genau dem ersten Schritt des Gradientenverfahrens.

a) Implementieren Sie das Quasi-Newtonverfahren in MATLAB in einer Datei `quasiNewtonMethod.m` unter Verwendung der Armijo-Regel zur Schrittweitensteuerung mit folgendem Funktionsaufruf

```
[X,fx,nit] = quasiNewtonMethod(f,gradf,x,tol,sigma,beta)
```

mit den Parametern

- **f** - die Funktion $f : \mathbb{R}^n \rightarrow \mathbb{R}$ als *function handle*
- **gradf** - der Gradient von f als *function handle*
- **x** - der Startpunkt $x \in \mathbb{R}^n$
- **tol** - die Toleranz für das Abbruchkriterium
- **sigma,beta** - die Konstanten $\sigma \in (0, 1)$ und $\beta \in (0, 1)$ für die Armijo-Regel

Zurückgegeben werden soll der gesamte Pfad, d.h. $\mathbf{X} = [x_0, x_1, x_2, \dots, x^*]$, der optimale Funktionswert $\mathbf{fx} = f(x^*)$ und die Anzahl der Iterationsschritte **nit**. Wählen Sie für die Approximation der Hessematrix oben genannte Vorschrift (BFGS) und setzen Sie $W_0 = I_{n \times n}$.

b) Laden Sie die Datei `compareMethods.m` von der Vorlesungshomepage herunter und testen Sie das Skript mit folgenden Parametern

- $\sigma = 0.5$, $\beta = 0.5$ und $x_0 = (1.9, 2)^T$
- $\sigma = 0.9$, $\beta = 0.2$ und $x_0 = (1.9, 2)^T$
- $\sigma = 0.1$, $\beta = 0.5$ und $x_0 = (0, 2)^T$.

Diskutieren Sie die numerischen Ergebnisse und erläutern Sie die Stärken und Schwächen der einzelnen Verfahren.