Universität Hamburg
Fachbereich Mathematik
Dr. Matthias Voigt, Dr. Heiko Kröner

Sommersemester 2019

Optimierung – 6. Übungsblatt.

Bitte bearbeiten Sie die Übungsaufgaben in festen Zweiergruppen.
Abgabetermin: *Donnerstag, 23.05.2019 vor der Übung. Bitte senden Sie die Lösung der Programmieraufgabe bis zum 22.05.2019, 18:00 Uhr an* `heiko.kroener@uni-hamburg.de`*. Alle zur Ausführung nötigen Dateien sollten in einem Archiv mit dem Namen* `opt_blattnr_aufgabennr_name1_name2.zip` *enthalten sein.*

Aufgabe 1 (4 Punkte): Es sei das Minimierungsproblem

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}x^\mathsf{T}Hx + x^\mathsf{T}b$$

für eine positiv definite Matrix $H \in \mathbb{R}^{n\times n}$ und einen Vektor $b \in \mathbb{R}^n$ gegeben. Betrachten Sie für dieses Problem ein Quasi-Newton-Verfahren mit BFGS-Update und mit exakter Schrittweite (siehe Übungsblatt 4, Aufgabe 1). Zeigen Sie, daß alle Suchrichtungen *H-orthogonal* sind, d. h. daß

$$\left\langle d^i, Hd^k \right\rangle = 0$$

für $0 \leq i < k$ und $k = 0,\, 1,\, 2,\, \ldots$ gilt.

Aufgabe 2 (5 Punkte): Zeigen Sie, daß der Suchraum $V_k = \operatorname{span}\left\{d^0,\, d^1,\, \ldots,\, d^{k-1}\right\}$ aus dem CG-Verfahren (mit den Bezeichnungen aus der Vorlesung) als

$$V_k = \operatorname{span}\left\{g^0,\, Hg^0,\, \ldots,\, H^{k-1}g^0\right\} =: \mathcal{K}_k\big(H, g^0\big)$$

ausgedrückt werden kann.
Räume der Form $\mathcal{K}_m\big(A, b\big)$ mit einer Matrix $A \in \mathbb{C}^{n\times n}$ und $b \in \mathbb{C}^n$ werden als *Krylov-Unterräume* bezeichnet. Diese spielen ein herausragende Rolle bei der Lösung hochdimensionaler Probleme, insbesondere bei der Lösung sehr großer linearer Gleichungssysteme. Zeigen Sie:

a) Falls $A^{m-1}b \neq 0$ und $A^m b = 0$ für ein $m \in \mathbb{N}$ gilt, dann ist

$$\dim \mathcal{K}_j(A, b) = j, \quad j = 1,\, \ldots,\, m.$$

b) Es gilt $\mathcal{K}_j(A, b) = \mathcal{K}_j(A - \lambda I_n, b)$ für alle $\lambda \in \mathbb{C}$.

Aufgabe 3 (7 Punkte): Betrachten Sie das quadratische Optimierungsproblem

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}x^\mathsf{T}Hx + x^\mathsf{T}b$$

mit einer symmetrischen und positiv definiten Matrix $H \in \mathbb{R}^{n\times n}$ und $b \in \mathbb{R}^n$.

a) Implementieren Sie das CG-Verfahren in einer MATLAB-Funktion mit dem Interface

```
[ xmin, fmin ] = cg( x0, H, b )
```

mit den folgenden Argumenten:

- `x0`: Startpunkt x^0 der Iteration,
- `H, b`: die Matrix H und der Vektor b,
- `xmin, fmin`: der Minimierer und der minimale Funktionswert.

Nutzen Sie das `sparse`-Speicherformat um H zu speichern. Testen Sie Ihre Implementierung anhand der Daten

$$H = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{10000\times 10000}, \quad b = \begin{bmatrix} 1 \\ \vdots \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^{10000}.$$

Achten Sie bei Ihrer Implementierung auf Effizienz.

b) Bei schlecht konditionierten Problemen konvergiert das CG-Verfahren oft nur sehr langsam. Man löst daher stattdessen ein vorkonditioniertes Optimierungsproblem

$$\min_{z\in\mathbb{R}^n} \frac{1}{2} z^\mathsf{T} L^{-1} H L^{-\mathsf{T}} z + z^\mathsf{T} L^{-1} b$$

mit $x = L^{-\mathsf{T}} z$. Formulieren Sie einen effizienten Algorithmus, der das vorkonditionierte CG-Verfahren ausführt. Implemtieren Sie diesen in einer MATLAB-Funktion

```
[ xmin, fmin ] = pcg( x0, H, b )
```

mit denselben Argumenten wie oben. Nutzen Sie als Vorkondionierer die unvollständige Cholesky-Zerlegung, die mit dem MATLAB-Befehl `ilu` bestimmt werden kann. Testen Sie diese Funktion an denselben Daten wie in a) und vergleichen Sie mit dem Standard-CG-Verfahren, indem Sie die Normen der Residuen $\|Hx^k + b\|$ in Abhängigkeit von k in einem gemeinsamen Plot darstellen.

Aufgabe 4 (4 Punkte): Das CG-Verfahren kann auch auf allgemeine nichtlineare Probleme erweitert werden. Der wesentliche Unterschied ist, daß g^k im allgemeinen Fall durch $\nabla f(x^k)$ ersetzt werden muß und nicht einfach durch $Hx^k + b$ ausgedrückt werden kann. Die Verfahrensvorschrift ist wie folgt:

1. Wähle einen Startpunkt $x^0 \in \mathbb{R}^n$ und eine Toleranz $\varepsilon > 0$ und setze $d^0 := -\nabla f(x^0)$ und $k := 0$.
2. Ist $\|\nabla f(x^0)\| \le \varepsilon$, dann Abbruch.
3. Berechne eine effiziente Schrittweite σ_k,

$$x^{k+1} := x^k + \sigma_k d^k, \quad \beta_k := \frac{\|\nabla f(x^{k+1})\|}{\|\nabla f(x^k)\|}, \quad d^{k+1} := -\|\nabla f(x^{k+1})\| + \beta_k d^k.$$

4. Setze $k := k + 1$ und gehe zu 2.

Implementieren Sie diesen als *Fletcher-Reeves-Verfahren* bekannten Algorithmus in einer MATLAB-Funktion mit dem Interface

```
[ xmin, fmin ] = fletcherreeves( x0, fun, grad )
```

mit den folgenden Argumenten:

- `x0`: Startpunkt x^0 der Iteration,
- `fun, grad`: zwei Function Handles zur Auswertung der Funktionen f und ∇f,
- `xmin, fmin`: der Minimierer und der minimale Funktionswert.

Testen Sie Ihre Implementierung an der Rosenbrock-Funktion von Übungsblatt 1 mit dem Startpunkt $x^0 = \left[\begin{smallmatrix} -1.9 \\ 2 \end{smallmatrix}\right]$. Wählen Sie zur Schrittweitensteuerung das Powell-Verfahren mit den Parametern $\beta = 0.9$ und $\delta = 0.1$ und als Startschrittweite $\sigma_{k,0} = 1$. Illustrieren Sie die Konvergenz des Verfahrens, indem Sie die Iterierten und die Niveaulinien der Rosenbrock-Funktion in einem gemeinsamen Plot darstellen.