Einführung in die Numerische Mathematik

Sommersemester 2022
Professor Dr. Carsten Burstedde
Tim Griesbach

UNIVERSITÄT BONN

Übungsblatt 6.

Abgabe am **Donnerstag, 19.05.2022**.

Aufgabe 1. (Vorkonditioniertes CG-Verfahren)

Für ein lineares Gleichungssystem $Ax = b$ mit $A \in \mathbb{R}^{d\times d}$ symmetrisch und positiv definit betrachten wir einen Vorkonditionierer $C \in \mathbb{R}^{d\times d}$, der ebenfalls symmetrisch und positiv definit ist. Zeigen Sie, dass das vorkonditionierte CG-Verfahren (siehe Algorithmus 1) dem nicht vorkonditionierten CG-Verfahren angewandt auf das lineare Gleichungssystem $C^{\frac{1}{2}}AC^{\frac{1}{2}}y = C^{\frac{1}{2}}b$ mit $y = C^{-\frac{1}{2}}x$ entspricht.

(6 Punkte)

Aufgabe 2. (Affin invariantes BFGS-Newton-Verfahren)

Es sei $A \in \mathbb{R}^{d\times d}$ symmetrisch positiv definit und $b, c \in \mathbb{R}^d$. Zudem sei $C \in \mathbb{R}^{d\times d}$ invertierbar. Wir betrachten die Funktionen

$$f(x) = \frac{1}{2}x^\top Ax - x^\top b\,, \quad g(x) = f(Cx + c)\,.$$

Zeigen Sie, dass das BFGS-Newton-Verfahren *affin invariant* ist: Wendet man das BFGS-Newton-Verfahren mit Startmatrix $B^{(0)} = CC^\top$ auf die Funktion f an, und wendet man das BFGS-Newton-Verfahren mit Startmatrix $D^{(0)} = I_d$ auf die Funktion g an, so gilt für die Iterierten

$$\forall k \in \mathbb{N} \cup \{0\}: \quad x^{(k)} = Cy^{(k)} + c\,,$$

sofern dies auch für die Startpunkte ($k = 0$) gilt.

Algorithmus 1: PCG ohne Abbruchkriterium für $Ax = b$

Eingabe: $C, A \in \mathbb{R}^{d\times d}$, $x_0 \in \mathbb{R}^d$ und $b \in \mathbb{R}^d$

$r_0 \leftarrow b - Ax_0$
$h_0 \leftarrow Cr_0$
$d_0 \leftarrow h_0$
for $k = 0, 1, \ldots$ **do**
  $z \leftarrow Ad_k$
  $\alpha_k \leftarrow \frac{r_k^T h_k}{d_k^T z}$
  $x_{k+1} \leftarrow x_k + \alpha_k d_k$
  $r_{k+1} \leftarrow r_k - \alpha_k z$
  $h_{k+1} \leftarrow Cr_{k+1}$
  $\beta_k \leftarrow \frac{r_{k+1}^T h_{k+1}}{r_k^T h_k}$
  $d_{k+1} \leftarrow h_{k+1} + \beta_k d_k$
end

(6 Punkte)

Aufgabe 3. (Symmetrie DFP- und BFGS-Verfahren)

Es wird die gleiche Notation wie in Aufgabe 2, Übungsblatt 5 verwendet. Zeigen Sie, dass für $B_{k+1}^{\mathrm{BFGS}} = (H_{k+1}^{\mathrm{BFGS}})^{-1}$ die inverse Aufdatierungsformel

$$B_{k+1}^{\mathrm{BFGS}} = B_k + \frac{(d_k - B_k y_k)d_k^\top + d_k(d_k - B_k y_k)^\top}{d_k^\top y_k} - \frac{(d_k - B_k y_k)^\top y_k}{(d_k^\top y_k)^2} d_k d_k^\top$$

gilt. Dabei sei H_k symmetrisch positiv definit, $B_k = H_k^{-1}$ und $d_k^\top y_k > 0$ vorausgesetzt.

(6 Punkte)

Programmieraufgabe 4. (BFGS-Verfahren)

In dieser Aufgabe soll das lokal inverse BFGS-Verfahren im $\mathbb{R}^d$ implementiert werden, wobei die Approximationen der inversen Hesse-Matrix nicht abgespeichert werden sollen, sondern direkt auf $\nabla f(x_k)$ angewendet werden sollen. Des Weiteren werden wir das BFGS-Verfahren so implementieren, dass es einen von der Anzahl der Iterationen unabhängigen Speicherbedarf hat.

(a) Zeigen Sie, dass die Aufdatierungsformel aus der Aufgabe 3 auf diesem Blatt mit d_k als Bezeichnung für die Suchrichtungen schreiben können als

$$B_{k+1}^{\mathrm{BFGS}} = V_k^T B_k V_k + \gamma_k d_k d_k^T$$

für $V_k \coloneqq I_d - \gamma_k y_k d_k^T$ mit I_d die d-dimensionale Einheitsmatrix und $\gamma_k \coloneqq \frac{1}{y_k^T d_k}$.

(b) Wir wollen die in der Teilaufgabe (a) hergeleitet Iterationsvorschrift verwenden, um die Anwendung von B_k auf einen Vektor mithilfe der Speicherung von y_k und d_k für $k \in \{k-m, \ldots, k-1\}$ mit $m \in \mathbb{N} \setminus \{0\}$ zu realisieren. Die wiederholte Anwendung der Iterationsvorschrift aus der Teilaufgabe (a) liefert Algorithmus 2. Hierbei wird für jede Iteration eine Startmatrix $B_k^0 = \xi_k I_d$ benötigt, die aber nicht gespeichert werden soll, sondern nur durch die entsprechende Skalierung implementiert wird. Hierbei wählen wir für $k \in \mathbb{N} \setminus \{0\}$

$$\xi_0 \coloneqq \frac{r^T r}{r^T \mathrm{diag}(H) r} \text{ mit } r = \nabla f(x_0) - \mathrm{diag}(H) x_0 \quad \text{und} \quad \xi_k \coloneqq \frac{d_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}},$$

wobei $\mathrm{diag}(H)$ die Diagonale der Hesse-Matrix der Zielfunktion ausgewertet in x_0 ist. Beweisen Sie, dass man durch die wiederholte Anwendung der in Teilaufgabe (a) gezeigten Iterationsvorschrift den Algorithmus 2 zur Anwendung der Approximation der inversen Hesse-Matrix auf $\nabla f(x_k)$ herleiten kann.

(c) Implementieren Sie mithilfe von Algorithmus 2 das lokal inverse BFGS-Verfahren, wobei die Suchrichtung mit $\sigma_k \in (0, 1]$ (Schrittweite) skaliert werden soll. Hierbei wird $\sigma_k \in \{1, \delta, \delta^2, \ldots\}$ minimal gewählt, sodass

$$\begin{aligned} f(x_k + \sigma_k d_k) &\leq f(x_k) + c_1 \sigma_k \nabla f(x_k)^T d_k, \\ \nabla f(x_k + \sigma_k d_k)^T d_k &\geq c_2 \nabla f(x_k)^T d_k, \end{aligned}$$

mit $c_1 = 10^{-4}$ und $c_2 = 0.9$ gilt. Wir wählen $\delta = 0.8$.

(d) Testen Sie Ihre Implementierung mit der Funktion

$$f(x, y) = (x-2)^4 + y^2(x-2)^2 + (y+1)^2,$$

wobei Sie als Startwert $x_0 = (1, 1)^T$ wählen und in jeder Iteration $B_0 = I_2$ verwenden. Plotten Sie den Fehler $\|x_k - x^*\|_2$ indem Ihr Programm Textdatei erstellt. Diese können Sie dann mit einem Programm Ihrer Wahl (z. B. Gnuplot oder Matplotlib in Python) plotten. Welche Konvergenzgeschwindigkeit erhalten Sie?

Algorithmus 2: L-BFGS

Eingabe: $\nabla f(x_k)$, $\{d_i, y_i\}_{i=k-m,\dots k-1}$ und $B_k^0 \in \mathbb{R}^{d\times d}$

$q \leftarrow \nabla f(k_k)$
for $j = k-1, k-2, \dots, k-m$ **do**
  $i \leftarrow \max(0, j)$
  $\alpha_i \leftarrow \gamma_i d_i^T q$
  $q \leftarrow q - \alpha_i y_i$
end
$r \leftarrow B_k^0 q$
for $j = k-m, k-m+1, \dots, k-1$ **do**
  $i \leftarrow \max(0, j)$
  $\beta \leftarrow \gamma_i y_i^T r$
  $r \leftarrow r + d_i(\alpha_i - \beta)$
end
return r

(1 + 2 + 5 + 4 = 12 Punkte)

Die Programmieraufgabe kann bis zum 26.05.2022 im PC-Pool des Nebengebäudes des Mathematikzentrums abgegeben werden. Bitte vereinbaren Sie mit Ihrer Tutorin bzw. Ihrem Tutor für die Abgabe der Programmieraufgabe einen Termin innerhalb des oben genannten Zeitrahmens Die Sprache der Programmieraufgaben ist C oder auch C++, solange der mathematische Kern C-kompatibel bleibt. Die Lösungen müssen auf den Computern des PC-Pools präsentiert werden.