2. Projekt

Quasi-Newton-Verfahren & Gauß-Newton-Verfahren

im Fach

Numerische Optimierung

Juni 2020

Maximilian Gaul

Aufgabe 1

Siehe Programmcode in `Project2.m`.

Aufgabe 2

$$I-\frac{s(As)^T}{s^TAs}+\frac{A^{-1}yy^T}{y^Ts}+\frac{(s-A^{-1}y)s^TA+s(s-A^{-1}y)^TA}{y^Ts}-\frac{(s-A^{-1}y)s^TAs(As)^T+s(s-A^{-1}y)^TAs(As)^T}{y^Tss^TAs}+\frac{(s-A^{-1}y)s^Tyy^T+s(s-A^{-1}y)^Tyy^T}{(y^Ts)^2}-\frac{(s-A^{-1}y)^Tyss^TA}{(y^Ts)^2}+\frac{(s-A^{-1}y)^Tyss^TAs(As)^T}{(y^Ts)^2s^TAs}-\frac{(s-A^{-1}y)^Tyss^Tyy^T}{(y^Ts)^2y^Ts}$$

$$I-\frac{ss^TA}{s^TAs}+\frac{A^{-1}yy^T}{y^Ts}+\frac{ss^TA-A^{-1}ys^TA+s(s^T-(A^{-1}y)^T)A}{y^Ts}-\frac{ss^TAss^TA-A^{-1}ys^TAss^TA+s(s^T-(A^{-1}y)^T)Ass^TA}{y^Tss^TAs}+\frac{ss^Tyy^T-A^{-1}ys^Tyy^T+s(s^T-(A^{-1}y)^T)yy^T}{(y^Ts)^2}-\frac{(s^T-(A^{-1}y)^T)yss^TA}{(y^Ts)^2}+\frac{(s^T-(A^{-1}y)^T)yss^TAss^TA}{(y^Ts)^2s^TAs}-\frac{(s^T-(A^{-1}y)^T)yss^Tyy^T}{(y^Ts)^2y^Ts}$$

$$I-\frac{ss^TA}{s^TAs}+\frac{A^{-1}yy^T}{y^Ts}+\frac{2ss^TA-A^{-1}ys^TA-sy^T}{y^Ts}-\frac{2ss^TA-A^{-1}ys^TA-sy^T}{y^Tss^TAs}\cdot ss^TA+\frac{2ss^T-A^{-1}ys^T-sy^T(A^{-1})^T}{(y^Ts)^2}\cdot yy^T-\frac{s^Ty-y^T(A^{-1})^T}{(y^Ts)^2}\cdot ss^TA+\frac{(s^T-y^T(A^{-1})^T)yss^TAss^TA}{(y^Ts)^2s^TAs}-\frac{(s^T-y^T(A^{-1})^T)yss^Tyy^T}{(y^Ts)^2y^Ts}$$

Aufgabe 3

Wenn die Suchrichtung des BFGS Verfahrens:

$$d = -B \cdot \nabla f(x)$$

keine Abstiegsrichtung ist, d.h. die Bedingung:

$$\nabla f(x)^T \cdot d < 0$$

nicht erfüllt ist, muss das Verfahren 'resettet' werden. In diesem Fall bietet es sich an, die Suchrichtung auf den negativen Gradienten zu setzen:

$$d = -\nabla f(x)$$

Da nun die Abstiegsrichtung nicht mehr zur approximierten Inversen der Hesse-Matrix B passt, muss diese ebenfalls für den nächsten Schritt neu bestimmt werden. Hierzu bieten sich verschiedene Möglichkeiten an:

- Wie beim Start des BFGS-Verfahrens $B = I$ setzen
  - Hierbei geht jeglicher berechnete Fortschritt verloren, es handelt sich um einen recht naiven Ansatz
- Die Hesse-Matrix einmalig aus Differenzenquotienten des Gradienten bestimmen und anschließend invertieren
  - Hoher Rechenaufwand von $\mathcal{O}(n^2)$ für die Hesse-Matrix und nochmal $\mathcal{O}(n^3)$ für das Invertieren
  - Problem wenn die Hesse-Matrix nicht invertierbar ist bzw. aufgrund von Auslöschung oder anderen numerischen Fehlern die Inverse schlecht konditioniert ist
  - Bisher berechneter Fortschritt geht ebenfalls verloren aber die Approximation der Hesse-Matrix ist sehr genau
- Man könnte, wie in den Vorlesungsfolien beschrieben, $\frac{y^T s}{y^T y} \cdot I_n$ als positivdefinitve Matrix verwenden

Um herauszufinden, welche dieser Methoden am besten geeignet ist (d.h. die richtig Lösung in der kürzesten Zeit findet), wird die Laufzeit des inversen BFGS-Verfahrens bestimmt. Startwerte:

- N-dim. Rosenbrock-Funktion: $\begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}$
- Himmelblau-Funktion: $\begin{bmatrix} 0 \\ -1 \end{bmatrix}$

- Bazaraa-Shetty: $\begin{bmatrix} 0 \\ -1 \end{bmatrix}$

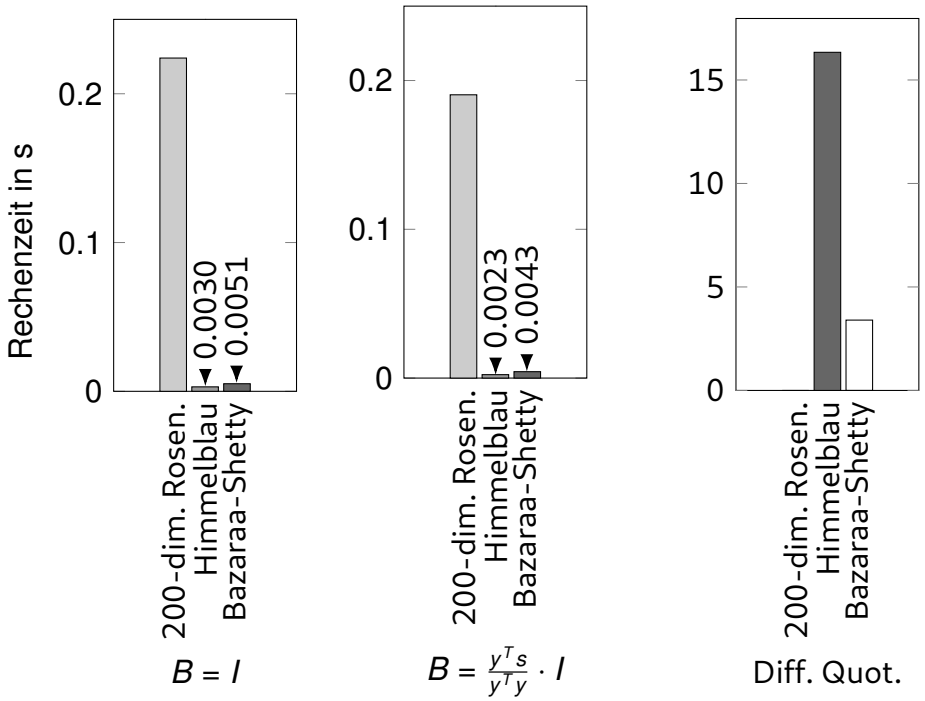

Abbildung 1: Vergleich der Rechenzeit für eine Genauigkeit von 10^{-8} gemittelt über 100 Durchläufe (Intel Core i3-7100U, 8GB RAM, Windows 10 64-Bit)

Alle drei Funktionen profitieren von $B = \frac{y^T s}{y^T y} \cdot I$, daher habe ich diesen Weg in meiner Implementierung gewählt.

Aufgabe 4

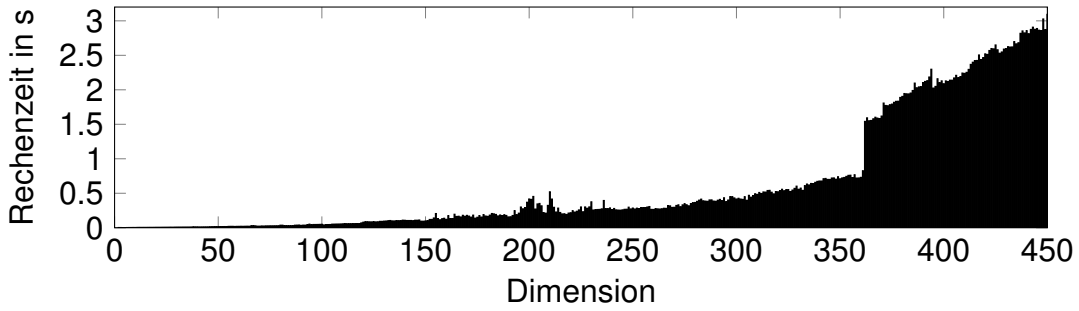

Abbildung 2: Rechenzeit der N-dimensionalen Rosenbrock-Funktion mit Startwert $[-1, \ldots, -1]^T$ für eine Genauigkeit von 10^{-8} gemittelt über 100 Durchläufe (Intel Core i3-7100U, 8GB RAM, Windows 10 64-Bit)

Man erkennt, das mit höheren Dimensionen die Rechenzeit drastisch zunimmt.