

2 Statische Optimierung mit Beschränkungen

In dieser Übung wird das statische Optimierungsproblem

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) \\ \text{u.B.v. } \mathbf{g}(\mathbf{x}) = \mathbf{0} \\ \mathbf{h}(\mathbf{x}) \leq \mathbf{0} \end{aligned} \quad (2.1)$$

betrachtet, d. h. es soll das Minimum der Kostenfunktion $f(\mathbf{x})$ bezüglich der Optimierungsvariablen \mathbf{x} unter Gleichungsbeschränkungen $\mathbf{g}(\mathbf{x})$ und Ungleichungsbeschränkungen $\mathbf{h}(\mathbf{x})$ ermittelt werden.

Zur Vorbereitung auf die Übung werden folgende Punkte empfohlen:

1. Lesen Sie die Theorie zu beschränkten Optimierungsproblemen, allen voran die Kapitel welche das SQP-Verfahren sowie die reduzierte Gradientenmethode behandeln.
2. Machen Sie sich mit dem MATLAB-Befehl `quadprog` zur Lösung von beschränkten quadratischen Optimierungsproblemen vertraut.
3. Machen Sie sich mit dem MATLAB-Befehl `fmincon` zur Lösung von beschränkten Optimierungsproblemen und den darin verwendeten Algorithmen vertraut.
4. Implementieren Sie eine Funktion `[f,df,ddf]=calcf_styblinski_tang_n(x)` welche den Funktionswert f , den Gradienten df sowie die Hessematrix ddf der n -dimensionalen *Styblisky-Tang*-Funktion

$$f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^n (x_i^4 - 16x_i^2 + 5x_i) \quad (2.2)$$

ausgewertet an der Stelle \mathbf{x} liefert.

Hinweis: Eine effiziente Implementierung ist mittels der MATLAB-Befehle `sum`, `diag` und der elementweisen Potenzfunktion `.^` möglich.

Folgende Aufgaben sollen in der Übungseinheit gelöst werden:

1. Erstellen Sie eine MATLAB-Funktion `xmin = fminsqp(fun, x0, constrfun)` welche das Optimierungsproblem (2.1) mittels des SQP-Verfahrens löst. Verwenden Sie *Function Handles* um die Kostenfunktion `fun` sowie die Beschränkungsfunktion

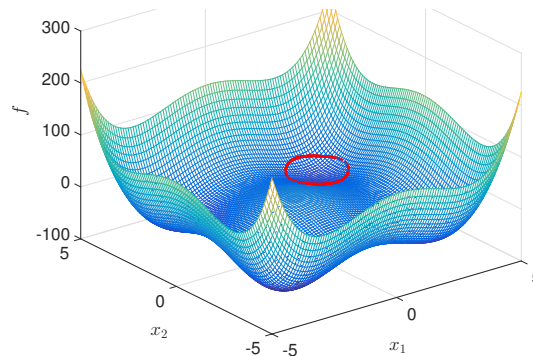


Abbildung 2.1: *Styblinski-Tang*-Funktion mit dem Rand der Beschränkung (2.3).

`constrfun` zu übergeben. Implementieren Sie `fminsqp`, sodass es mit einer skalaren Gleichungsbeschränkung und einer skalaren Ungleichungsbeschränkung umgehen kann.

Die Beschränkungsfunktion soll in der mit dem MATLAB-Befehl `fmincon` kompatiblen Form `[h,g,dh,dg,ddh,ddg]=constrfun(x)` implementiert werden. Die Rückgabewerte `h`, `dh` und `ddh` bezeichnen den Funktionswert, Gradienten und die Hessematrix der skalaren Ungleichungsbeschränkung h ausgewertet an der Stelle \mathbf{x} und die Rückgabewerte `g`, `dg` und `ddg` bezeichnen die entsprechenden Größen der skalaren Gleichungsbeschränkung g .

Hinweis: Verwenden Sie für den Aufruf von `quadprog` den Algorithmus `interior-point-convex`. Prüfen Sie vor der Ausführung von `quadprog`, ob die Hessematrix der Lagrangefunktion positiv definit ist. Sollte dies nicht der Fall sein, wenden Sie eine geeignete Methode an um dies numerisch zu korrigieren.

2. Testen Sie Ihre Funktion `fminsqp` anhand der *Styblinski-Tang*-Funktion (2.2) für $n = 2$ unter Beachtung der skalaren Ungleichungsbedingung

$$h(\mathbf{x}) = (x_1 - 2.75)^2 + (x_2 - 2.75)^2 \geq 1, \quad (2.3)$$

deren Rand in Abbildung 2.1 dargestellt ist. Implementieren Sie (2.3) in der MATLAB-Funktion `[h,g,dh,dg,ddh,ddg]=calchg(x)`. Da keine Gleichungsbeschränkung erfüllt werden muss, soll `calchg` für die Gleichungsbedingung und ihre Ableitungen Null-Matrizen geeigneter Dimension liefern.

Vergleichen Sie die Ergebnisse Ihrer Implementierung mit dem Algorithmus `sqp` der MATLAB-Funktion `fmincon`. Verwenden Sie hierbei verschiedene Startwerte und lassen Sie sich den Verlauf der Iterationen in die Kostenfunktion einzeichnen.

- Konvergieren die Verfahren zu einem globalen/lokalen Minimum?
- Werden die Beschränkungen eingehalten?
- Welche Methode weist die besseren Konvergenzeigenschaften auf? Wie könnte man die Konvergenz verbessern?

3. Eine große Zahl an Gleichungsbeschränkungen legt nahe, dass ein Minimierungsproblem (2.1) effizient mittels der reduzierten Gradientenmethode erfolgen kann. Betrachten Sie nun die *Styblinski-Tang*-Funktion (2.2) mit der Dimension $n = 20$ und den Gleichungsbeschränkungen

$$\left. \begin{array}{l} x_3 = \frac{1}{2}(x_1 + x_2) \\ x_4 = \frac{1}{2}(x_1 + x_3) \\ \vdots \\ x_n = \frac{1}{2}(x_1 + x_{n-1}) \end{array} \right\} \bar{p} = n - 2. \quad (2.4)$$

Implementieren Sie ein MATLAB-Skript, welches die Kostenfunktion (2.2) unter der Bedingung von (2.4) mittels der reduzierten Gradientenmethode minimiert. Lösen sie hierbei das unterlagerte Liniensuchproblem mit Hilfe der MATLAB-Routine `fminsearch`. Für die Partitionierung in unabhängige \mathbf{x}_I und abhängige \mathbf{x}_D Variablen ist die Wahl

$$\mathbf{x}_I = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{und} \quad \mathbf{x}_D = \begin{bmatrix} x_3 \\ \vdots \\ x_n \end{bmatrix} \quad (2.5)$$

geeignet.

Hinweis: Die Matrix $\mathbf{A}_D = \frac{\partial \mathbf{g}}{\partial \mathbf{x}_D}$ kann effizient mittels der MATLAB-Routine `diag` generiert werden. Beachten Sie hierzu das zweite Argument des `diag` Befehls.

Hinweis: Die Transformation $\mathbf{x}_D = \mathbf{g}_D^{-1}(\mathbf{x}_I)$ kann in einer separaten MATLAB-Funktion `xD = gDinv(xI)` einfach mittels wiederholtem Einsetzen der Zwischenergebnisse x_3, \dots, x_{n-1} in (2.4) ausgewertet werden.