1 Statische Optimierung ohne Beschränkungen

Diese Übung beschäftigt sich mit zwei Themengebieten: Zuerst soll die Genauigkeit numerischer Differenzierungsalgorithmen am Beispiel des zentralen Differenzenquotienten und der komplexen Funktionsauswertung untersucht werden. Weiters soll das Minimum einer Kostenfunktion $f(\mathbf{x})$ bezüglich der Optimierungsvariable $\mathbf{x}$ in einem unbeschränkten Gebiet gesucht werden. Hierzu soll die *Newton*-Methode in MATLAB implementiert und anschließend anhand von zwei Testfunktionen mit den von der *Optimization Toolbox* zur Verfügung gestellten Algorithmen verglichen werden.

Diese Übung ist in zwei Teile gegliedert. Der erste Teil besteht aus vorbereitenden Aufgaben welche *vor der Übungseinheit* bearbeitet werden sollen. Im zweiten Teil wird eine Aufgabe zum numerischen Differenzieren und ein unbeschränktes Optimierungsproblem gelöst. Diese Aufgaben werden *in der Übungseinheit* bearbeitet.

<u>Bearbeiten Sie die folgenden Aufgaben als Vorbereitung für die Übungseinheit:</u>

1. Implementieren Sie die *Booth*-Funktion

   $$f_{\text{Booth}}(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

   und die *Styblinski-Tang*-Funktion

   $$f_{\text{Styblinski-Tang}}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}\Big(x_i^4 - 16x_i^2 + 5x_i\Big)$$

   für $n = 2$ in MATLAB als separate Funktionen `[f,df,ddf]=calcf_booth(x)` bzw. `[f,df,ddf]=calcf_styblinski_tang(x)`, welche den jeweiligen Funktionswert `f`, den analytisch berechneten Gradienten `df` und die analytisch berechnete Hessematrix `ddf` liefern. Stellen Sie diese beiden Funktionen, wie in Abbildung 1.1 gezeigt, mit dem MATLAB-Befehl `mesh` dar. Zeichnen Sie des Weiteren mit Hilfe des `plot3`-Befehls eine Linie in den Graphen ein.

   Veranschaulichen Sie sich darüber hinaus die geometrische Form der Funktionen über eine Darstellung der Höhenlinien mit dem MATLAB-Befehl `contour`. Der MATLAB-Befehl `meshc` vereint die beiden Befehle `mesh` und `contour`.

2. Machen Sie sich mit der *Optimization Toolbox*, allen voran mit dem Befehl `fminunc`, vertraut. Studieren Sie dazu die MATLAB-Dokumention dieses Befehls.

3. Studieren Sie die Theorie zur Ableitungsberechnung mit Differenzenquotienten, dem komplexen Differenzieren, sowie zur *Newton*-Methode und zu Liniensuchverfahren.

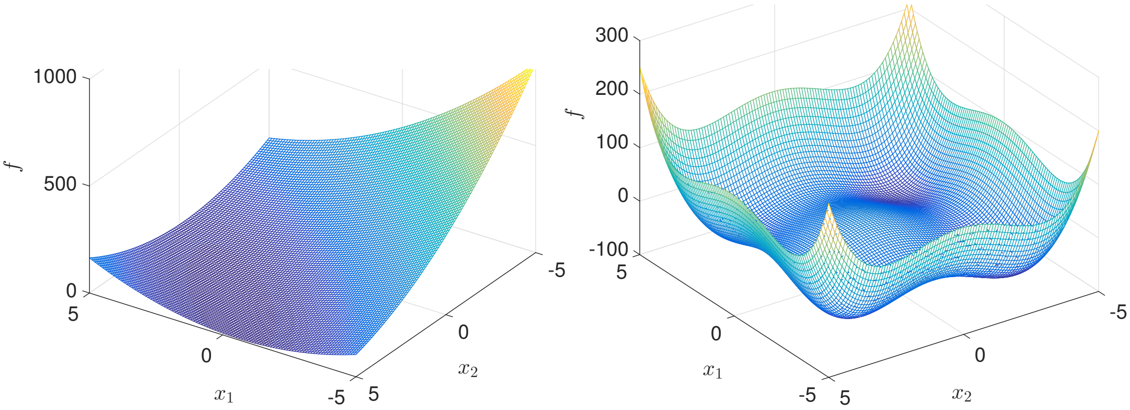

Abbildung 1.1: Booth-Funktion und Styblinski-Tang-Funktion.

Die folgenden Aufgaben sind in der Übungseinheit zu lösen:

1. Implementieren Sie die beiden Unterprogramme `[f,df]=diff_central(fun,x,h)` und `[f,df]=diff_complex(fun,x,h)`, welche den Gradienten einer `fun` mittels zentralem Differenzenquotienten bzw. durch komplexe Funktionsauswertung einer Funktion ermittelt. Beachten Sie dabei, dass das Argument `x` ein Vektor ist. Die Unterprogramme sollen den Funktionswert `f` sowie den Gradienten `df` mit der Schrittweite `h` an der Stelle `x` ermitteln, wobei die Funktion `fun` als *Function Handle* (`@`-Symbol in Matlab) übergeben wird.

   Die Genaugikeit der numerischen Algorithmen wird mit Hilfe der Richtungsableitung der Funktion $f_{\text{Styblinski-Tang}}(\mathbf{x})$ entlang einer Kurve $\mathbf{x}(t) \in \mathbb{R}^2$ gemäß

   $$d(t) = \nabla f_{\text{Styblinski-Tang}}(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}(t)} \cdot \mathbf{x}'(t) \tag{1.1}$$

   mit $\mathbf{x}'(t) = \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}$ beurteilt. Wählen Sie $\mathbf{x}(t)$ als Gerade $\mathbf{x}(t) = \mathbf{a} + \mathbf{b}t$ mit $\mathbf{a} = [-5, -5]^{\mathrm{T}}$ und $\mathbf{b} = \frac{1}{\sqrt{2}}[1, 1]^{\mathrm{T}}$.

   Berechnen Sie den Gradienten $\nabla f_{\text{Styblinski-Tang}}(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}(t)}$ für die Richtungsableitung (1.1) sowohl analytisch, als auch mit den Unterprogrammen `diff_central` und `diff_complex`. Die Richtung der Gerade $\mathbf{x}'(t) = \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}$ in (1.1) soll analytisch berechnet werden.

   Berechnen Sie weiters die absoluten Fehler zwischen der analytischen und den numerisch berechneten Richtungsableitungen für verschiedene Schrittweiten $h \in [10^{-9}, 10^{-1}]$ an 10 Punkten im Intervall $t \in [0, 10\sqrt{2}]$ und mitteln Sie diese für jede Schrittweite.

   - Wie verhält sich der Fehler für kleiner werdende Schrittweiten h?
   - In welchen Bereichen ist der Abschneidefehler bzw. der Rundungsfehler dominant? Wo tritt ein Auslöschungsfehler auf?

2. Programmieren Sie eine Funktion `[x,fval]=newton(fun,x0)`, die mittels der *Newton*-Methode das Minimum einer Kostenfunktion ausgehend vom Startwert $\mathbf{x}_0$

berechnet. Dabei soll die Kostenfunktion als *Function Handle* und der Startwert als Vektor `x0` übergeben werden. Wählen Sie eine konstante Schrittweite $\alpha_k = 1$ (ungedämpfte Newton-Methode) und zeichnen Sie die Iterationsschritte als Linienzug in einem dreidimensionalen Plot ein.

> **Hinweis:** Benutzen Sie hier den analytisch berechneten Gradienten sowie die analytisch berechnete Hessematrix.

3. Testen Sie Ihre Methode anhand der *Booth*-Funktion und der *Styblinski-Tang*-Funktion. Starten Sie die Iterationen jeweils an den Punkten $\mathbf{x}_0 = [5, 0]^\mathrm{T}$ bzw. $\mathbf{x}_0 = [5, 5]^\mathrm{T}$.
   - Konvergiert die Methode zu einem globalen Minimum?
   - Wieso werden ggf. je nach Startwert unterschiedliche Punkte erreicht?
   - Wie kann das Konvergenzverhalten verbessert werden?

   > **Hinweis:** Das globale Minimum befindet sich bei $\mathbf{x} = [1, 3]^\mathrm{T}$ für die Booth-Funktion und bei $\mathbf{x} = [-2.903534, -2.903534]^\mathrm{T}$ für die Styblinski-Tang-Funktion.

4. Vergleichen Sie Ihre Ergebnisse mit der von MATLAB zur Verfügung gestellten Funktion `fminunc` unter Verwendung der *Methode der Vertrauensbereiche* (*Trust-Region Method*). Achten Sie auf eine korrekte Übergabe der Optionen, damit tatsächlich diese Methode verwendet wird.

   > **Hinweis:** Setzen Sie für den MATLAB-Befehl `fminunc` die Option `Display` auf `'iter'` (mit Hilfe von `optimoptions`). Damit wird der Fortschritt der Optimierung nach jedem Iterationsschritt des Algorithmus ausgegeben.