Programmierblatt 3.

Besprechungswoche: **01.12. – 05.12.2025**

Optimierungsprobleme treten in zahlreichen Anwendungen auf, seien sie wissenschaftlicher, technischer oder wirtschaftlicher Natur. Viele Probleme aus der Physik lassen sich auf Energieminimierungsprobleme zurückführen, während Finanzdienstleister an der Maximierung eines Ertrages interessiert sind. Weiter sind Optimierungsverfahren notwendig in bildgebenden Verfahren, welche in der Medizin und der Geologie aber zum Beispiel auch zur Qualitätskontrolle in der Produktion verwendet werden, und sind auch ein grundlegender Baustein im Maschinellen Lernen. Aus mathematischer Sicht spielt es dabei keine Rolle, ob eine Funktion minimiert oder maximiert wird, da das Minimieren einer Funktion $f : \mathbb{R}^n \to \mathbb{R}$ äquivalent zum Maximieren der Funktion $g = -f$ ist.

Wir werden auf diesem Programmierblatt verschiedene Algorithmen betrachten und ihr Verhalten anhand von bekannten Testfunktionen beleuchten. Beispielsweise ist die skalierte *Rosenbrock-Funktion*

$$f_R(\mathbf{z}) = s\left[(a - z_1)^2 + b(z_2 - z_1^2)^2\right] \tag{1}$$

mit $s, a, b > 0$, anspruchsvoll zu minimieren, da deren globales Minimum $\mathbf{x} = [1, 1]^\intercal$ in einem schmalen, parabolischen Bereich liegt, was die Minimierung besonders anspruchsvoll macht. Andererseits ist auch die skalierte *Ackley-Funktion*

$$f_A(\mathbf{z}) = s\left[a - a\exp\left(-b\sqrt{\frac{z_1^2 + z_2^2}{2}}\right) + e - \exp\left(\frac{\cos(cz_1) + \cos(cz_2)}{2}\right)\right] \tag{2}$$

mit $s, a, b, c > 0$, anspruchsvoll zu minimieren, da die Funktion neben dem eigentlichen, globalen Minimum $\mathbf{x} = [0, 0]^\intercal$ viele weitere lokale Minima aufweist.

Abstiegsverfahren

Ausgehend von einem Startpunkt $\mathbf{x}_0$ versuchen wir, die Funktion iterativ zu minimieren, wobei wir eine Folge Iterierter $(\mathbf{x}_k)_k$ erhalten, welche der Rekursion

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k, \qquad k = 0, 1, \ldots$$

genügen. Dabei sind $\alpha_k > 0$ Schrittweiten und $\mathbf{d}_k$ sinnvoll gewählte Suchrichtungen. Hierbei soll $\mathbf{d}_k$ stets eine Abstiegsrichtung sein, was heisst, dass wir für alle $k \geq 0$ die Bedingung $f'(\mathbf{x}_k)\mathbf{d}_k < 0$ fordern.

Klassische Armijo-Liniensuche

In den Abstiegsverfahren muss zu dem Punkt $\mathbf{x}_k$ und einer gegebenen Suchrichtung $\mathbf{d}_k$ die Schrittweite

$$\alpha_k \approx \arg\min_{t \in \mathbb{R}} f(\mathbf{x}_k + t\mathbf{d}_k)$$

näherungsweise berechnet werden. Ausgehend von $\alpha_k = 1$ halbiert man α_k so lange, bis

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \leq f(\mathbf{x}_k) + \sigma\alpha_k f'(\mathbf{x}_k)\mathbf{d}_k$$

gilt; dabei ist $\sigma \in (0, 1)$ ein fest gewählter Parameter, der steuert, ob der effektive Abstieg

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) - f(\mathbf{x}_k)$$

mindestens dem von der Linearisierung erwartetem Abstieg

$$\alpha_k f'(\mathbf{x}_k)\mathbf{d}_k < 0$$

bis auf den Faktor σ erreicht oder gar übertrifft.

Aufgabe 1. Schreiben Sie Funktionen

```
function xs = gradientDescent(f, Df, x0, tol, maxIter, sigma)
function xs = quasiNewtonBFGS(f, Df, x0, tol, maxIter, sigma)
function xs = nonlinearcgPR(f, Df, x0, tol, maxIter, sigma)
```

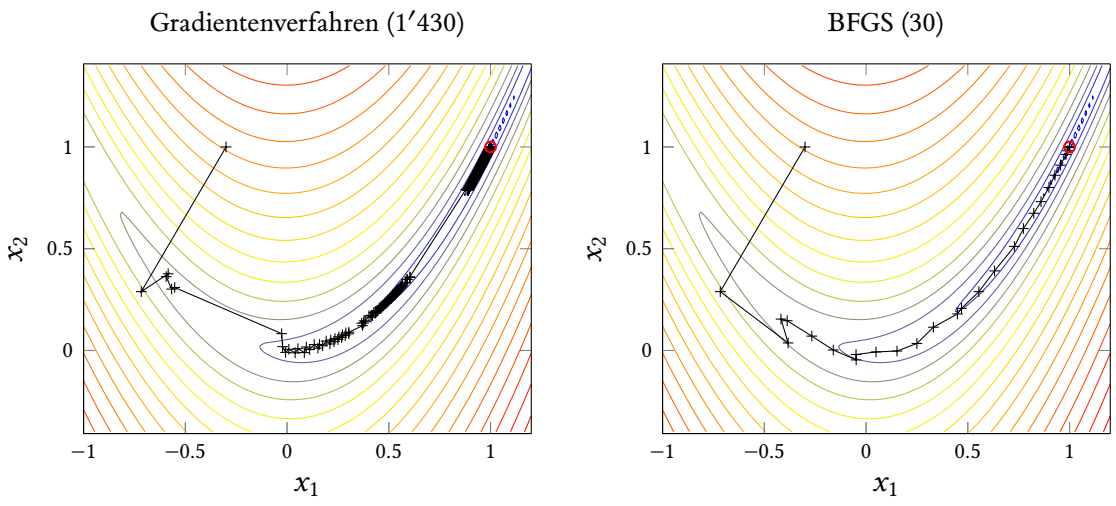
welche die Minimierungsalgorithmen aus dem Skript implementieren. Die Funktionen implementieren Sie gemäss den Algorithmen des Skripts, sprich `gradientDescent` ist analog zu Algorithmus 3.2, `quasiNewtonBFGS` gemäss Algorithmus 4.10 und `nonlinearcgPR` gemäss Algorithmus 4.13 zu implementieren.

Für die Liniensuche benutzen Sie jeweils die klassiche Armijo-Liniensuche. Die Funktionen sollen maximal `maxIter` Iterierte berechnen und jeweils abbrechen, wenn die 2-Norm des Gradienten kleiner als `tol` wird. Vermeiden Sie weiter die Funktion f und deren Ableitung am gleichen Punkt mehrmals auszuwerten, indem Sie stattdessen die aktuellen, notwendigen Auswertungen speichern. Die Funktions handles `f` und `Df` sollen den Auswertungspunkt als Spaltenvektor erhalten und den Funktionswert als Skalar respektive die Ableitung als Zeilenvektor ausgeben. Der Gradient in Ihrer Implementirung soll dann der Spaltenvektor sein, der durch transponieren aus der Ableitung hervorgeht.

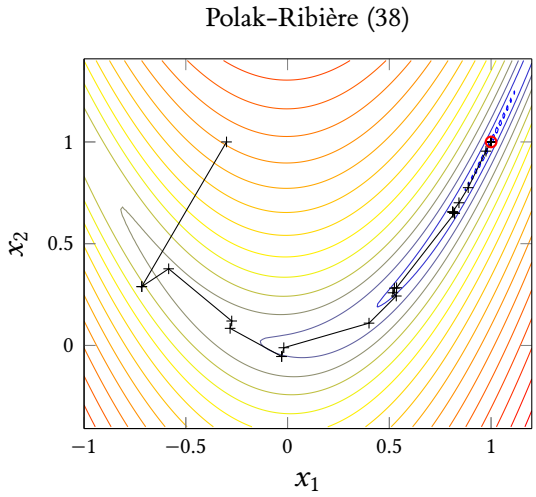
Aufgabe 2. Nutzen Sie die Funktionen aus der Aufgabe 1, um die klassische Rosenbrock-Funktion (1) mit $s = 1$, $a = 1$, $b = 100$ zu minimieren. Als Parameter benutzen Sie $\sigma = 0.3$, `tol` $=$ `1e-6` und `maxIter` $=$ `5000`. Als Startwert verwenden Sie den Punkt

$$\mathbf{x}_0 = \frac{1}{10}\begin{bmatrix} -3 \\ 10 \end{bmatrix}.$$

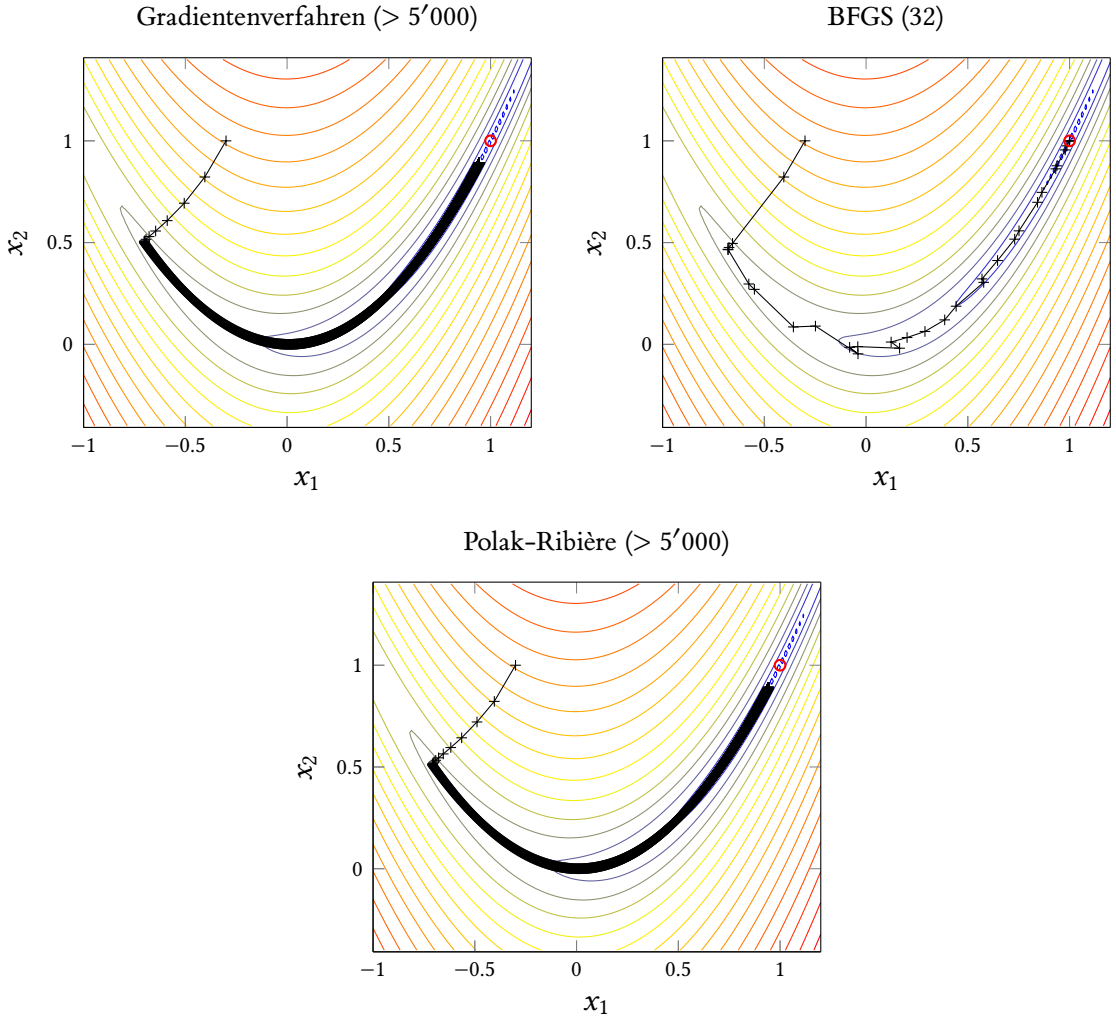
Stellen Sie in Subplots sowohl die Konturlinien der Rosenbrock-Funktion[1] als auch die Pfade der Optimierungsfunktionen gemäss Abbildung graphisch dar. Geben Sie auch die Anzahl der benötigten Iterationen an.



[1] Nutzen Sie hierfür den Matlab-Befehl `contour` um die Konturlinien von $\sqrt[3]{f}$ zu zeichnen.



Aufgabe 3. Wiederholen Sie Aufgabe 2, wobei Sie anstatt $s = 1$ neu die Skalierung der Rosenbrockfuntion auf $s = 2^{-10}$ setzen.



Modifizierte Armijo-Liniensuche

Die klassische Armijo-Liniensuche kann in der Praxis zu Problemen führen, da sie maximal die Schrittweite 1 erlaubt. Je nach Wahl der Abstiegsrichtung kann es aber sein, dass man

Schrittweiten signifikant grösser als 1 wählen sollte, damit man nicht zuviele kleine Schritte macht. Aufgabe 3 legt nahe, dass dies insbesondere bei dem Gradientenverfahren und den nichtlinearen CG-Verfahren die Konvergenz signifikant beeinträchtigen kann; aufgrund der (approximativen) Hess'schen Information sind dementgegen (Quasi-)Newtonverfahren diesbezüglich stabiler. Umgekehrt kann es aber auch sein, dass die Wahl der Abstiegsrichtung und dem σ bedingen, dass sämtliche Schrittweiten signifikant kleiner als 1 gewählt werden müssen. In diesem Fall wird die klassische Armijo-Liniensuche also für jede Iterierte diese kleinere Schrittweite ausgehend von 1 neu ermitteln, was viele Funktionsauswertungen von f bedingen kann.

Um solche Probleme zu vermeiden, kann man die Armijo-Liniensuche mit verschiedenen Heuristiken modifizieren. Wir werden uns auf die folgende Modifikation beschränken:

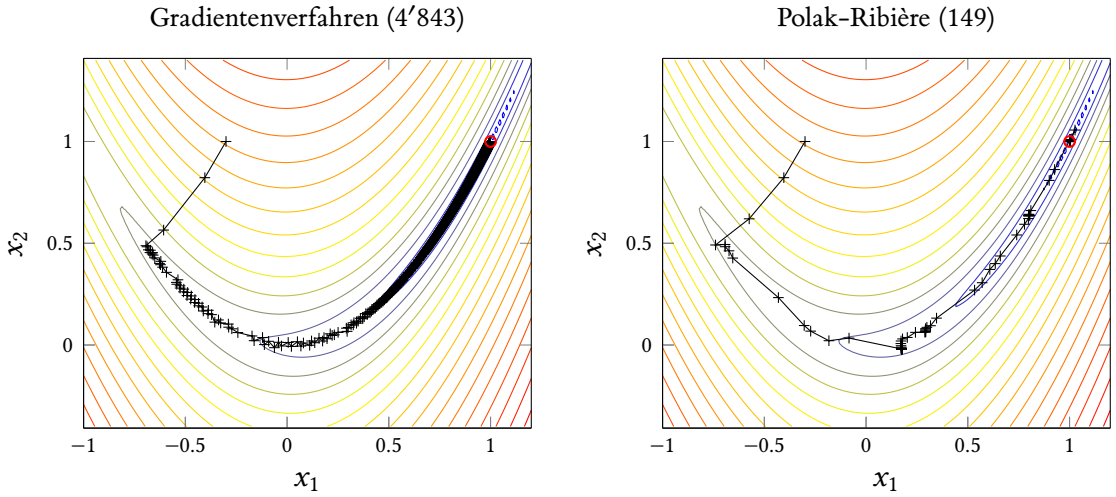
1. Die erste Schrittweite α_0 wird ausgehend von $\alpha_0 = 1$ ermittelt.
2. Alle folgenden Schrittweiten werden ausgehend von der verdoppelten, vorherigen Schrittweite besstimmt; sprich man startet die Liniensuche mit $\alpha_k = 2\alpha_{k-1}$ für $k \in \mathbb{N}^+$.

Aufgabe 4. Schreiben Sie Funktionen

```
function xs = gradientDescentMod(f, Df, x0, tol, maxIter, sigma)
function xs = nonlinearcgPRMod(f, Df, x0, tol, maxIter, sigma)
```

ausgehend von denen in Aufgabe 1, welche die klassischen Armijo-Liniensuche mit der modifizierten ersetzen.

Aufgabe 5. Wiederholen Sie Aufgabe 3 mit den modifizierten Funktionen von Aufgabe 4.



Globalisierungsstrategie

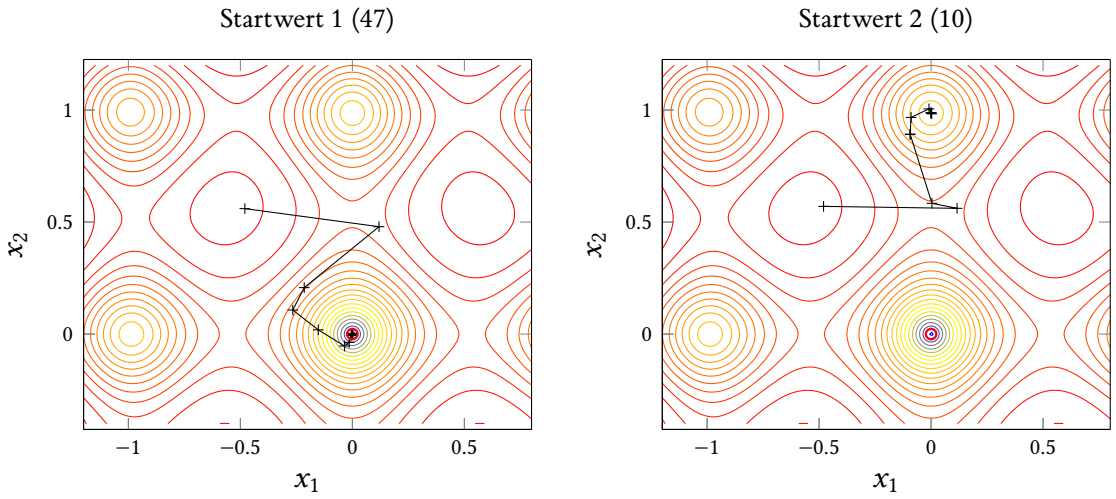
Ein Nachteil von Abstiegsverfahren ist, dass sie bei kritischen Stellen der Funktion gefangen werden können. Während man beim zufälligen Erreichen von einem lokalen Maximum oder Sattelpunkt, den Algorithmus mit einer kleinen zufälligen Störung neu starten kann und dadurch mit hoher Wahrscheinlichkeit einen weiteren Abstieg erreichen wird können, ist dies beim Erreichen eines lokalen Minimums nicht der Fall.

Aufgabe 6. Nutzen Sie die Funktion `quasiNewtonBFGS` aus der Aufgabe 1, um die klassische Ackley-Funktion (2) mit $s = 1$, $a = 20$, $b = 0.05$ und $c = 2\pi$ zu minimieren. Als Parameter

benutzen Sie $\sigma = 0.3$, `tol` $= 1e-6$ und `maxIter` $= 1000$. Als Startwert verwenden Sie die Punkte

$$\mathbf{x}_0 = \begin{bmatrix} -0.48 \\ 0.56 \end{bmatrix} \qquad \text{und} \qquad \mathbf{x}_0 = \begin{bmatrix} -0.48 \\ 0.57 \end{bmatrix}.$$

Stellen Sie in Subplots sowohl die Konturlinien der Ackley-Funktion[2] als auch die Pfade der zwei Startwerte gemäss Abbildung graphisch dar. Geben Sie auch die Anzahl der benötigten Iterationen an.



Wie in Aufgabe 6 klar ersichtlich, kann eine kleine Änderung in dem Startwert dazu führen, dass nicht das globale Minimum, sondern nur ein lokales gefunden wird. Ziel einer Globalisierungsstrategie ist es, ein Abstiegsverfahren so anzuwenden oder anzupassen, dass es bessere Chancen hat, das globale Minimum, oder zumindest bessere lokale Minima, zu finden. Im Allgemeinen basieren solche Strategien auf Heuristiken und machen insbesondere Gebrauch von Zufallszahlen, damit die Funktion hoffentlich für eine grössere Spanne an Argumenten untersucht wird.

Aufgabe 7. Laden Sie die Datei `prog03evolutionary.zip` herunter. Darin finden Sie die MATLAB Funktion `evolutionaryBFGS` und das MATLAB Skript `main7`, welche ausgehend von der Funktion `quasiNewtonBFGS` den Versuch einer Globalisierungsstrategie und deren Anwendung darstellt. Überlegen Sie sich, wie die Globalisierungsstrategie in der Funktion `evolutionaryBFGS` aussieht, und führen Sie das Skript `main7` mehrmals aus.

[2] Nutzen Sie hier den Matlab-Befehl `contour` um die Konturlinien von $\sqrt{f}$ zu zeichnen.