Übung 3: Downhill-Simplex-Verfahren

Computational Physics 1 - WS 2020/2021

07.12.2020

Programmieren Sie eine Funktion, die das Downhill-Simplex-Verfahren in einem 2-dimensionalen Gebiet ($n = 2$) realisiert um **ein** Minimum $(x_1, x_2)_{\min}$ einer beliebigen vorgegebenen Funktion ausgehend von einem Startpunkt $(x_1, x_2)_{\text{start}}$ zu finden. Der Benutzer soll die gewünschte Genauigkeit p sowie die maximale Schrittanzahl $N_{\max}$ angeben können, nach der das Programm abbricht. Dabei soll p eine obere Grenze für den Fehler sowohl für die Koordinaten des Minimums als auch für den Funktionswert sein.

Benutzen Sie den vorgegebenen Funktionskörper **simplex_skeleton.py** und die dort empfohlenen Werte für die Größe des Startsimplex und dessen Transformationsfaktoren. Testen Sie ihre Funktion in einem separaten Skript mit der Himmelblau-Funktion

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \tag{1}$$

für verschiedene Startkoordinaten. Finden Sie dabei alle Minima der Funktion:

$$\begin{aligned}(x_1, x_2)_{\min} \in \{&(3; 2),\\ &(-2,805118; 3,131312),\\ &(-3,779310; -3,283186),\\ &(3,584428; -1,848126)\},\end{aligned} \tag{2}$$

für alle Minima gilt $f(x_1, x_2)_{\min} = 0$. Die Himmelblau-Funktion wird in der Datei **himmelblau.py** zur Verfügung gestellt und kann an **simplex.py** übergeben werden.

Erläutern Sie den Zusammenhang zwischen gefundenem Minimum und gewählten Startkoordinaten. Untersuchen Sie außerdem für $(x_1, x_2)_{\text{start}} = (-1; -1)$ und $p = 0$ (maximale Genauigkeit) das Konvergenzverhalten indem Sie $f(x_1, x_2)_{\min}$ in Abhängigkeit von $N_{\max}$ graphisch darstellen (logarithmische Darstellung von $f(x_1, x_2)_{\min}$ könnte sinnvoll sein). Interpretieren Sie die Ergebnisse.

1 Hinweise

- `np.zeros` erzeugt eine Matrix aus Nullen
- `np.argsort` liefert die Indizes, die ein Array in aufsteigender Ordnung sortieren würden

- `np.mean` berechnet den Mittelwert
- `np.std` berechnet die Standardabweichung
- `np.max` berechnet die Maxima der Spalten einer Matrix
- `or` logisches ODER, z.B. `if (a>0) or (b <= c):`
- `and` logisches UND, z.B. `while (a>0) and (b <= c):`

2 Einreichung per E-mail

Die Lösungen sind als E-Mail-Anhang bis spätestens 21.12.2020, 04:00 Uhr morgens an teaching-nanooptics@uni-jena.de zu senden. Bitte Name, Matrikelnummer und Nummer der Übungsserie in die Betreffzeile der E-Mail schreiben. Lösungen ohne Angabe von Name und Matrikelnummer, mit Funktionsdefinitionen die von den Vorgaben abweichen oder zu spät eingereichte Lösungen können nicht berücksichtigt werden.

Die Funktion **simplex.py** soll nachdem Schema der Python-Datei **simplex_skeleton.py** aufgebaut sein und in einem externen Modul **eval.py** definiert werden.

Einzusenden ist **eine zip-Datei (nicht 7-zip!)** mit mindestens **simplex.py**, **eval.py** und eine 1-2 seitige Darstellung Ihrer Ergebnisse, welche die erzeugten Bilder und deren Beschreibung sowie Interpretation beinhaltet.