Programming C++

Project **Nullstellen im 2D-Gebiet**

Status: 13. Mai 2024, 11:03

Supervisor: Prof.Dr. G. Haase, gundolf.haase@uni-graz.at

Aufgabe: Es sind reelle **Nullstellen in 2D** einer stetigen Funktion $f(x,y) : \mathbf{R}^2 \mapsto \mathbf{R}$ im Gebiet $\Omega = [a,b] \times [c,d]$ zu finden.
Hierbei sollen die Nullstellen (x^*, y^*) mit einer vorgegebenen Genauigkeit h bestimmt werden, d.h., das Bestimmen des Teilrechtecks/Pixels $\Omega^h := [x^* - \frac{h}{2}, x^* + \frac{h}{2}] \times [y^* - \frac{h}{2}, y^* + \frac{h}{2}]$ reicht aus.

Im Gegensatz zum Finden der Nullstellen einer Funktion $s(x) : \mathbf{R} \mapsto \mathbf{R}$ in 1D sind die Nullstellen in 2D üblicherweise Kurven (traces) in der xy-Ebene. Daher wird es in 2D unendlich viele diskrete Nullstellen geben und wir können nur eine repräsentative Teilmenge ($\leftarrow h$) davon bestimmen.

i) Auf Seite 2 sind die Lösungsansätze A(h) und B(h,H) beschrieben. Der Parameter h stellt hierbei stets die zu erreichenden Genauigkeit bzgl. der Nullstellen dar während $H \gg h$ in Ansatz B die frei wählbare Kantenlänge der groben Gebietsaufteilung darstellt.
Implementieren Sie beide Ansätze und testen Sie diese für die gegebenen Funktionen. (6 Pkt.)
Die Anzahl der jeweils gefundenen Ω^h_{ij} kann für die beiden Ansätze unterschiedlich sein.

ii) Führen Sie für unterschiedliche h und H Laufzeitmessungen der beiden Ansätze aus.
Speichern Sie diese Laufzeiten (und zugehörige h, H) in einem geeigneten Fileformat und veranschaulichen Sie die entstehenden Graphen mit Matlab/Python/sage/excel/... (+2 Pkt.)
Welcher der beiden Ansätze ist schneller?

iii) (*) Nutzen Sie alle Cores Ihrer CPU für die Loops beider Ansätze. (+2 Pkt.)
Mögliche Programmieransätze sind *execution policies* in C++ oder OpenMP-Parallelisierung (oder sogar Tasks oder auch CUDA).
Diese Zusatzaufgabe ist am besten unter Linux/MacOS mit aktuellen Compilern zu bearbeiten.

Testfunktionen:

- $f_1(x,y) = x + y$ in $\Omega = [-1,1] \times [-1,1]$
- $f_2(x,y) = x^2 + y^2 - 1$ in $\Omega = [-2,2] \times [-2,2]$
- $f_3(x,y) = \frac{(x+y)\left(x+y+\frac{11}{2}\right)(x+y-14)\left(x+y-\frac{19}{20}\right)\left(x+y-\frac{21}{20}\right)\left(x+y+\frac{1}{100}\right)\left(x+y+\frac{8951599254893357}{4503599627370496}\right)}{10000000000}$
in $\Omega = [-15,6] \times [-15,6]$
- $f_4(x,y) = \sin\left(\frac{1}{x^2+y^2+\frac{1}{10}}\right)$ in $\Omega = [0,0.5] \times [0,0.5]$.

Wie bestimme ich Nullstellen im Rechteck $[a, b] \times [c, d]$?

Es gibt keinen allgemeingültigen Algorithmus zur globalen Bestimmung der Nullstellen einer allgemeinen Funktion $f(x, y)$ im Gebiet Ω.

Lösungsidee A (brute force):

(a) Unterteilung von Ω in Teilrechtecke $\Omega_{ij}^h := [x_{j-1}, x_j] \times [y_{i-1}, y_i] \subset \Omega$ mit einer Kantenlänge $<= h$.

(b) Alle Ω_{ij} mit mindestens einem Vorzeichenwechsel von $f(x, y)$ in den 4 Eckpunkten zählen als Nullstellenrechteck. Nehmen Sie $f(x_s, y_s) * f(x_t, y_t) <= 0$ mit $t, s \in$ {Eckpunkte} als Kriterium für den Vorzeichenwechsel.

(c) Speicherung der Mittelpunkte obiger Nullstellenrechtecke Ω_{ij} als gefundene Nullstellen.

Lösungsidee B (adaptiv strategy):

(a) Wahl eines „hinreichend kleinen“ $H \gg h$ und Unterteilung von Ω in Teilrechtecke Ω_{kl}^H mit dieser Kantenlänge H.

(b) Alle Ω_{kl}^H mit mindestens einem Vorzeichenwechsel von $f(x, y)$ in den 4 Eckpunkten zählen als grobes Nullstellenrechteck.

(c) Unterteilung aller Nullstellenrechtecke in vier Teilrechtecke.

(d) Rekursive Fortführung der Unterteilung in den Teilrechtecken mit Vorzeichenwechsel.

(e) Ausführung von (b), (c) und (d) bis die Kantenlänge von $\Omega_{ij} <= h$ ist.

(f) Speicherung der Mittelpunkte obiger Nullstellenrechtecke Ω_{ij}^h als gefundene Nullstellen.

!! Es können 0 bis 4 Teilrechtecke als potentielle Nullstellenrechtecke in der Rekursion auftreten.

Auch dieser Algorithmus garantiert nicht, daß alle Nullstellen gefunden werden, er ist aber eine brauchbare Heuristik.
Bei mehr Information über die Funktion $f(x)$ und Ausnutzung dieser vorhandenen Strukturen kann man bessere Algorithmen zur Nullstellenbestimmung ableiten.