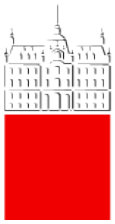
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Numerična minimizacija

3. naloga pri Modelski Analizi 1

Avtor: Marko Urbanč (28232019)
Predavatelj: prof. dr. Simon Širca
Asistent: doc. dr. Miha Mihovilovič

24.2.2023

Kazalo

1 Uvod

Numerično minimizacijo poznamo tudi pod širšim imenom matematična optimizacija. V zelo preprostih pojmih gre za izbito najbolj primernega elementa iz neke množice, glede na podane kriterije. Običajno je ta množica neka funkcija, ki jo želimo minimizirati. Kriterij pa je podan z neko funkcijo, ki nam pove kako dober je neki element. Če se to sliši zelo podobno kot uvod pri prejšnji nalogi, kjer smo si pogledali Linearno programiranje, je to zato, ker je to v bistvu ista stvar. Razlika je le v tem, da je pri linearnem programiranju funkcija, ki jo minimiziramo linearna, pri numerični minimizaciji pa je ta funkcija lahko poljubna.

Z največjim veseljem bi napisal še kakšen bolj matematičen uvod, ampak to bi pomenilo, da bi se moral spustiti v podrobnosti delovanja posameznih optimizacijskih algoritmov, kar pa je precej obsežna tema, še sploh če bi se želel dotakniti vseh, ki sem jih poskusil, ker jih je res veliko.

Veliko problemov se prevede na optimizacijske probleme, zato je to zelo pomembna tema. Če ne drugega, je dandanes strašno priljubljeno strojno učenje, ki je v bistvu nič drugega kot optimizacija neke funkcije, ki nam pove kako dobro se nek model prilega podatkom. Tu se pojavi poanta, ki sem jo želel (a neuspešno) povedati v prejšnji nalogi. Optimizacija funkcije z npr. 40 parametri je zelo malo. V praksi smo zmožni optimizirati funkcije z več milijoni parametrov in tu se vmeša meni priljubljen High Performance Computing in to da sem želel pri prejšnji nalogi 400k dimenzij. Tukaj sicer ne bomo počeli tega, ne ker ne bi želel, ampak ker žal nisem utegnil.

2 Naloga

Naloga je sestavljena iz dveh delov. Poglejmo:

2.1 Thomsonov problem

Thomsonov problem je problem, kjer želimo najti najboljšo porazdelitev n točk po sferi, tako da bo potencial (lahko si mislimo kot električni potencial) čim manjši. To je v bistvu problem, ki ga rešujemo pri modeliranju atomov. Naloga želi, da za različne metode optimizacije preštudiramo pojav in natančnost rešitev.

2.2 Vožnja skozi semafor

Vrača se primer, ki smo ga imeli za prvo nalogo pri Modelski Analizi 1 z to razliko, da bomo tokrat namesto variacijskega problema reševali optimizacijski problem. Lagrangian je potrebno zapisati v primerni obliki, nato pa lahko uporabimo neko metodo optimizacije, da dobimo rešitev.

3 Opis reševanja

Reševanja sem se, kot je navada, lotil v Pythonu. Za optimizacijo sem uporabil knjižnico `scipy`, ki vsebuje veliko različnih metod za optimizacijo. Poskušal sem tudi z komercialnim solverjem `gurobipy` do katerega sem dobil dostop v kontekstu prejšnje naloge, ampak nisem dosegel željenih rezultatov. Ideja je bila, da bi potem reševanje paraleliziral. Poleg tega sem pa uporabil praktično stalen nabor knjižnic, ki torej `numpy`, `matplotlib`, `pandas` etc.

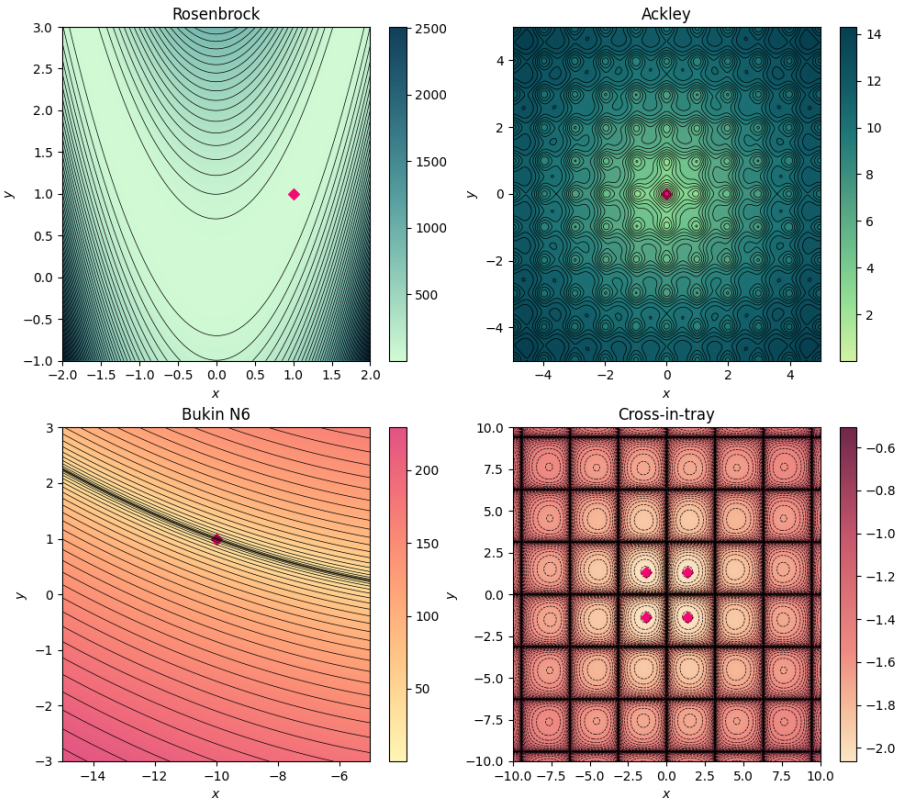
3.1 Suite of benchmarks

Za primerjavo uspešnosti različnih metod globalne optimizacije sem definiral suito funkcij, ki so posebej patološke oz. primerne za testiranje kvalitete optimizacijskih metod. Pogledal sem si:

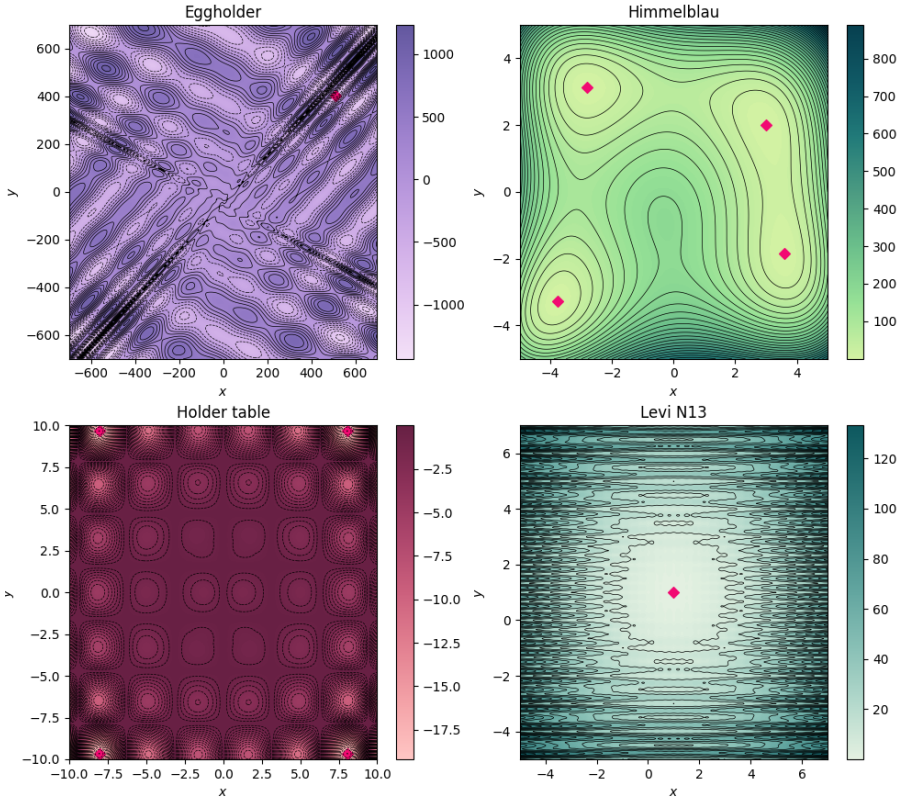
- Basin Hopping
- Differential Evolution
- Dual Annealing
- Simplicial Homology Global Optimization (SHGO)

Predstavljeno v enem stavku, Basin Hopping deluje tako, da delamo lokalno optimizacijo, nato pa naključno spreminjamo parametre in ponovimo postopek. Differential Evolution je evolucijski algoritem, ki deluje tako, da naključno generira populacijo, nato pa jo spreminja z nekimi pravili. Dual Annealing je algoritem, ki deluje na principu simuliranega ohlajanja. SHGO pa je algoritem, ki deluje tako, da razdeli prostor parametrov na simplicialne komplekse in nato v vsakem izvede lokalno optimizacijo. Podrobneje je SHGO opisan v članku [1].

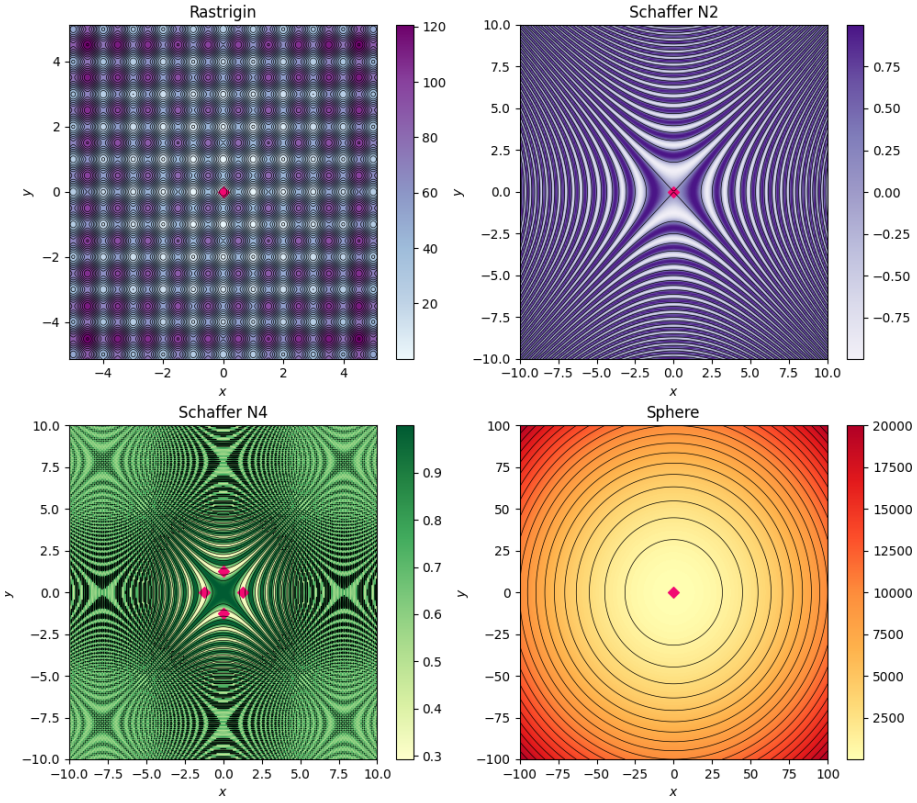
Funkcije, ki sem jih definiral sem narisal na slikah (1, 2, 3). Performance metod pri njihovi optimizaciji je prikazan v rezultatih.



Slika 1: Prve 4 funkcije iz suite of benchmarks.



Slika 2: Naslednje 4 funkcije iz suite of benchmarks.



Slika 3: Zadnje 4 funkcije iz suite of benchmarks.

3.2 Thomsonov problem

Najprej sem napisal funkcijo, ki je za dobljene kote θ in ϕ na enotski sferi izračunala potencial. Pri temu sem upošteval, da imamo pravzaprav $m = n - 1$ točk, ker je ena točka fiksna. Njo sem postavil na severni pol. Preostale točke sem porazdelil enakomerno po ekvatorju. Pogledal sem si natančnost različnih metod, ki jih nudi `scipy.optimize.minimize()` pri $m = 1$, kajti to je edini primer za katerega sem se počutil samozavestno, da poznam analitičen odgovor. To pomeni, da sta dva naboja na krogli, kar pomeni, da gre ne fiksirani naboj v drug pol. Nato sem si pogledal še nekaj primerov in naredil animacijo za $m = 9$.

3.3 Vožnja skozi semafor

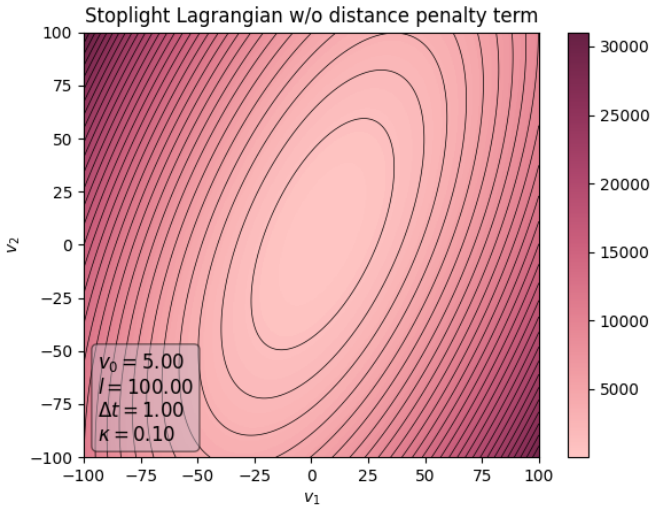
Najprej sem napisal funkcijo $f(v_0, v_1, v_2, ..., v_n)$ po trapeznem pravilu, definirano takole:

$$f = \frac{1}{2}\left(\frac{v_1 - v_0}{\Delta t}\right)^2 + \left(\frac{v_2 - v_1}{\Delta t}\right)^2 + ... + \left(\frac{v_{n-1} - v_{n-2}}{\Delta t}\right)^2 + \frac{1}{2}\left(\frac{v_n - v_{n-1}}{\Delta t}\right)^2, \tag{1}$$

kjer je v_0 začetna hitrost, v_n končna hitrost, v_i hitrost v i-tem trenutku, Δt pa časovni korak. Tej funkciji sem še prištel vez s katero sem zahteval, da je ploščina pod grafo enaka l, kjer je l razdalja do semaforja. Vez $g(v_0,\ v_1,\ v_2,\ ...,\ v_n)$ je definirana takole:

$$g = 1 + \exp\left[\kappa\left(\frac{1}{2}v_0 + v_1 + ... + v_{n-1} + \frac{1}{2}v_n - \frac{l}{\Delta t}\right)\right]. \tag{2}$$

Skupno funkcijo sem minimiziral z metodami za lokalno optimizacijo, ki jih ponuja `scipy.optimize.minimize()`. Ker sem imel od prvega dela naloge največ uspega z `Powell` metodo, sem se odločil, da jo uporabim tudi tukaj. Začetne pogoje sem nastavil tako, da so vse hitrosti enake in tako sem dobil neke rezultate. Bi si pa želel, da bi posvetil lahko več časa temu delu naloge, ker se bi mi zdelo zanimivo primerjati bolj podrobno dobljeno z rezultati prve naloge pri Modelski Analizi 1. Žal je oddajni rok že tu. Lahko si pa pogledamo s kakšno funkcijo imamo opravka. Na sliki (4) je prikazana funkcija, ki jo minimiziramo.



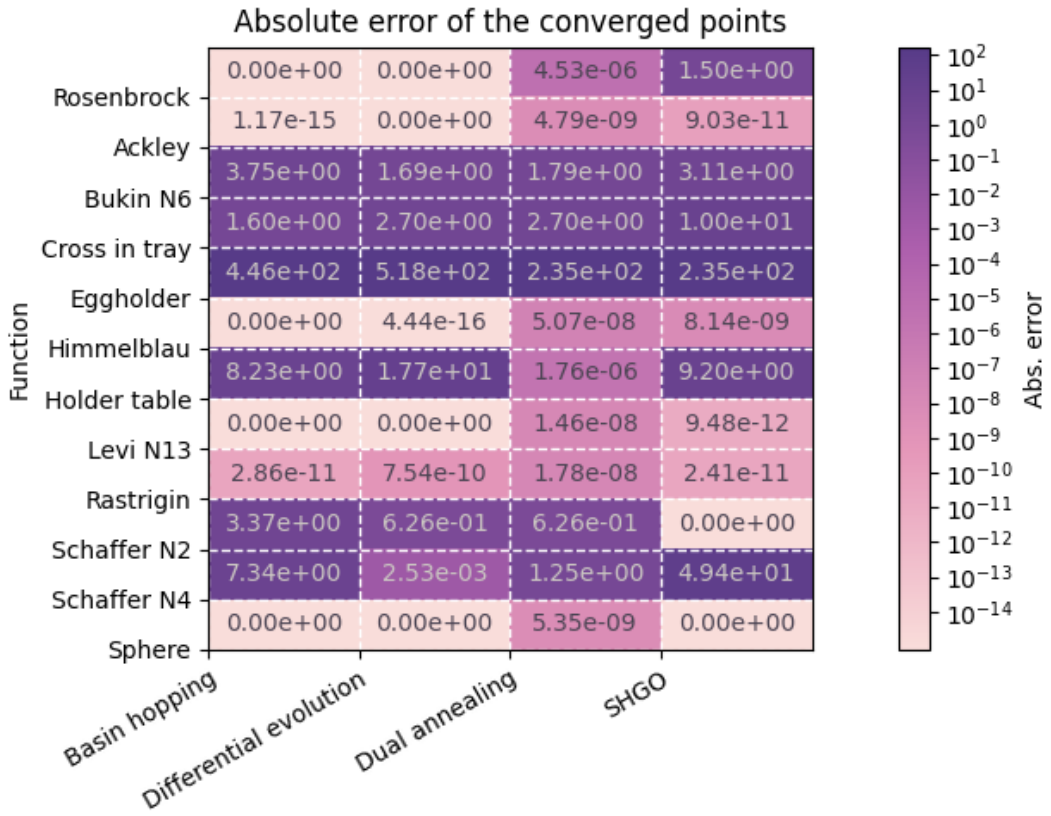
Slika 4: Funkcija, ki jo minimiziramo.

4 Rezultati

4.1 Suite of benchmarks

Kot sem že omenil, sem za primerjavo uspešnosti različnih metod globalne optimizacije definiral suito funkcij, ki so posebej patološke oz. primerne za testiranje

kvalitete optimizacijskih metod. Nato sem si pogledal, za koliko odstopajo rezultati od pravih vrednosti. Rezultat je prikazan na sliki (5).



Slika 5: Rezultati optimizacije funkcij iz suite of benchmarks.

Vidimo, da je od vseh funkcij najbolj zoprna Eggholderjeva funkcija. Če pogledamo še na sliki (2) vidimo, da tudi res zgleda tako. Meni se sploh ne zdi očiten tisti minimum. Pa še tako nazobčeno je vse skupaj, kar pomeni, da se je zelo težko znebiti lokalnih minimumov. Najlažje je pa optimizirati Sferično simetrično funkcijo, kar je tudi zelo smiselno. V splošnem zgleda najbolj uspešna metoda kar Basin Hopping

5 Komentarji in izboljšave

Literatura

[1] Stefan C. Endres, Carl Sandrock, and Walter W. Focke. A simplicial homology algorithm for lipschitz optimisation. *Journal of Global Optimization*, 72(2):181–217, Oct 2018.