2. DOMAĆA ZADAĆA – AK. GOD. 2016/17

Domaća zadaća

U okviru ove domaće zadaće potrebno je implementirati sljedeće metode optimiranja:

Postupak zlatnog reza

Algoritam je dan na web stranici predmeta (http://www.fer.unizg.hr/_download/repository/zlatni_rez.txt) i na predavanjima. U okviru ovog postupka potrebno je također implementirati metodu pronalaženja unimodalnog intervala (**napomena**: *kao što je pokazano na predavanjima, po* http://www.fer.unizg.hr/_download/repository/unimodalni.txt). Postupak zlatnog reza mora se moći pokrenuti ili s početnom točkom, pomoću koje se onda određuje unimodalni interval ili s predefiniranim intervalom, u kojem slučaju se preskače postupak traženja unimodalnog intervala i koristi se zadani interval. U svakom koraku postupka potrebno je ispisati sve četiri točke i vrijednosti funkcije cilja u njima. Potrebno je omogućiti da se bez prevođenja programa mogu definirati preciznost *e* te početna točka ili početni interval pretraživanja (npr. učitavanjem iz datoteke). Pretpostavljene vrijednosti: 10^{-6} za *e*. (**napomena**: *za iznos konstante k uzmite što precizniju vrijednost, računanjem kao* $0.5\cdot\left(\sqrt{5}-1\right)$).

Pretraživanje po koordinatnim osima

Algoritam je opisan u skripti (str. 4-33, 4-34) i na predavanjima. U sklopu ove metode iskoristite metodu zlatnog reza za minimizaciju u jednoj dimenziji. Potrebno je omogućiti da se bez prevođenja programa mogu definirati vektor granične preciznosti *e* te početna točka pretraživanja (npr. učitavanjem iz datoteke). Pretpostavljene vrijednosti: 10^{-6} za sve elemente *e*.

Simpleks postupak po Nelderu i Meadu

Algoritam je dan na web stranici predmeta (http://www.fer.unizg.hr/_download/repository/simplex.html). Početni simpleks generirajte na način da se za jednu konkretnu novu točku pomaknete od početne točke za određeni pomak u jednoj dimenziji (**primjer**: *ako je početna točka (0,0) i pomak 1, tada ćete generirati točke (1,0) i (0,1)*). U svakom koraku postupka potrebno je ispisati centroid točaka i vrijednost funkcije cilja u njoj. Potrebno je omogućiti da se bez prevođenja mogu definirati početna točka pretraživanja, preciznost *e*, pomak koji se koristi za generiranje simpleksa, parametri *alfa*, *beta*, *gamma* te vrijednost za koju se sve točke pomiču prema najboljoj točki simpleksa (*sigma*). Pretpostavljene vrijednosti za parametre: 10^{-6} za *e*, 1 za pomak koji se koristi kod generiranja simpleksa, *alfa* = 1, *beta* = 0.5, *gamma* = 2 i *sigma* = 0.5.

Hooke-Jeeves postupak

Algoritam je dan na web stranici predmeta (http://www.fer.hr/_download/repository/hj.html) i opisan na predavanjima (**napomena**: *ne implementirati algoritam u skripti*, koji ima nešto drugačije ponašanje!). U svakom koraku postupka potrebno je ispisati baznu točku, početnu točku pretraživanja i točku dobivenu pretraživanjem, kao i vrijednosti funkcije cilja u tim točkama. Potrebno je omogućiti da se bez prevođenja programa mogu definirati: vektor pomaka *dx*, vektor granične preciznosti *e* te početna točka pretraživanja (npr. učitavanjem iz datoteke). Pretpostavljene vrijednosti: 0.5 za sve elemente *dx* te 10^{-6} za sve elemente *e*.

Svi algoritmi (osim postupka zlatnog reza) moraju podržavati proizvoljno veliku dimenzionalnost rješenja. *Funkciju cilja* potrebno je implementirati tako da vodi evidenciju o broju pozivanja, jer će algoritme biti potrebno usporediti na temelju broja evaluacija (preporučuje se ostvariti kao apstraktni razred).

Funkcije cilja

- $f_1(\mathbf{x}) = 100 \cdot (x_2 - x_1^2)^2 + (1 - x_1)^2$ (*Rosenbrockova 'banana' funkcija*)

  Početna točka: $\mathbf{x}_0 = (-1.9, 2)$, minimum: $\mathbf{x}_{\min} = (1,1)$, $f_{\min} = 0$
- $f_2(\mathbf{x}) = (x_1 - 4)^2 + 4 \cdot (x_2 - 2)^2$

  Početna točka: $\mathbf{x}_0 = (0.1, 0.3)$, minimum: $\mathbf{x}_{\min} = (4,2)$, $f_{\min} = 0$
- $f_3(\mathbf{x}) = \sum_i (x_i - i)^2$

  Početna točka: $\mathbf{x}_0 = \mathbf{0}$ (nul vektor), minimum: $\mathbf{x}_{\min} = (1, 2, 3, ..., n)$, $f_{\min} = 0$
- $f_4(\mathbf{x}) = \left|(x_1 - x_2) \cdot (x_1 + x_2)\right| + \sqrt{x_1^2 + x_2^2}$ (*Jakobovićeva funkcija* :)

  Početna točka: $\mathbf{x}_0 = (5.1, 1.1)$, minimum: $\mathbf{x}_{\min} = (0,0)$, $f_{\min} = 0$
- $f_6 = 0.5 + \dfrac{\sin^2 \sqrt{\sum x_i^2} - 0.5}{\left(1 + 0.001 \cdot \sum x_i^2\right)^2}$ (*Schaffer's function f6*)

  minimum: $\mathbf{x}_{\min} = \mathbf{0}$ (nul vektor), $f_{\min} = 0$

Laboratorijska vježba

1. Definirajte jednodimenzijsku funkciju br. 3, koja će imati minimum u točki 3. Kao početnu točku pretraživanja postavite točku 10. Primijenite sva tri postupka na rješavanje ove funkcije te ispišite pronađeni minimum i broj evaluacija funkcije za svaki pojedini postupak. Probajte sve više udaljavati početnu točku od minimuma i probajte ponovo pokrenuti navedene postupke. Što možete zaključiti?
2. Primijenite simpleks po Nelderu i Meadu, Hooke-Jeeves postupak te pretraživanje po koordinatnim osima na funkcije 1-4 uz zadane parametre i početne točke (broj varijabli funkcije 3 najmanje 5). Za svaki postupak i svaku funkciju odredite minimum koji su postupci pronašli i potrebni broj evaluacija funkcije cilja koji je potreban do konvergencije (prikažite tablično). Što možete zaključiti iz rezultata?
3. Primijenite postupak Hooke-Jeeves i simpleks po Nelderu i Meadu na funkciju 4 uz početnu točku (5, 5). Objasnite rezultate!
4. Primijenite simpleks po Nelderu i Meadu na funkciju 1. Kao početnu točku postavite točku (0.5,0.5). Provedite postupak s nekoliko različitih koraka za generiranje početnog simpleksa (primjerice iz intervala od 1 do 20) i zabilježite potreban broj evaluacija funkcije cilja i pronađene točke minimuma. Potom probajte kao početnu točku postaviti točku (20,20) i ponovo provesti eksperiment. Što možete zaključiti?
5. Primijenite jedan postupak optimizacije na funkciju 6 u dvije dimenzije, tako da postupak pokrećete više puta iz slučajno odabrane početne točke u intervalu [-50,50]. Možete li odrediti vjerojatnost pronalaženja *globalnog* optimuma na ovaj način? (smatramo da je algoritam locirao globalni minimum ako je nađena vrijednost funkcije cilja manja od 10^{-4})¸

Napomena

Kako bi brojevi poziva funkcije za svaki od postupaka bili što relevantniji, probajte, gdje god je to moguće, izračunatu vrijednost funkcije za neku točku pohraniti i iskoristiti ju ponovo, umjesto da ju ponovo računate u idućem koraku.

Prije dolaska na laboratorijsku vježbu pripremite svaki od zadataka u obliku funkcije koju možete pokrenuti, tako da se svaki od zadataka može demonstrirati bez prevelikih promjena u kodu. Alternativno, možete koristiti konfiguracijske datoteke u kojima specificirate sve bitne parametre, tako da se vježba može pokrenuti za različite postupke bez potrebe za ponovnih prevođenjem.

Demonstracija funkcionalnosti u MATLAB-u

Ovaj dio vježbe izvodi se na predavanjima.

Potrebno je odabrati dvije optimizacijske funkcije, definirati ih kao funkcije u MATLABU i pronaći njihov minimum (bez ograničenja) uporabom ugrađenih MATLABovih funkcija. Neke od funkcija za optimiranje su sljedeće:

- `fminbnd`: minimum funkcije jedne varijable; koristi algoritam zlatnog reza i kvadratne interpolacije
- `fminsearch`: minimum funkcije više varijabli; koristi simplex postupak po Nelderu i Meadu
- `fminunc`: minimum funkcije više varijabli; ova funkcija može, ovisno o proslijeđenim parametrima, uporabiti nekoliko optimizacijskih algoritama, između ostaloga i postupak po Fletcheru i Powellu, poopćeni Newtonov postupak, metodu najbržeg spusta itd.
- `fmincon`: minimum funkcije više varijabli uz ograničenja

Funkcije se u MATLAB-u mogu definirati u posebnim .m datotekama, kao u sljedećem primjeru:

```
function f = apr_primjer(x)  % mora biti u prvom retku!
% Bezvezna funkcija dvije varijable
f = x(1) + x(2);
```

Datoteku je potrebno nazvati istim imenom kao i funkcija (apr_primjer.m). Iz MATLAB-a se funkciji tada može pristupiti s `apr_primjer([1,1])`.

Odabrana poglavlja iz MATLAB Helpa:

- DEMOS: Optimization: Minimization of the "banana function"
- MATLAB: Mathematics: Function Functions: Minimizing Functions and Finding Zeros
- Optimization Toolbox: Standard Algorithms: Unconstrained Optimization