2. domaća zadaća – numeričke optimizacije

Za rješavanje ove domaće zadaće dozvoljeno je koristiti biblioteku za rad s matricama: JAMA (http://math.nist.gov/javanumerics/jama/) ili biblioteku Apache Commons Math (http://commons.apache.org/proper/commons-math/).

Modelirajte u Javi sučeljem `IFunction` proizvoljnu skalarnu funkciju nad *n* dimenzijskim vektorom realnih brojeva. Sučelje bi trebalo definirati:

- metodu koja vraća broj varijabli nad kojima je definirana,
- metodu koja vraća vrijednost funkcije u predanoj točki,
- metodu koja vraća vrijednost gradijenta funkcije u predanoj točki.

Modelirajte u Javi sučeljem `IHFunction` funkciju koja je `IFunction` i dodatno nudi još i sljedeću metodu:

- metodu koja vraća Hesseovu matricu u predanoj točki.

Napravite općenitu implementaciju algoritma gradijentnog spusta kao statičku metodu u razredu `NumOptAlgorithms`. Metoda treba primati referencu na objekt koji implementira sučelje `IFunction` te maksimalni broj iteracija koji je dozvoljeno napraviti prije no što algoritam vrati rješenje.

Napravite općenitu implementaciju algoritma koji koristi Newtonovu metodu kao statičku metodu u razredu `NumOptAlgorithms`. Metoda treba primati referencu na objekt koji implementira sučelje `IHFunction` te maksimalni broj iteracija koji je dozvoljeno napraviti prije no što algoritam vrati rješenje.

Zadatke 1a, 1b, 1c i 1d napišite kao program `hr.fer.zemris.optjava.dz2.Jednostavno` koji prima 2 ili 4 argumenta. Prvi argument je uvijek slovo "1a", "1b", "2a" ili "2b", ovisno o tome što treba napraviti. Drugi argument je maksimalni broj iteracija. Treći i četvrti argument, ako su zadani, čine početnu točku iz koje treba pokrenuti postupak pretrage. Ako nisu, točka se generira slučajno. Prilikom testiranja rada algoritma slučajno generirajte početnu točku. Prilikom odgovaranja na postavljena pitanja u sva četiri slučaja krenite od iste točke (npr. [-5, -5]).

Zadatak 1a.
Napišite implementaciju funkcije $f_1(x_1, x_2) = x_1^2 + (x_2 - 1)^2$. Inicijalno rješenje generirajte iz prostora [-5,5]. Napisanim algoritmom gradijentnog spusta pronađite točku u kojoj ta funkcija poprima minimum. Ispišite na zaslon rješenje nakon svakog koraka algoritma. Nacrtajte tu trajektoriju u x1-x2 ravnini (GIF, PNG ili bilo koji sličan format).

Zadatak 1b.
Rješite isto napisanom Newtonovom metodom i nacrtajte trajektoriju. Koja je razlika u odnosu na a) dio?

Zadatak 2a.
Napišite implementaciju funkcije $f_2(x_1, x_2) = (x_1 - 1)^2 + 10 \cdot (x_2 - 2)^2$. Inicijalno rješenje generirajte iz prostora [-5,5]. Napisanim algoritmom gradijentnog spusta pronađite točku u kojoj ta funkcija poprima minimum. Ispišite na zaslon rješenje nakon svakog koraka algoritma. Nacrtajte tu trajektoriju u x1-x2 ravnini (GIF, PNG ili bilo koji sličan format). Usporedite s trajektorijom u zadatku 1a. Je li potreban isti broj iteracija? Zašto?

Zadatak 2b.
Rješite isto napisanom Newtonovom metodom i nacrtajte trajektoriju. Koja je razlika u odnosu na a) dio te u odnosu na rješenja 1. zadatka?

Zadatak 3.
U datoteci `zad-sustav.txt` dani su podatci o linearnom sustavu. Riješite ga uporabom razvijenih metoda (formulirajte postupak rješavanja kao optimizacijski problem i potom ga riješite). Neka je glavni program `hr.fer.zemris.optjava.dz2.Sustav` koji prima tri argumenta. Prvi je "grad" ili "newton" (ovisno o tome koji postupak treba upotrijebiti), drugi je maksimalni broj iteracija a treći putanja do datoteke sa zadatkom. Program treba ispisati pronađeno rješenje te iznos pogreške u trenutku kada je prekinuo s radom.

Zadatak 4.
Poznato je da je prijenosna karakteristika nekog sustava dana izrazom:
$y(x_1, x_2, x_3, x_4, x_5) = a \cdot x_1 + b \cdot x_1^3 x_2 + c\, e^{d \cdot x_3}(1 + \cos(e \cdot x_4)) + f \cdot x_4 x_5^2$.
Rad sustava snimljen je u više točaka i podatci su pohranjeni u datoteku `zad-prijenosna.txt`. Napišite program koji će temeljem tih podataka utvrditi koeficijente *a*, *b*, *c*, *d* i *e*. Neka je glavni program `hr.fer.zemris.optjava.dz2.Prijenosna` koji prima tri argumenta. Prvi je "grad" ili "newton" (ovisno o tome koji postupak treba upotrijebiti), drugi je maksimalni broj iteracija a treći putanja do datoteke s parovima (ulaz, izlaz) temeljem kojih se traže koeficijenti. Program treba ispisati pronađeno rješenje te iznos pogreške u trenutku kada je prekinuo s radom.

Napomene:

Svi postupci trebaju koristiti algoritam bisekcije za pronalazak optimalnog lambda koji u svakom koraku govori koliko treba modificirati trenutno rješenje.

Rok za predaju Eclipse projekta je sljedeći četvrtak do termina predavanja (14:00). Generirane slike stavite u vršni direktorij projekta. Slike možete crtati sami ili za njihovo generiranje možete koristiti odgovarajući program.