SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Interaktivna 3D vizualizacija gradijentne optimizacije

Petar Afrić
Voditelj: *doc. dr. sc. Siniša Šegvić*

Zagreb, siječanj 2018.

SADRŽAJ

1. Uvod **1**

2. Processing **2**

3. Ostvareni rad **4**

3.1. Koordinatni sustav 5

3.2. Funkcije 6

3.3. Optimizacijski postupci 11

4. Slični alati **13**

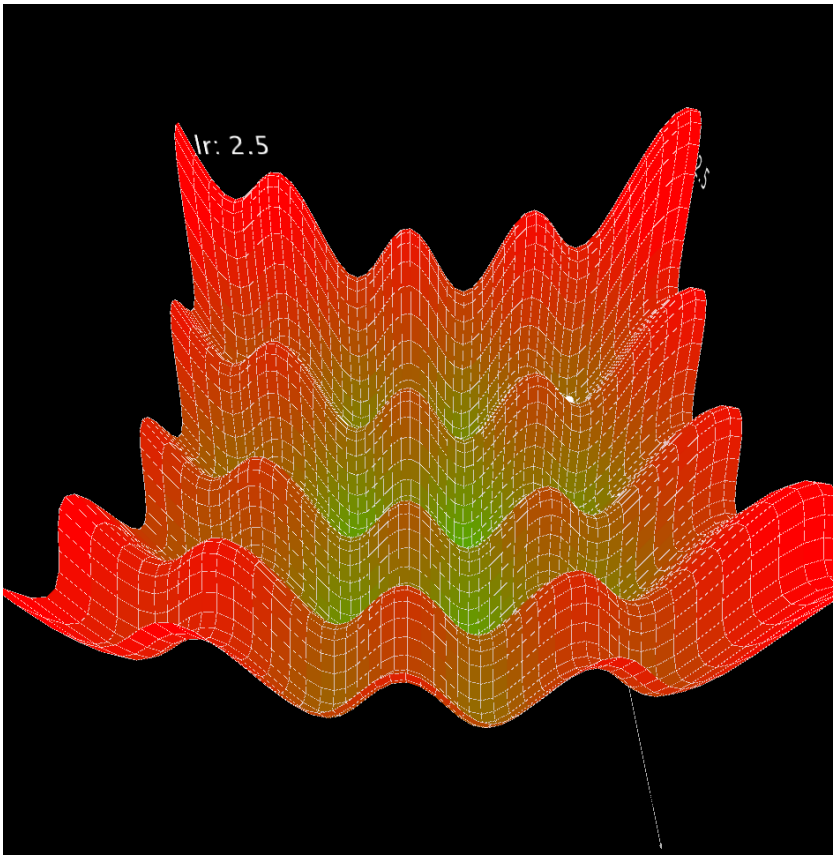
5. Zaključak **14**

6. Literatura **15**

1. Uvod

U sklopu ovog seminarskog rada razvijen je jednostavan alat za interaktivnu 3D vizualizaciju gradijentne optimizacije. Alat je razvijen u programskom jeziku Processing (Java) koristeći istoimeno razvojno okruženje.

Osnovna motivacija za razvoj alata bila je bolje razumijevanje gradijentne optimizacije odnosno, preciznije govoreći, bolje razumjevanje razlika između različitih algoritama gradijentne optimizacije. Kroz vizualizaciju rada različitih algoritama stječe se puno bolji osjećaj za njihove razlike te dodatno bolji osjećaj za utjecaj stope učenja na ponašanje algoritma.



Slika 1.1: Primjer ostvarene vizualizacije

2. Processing

Processing (1) je open-source programski jezik i integrirano razvojno okruženje fokusirano na digitalnu vizualizaciju i umjetnost razvijeno s ciljem učenja opće populacije o osnovma programiranja. Jezik je zasnovan na programskom jeziku Java, ali koristi ograničenu i pojednostavljenu sitaksu. Obzirom na svrhu za koju je izgrađen, jezik nije pogodan za razvoj velikih sustava, ali je izuzetno praktičan za razvoj manjih vizualizacijskih alata.

Svaki projekt u processingu naziva se *sketch* koji se razvija u *sketchbook-u* odnosno Processing IDE-u. Svaki sketch zapravo je podklasa PApplet klase koja implementira većinu značajki Processing framework-a. Sve ostale klase definirane u projektu u prevođenju se tretiraju kao ugnježđene klase početne sketch klase.

Processing programi imaju dvije glavne funkcije:

1. setup - inicijalizira program
2. draw - crta sadržaj programa

U nastavku je prikazan kod jednostavnog programa koji iscrtava jednu liniju koja prati poziciju miša na platnu:

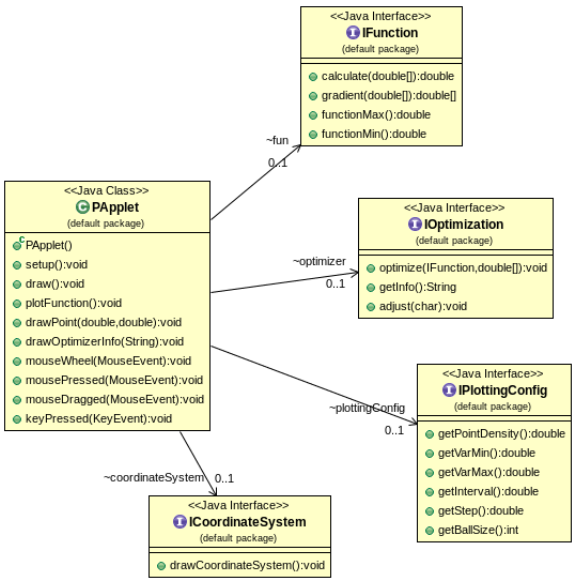
```
void setup() {
  size(400, 400); // Definira velicinu prozora
  loop(); // Definira kontinuirano pozivanje draw metode
  stroke(255); //Definira osnovnu boju pri crtanju
}

void draw() {
  //Definira boju pozadine platna
  background(192, 64, 0);
  //Crta liniju od pocetne pozicije
  //do pozicije misa korisnika
  line(150, 25, mouseX, mouseY);
}
```

3. Ostvareni rad

Kao što je već rečeno u sklopu rada ostvarena je interaktivna 3D vizualizacija gradijentne optimizacije. Vizualizacija se sastoji od 3D grafa funkcije kojoj se traži minimum i kuglice koja označava poziciju trenutnog rješenja. Pomak kuglice po površini funkcije određen je gradijentom u trenutnoj lokaciji kuglice i gradijentnim optimizacijskim algoritmom koji računa idući položaj kuglice.

Susav se sastoji od 5 glavnih komponenti:

- PApplet - glavna klasa koja se brine za crtanje funkcije i trenutne lokacije u optimizaciji.
- ICoordinateSystem - strategija iscrtavanja koordinatnog sustava.
- IPlottingConfig - strategija koja pruža parametre za iscrtavanje.
- IFunction - funkcija koja se prikazuje i optimira
- IOptimization - strategija optimizacijskog postupka.



Slika 3.1: Arhitektura sustava na najvišoj razini

Osnovna interakcija sa sustavom prikazana je u idućoj tablici:

Tablica 3.1: Interakcija sa sustavom

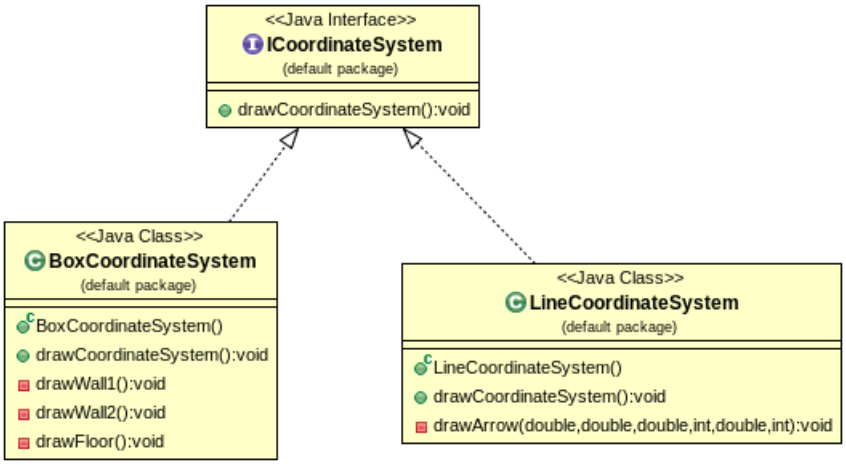
Kontrola	Radnja
Okretanje kotačića miša	zumiranje/odzumiranje
Klik kotačića miša i povlačenje	rotacija oko z osi
Desni klik miša i povlačenje	rotacija oko x osi
Lijevi klik miša i povlačenje	translacija grafa u prostoru
R	Postavlja kuglicu na nasumičnu poziciju u grafu i ponovno pokreće optimizaciju
+	Povećava stopu učenja na trećoj decimali
-	Smanjuje stopu učenja na trećoj decimali
*	Povećava stopu učenja za 2 puta od trenutne vrijednosti
/	Smanjuje stopu učenja za 2 puta od trenutne vrijednosti

Za odabir funkcije koja će se optimirati, postupka te ostalih parametara iscrtavanja potrebno je ručno promijeniti kod u glavnoj klasi. Ručne promjene svode se na zakomentiravanje i odkomentiravanje dijelova koda.

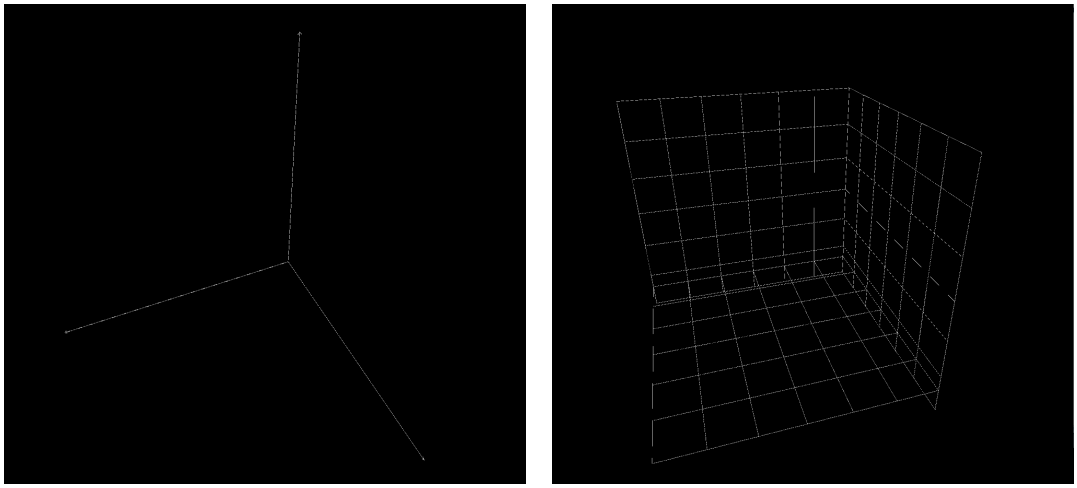
3.1. Koordinatni sustav

U sklopu rada podržana su dva načina prikaza koordinatnog sustava:

- Box koordinatni sustav - prikazuje rešetku po svakoj dimenziji
- Line koordinate sustav - prikazuje samo os po svakoj dimenziji



Slika 3.2: Arhitektura koordinatnog podsustava



(a) Line koordinatni sustav **(b)** Box koordinatni sustav

Slika 3.3: Koordinatni sustavi

3.2. Funkcije

Za dodavanje funkcije potrebno je dodati implementaciju *IFunction* sučelja. Sučelje propisuje 4 funkcije:

- calculate - vraća vrijednost funkcije u danoj točki prostora
- gradient - vraća vrijednost gradijenta u danoj točki prostora
- functionMax - procjena maksimalne vrijednosti koju će funkcija vratiti unutar područja iscrtavanja
- functionMin - procjena minimalne vrijednosti koju će funkcija vratiti unutar područja iscrtavanja

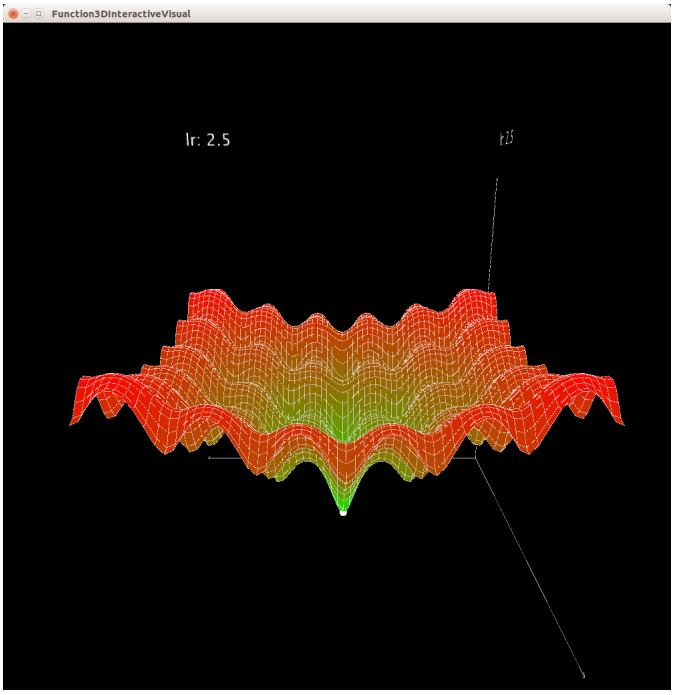
U alat su dodane iduće funkcije:

- Ackley
- Banana funkcija
- Deceptivna
- Eliptic
- Kvadratna pogreška pri učenju
- Matyas
- MultyLocal
- Sigmoid

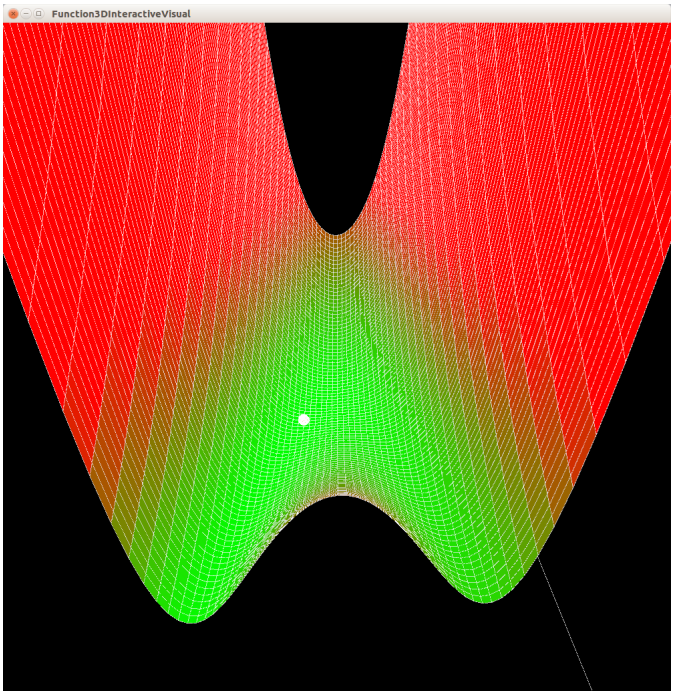
Uz navedene funkcije implementiran je dekorator imena *DerivativeFunctionWrapper* koji izračunava derivaciju funkcije temeljem njenih vrijednosti. Korištenjem ovog dekoratora moguće je brzo iscrtati funkciju jer nije potrebno analitički izračunatni njenu derivaciju. Derivacija se aproksimira koristeći funkciju:

$$f'(x) \approx \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}$$

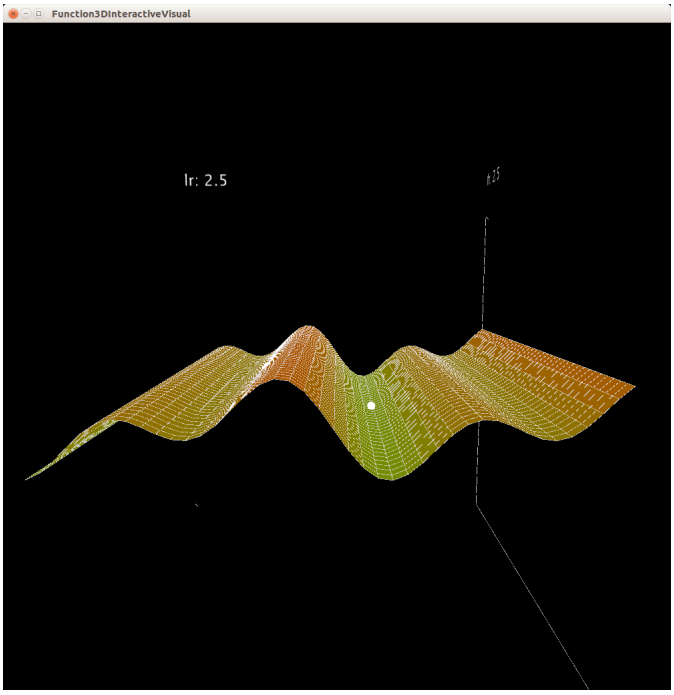
Slika 3.4: Procjena prve derivacije funkcije



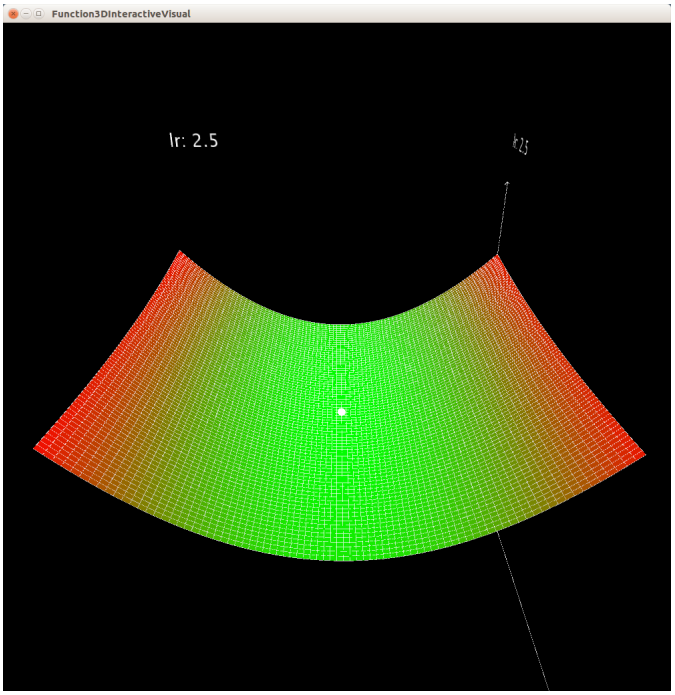
Slika 3.5: Ackley funkcija



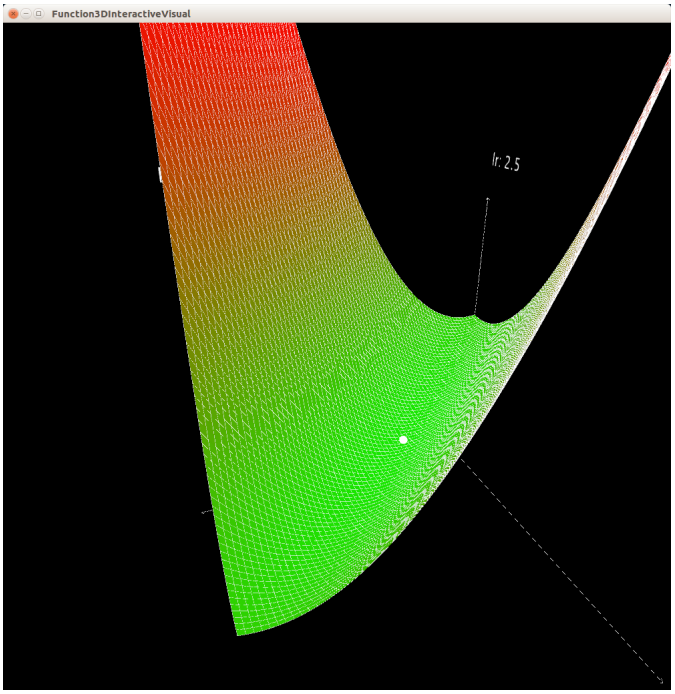
Slika 3.6: Banana funkcija



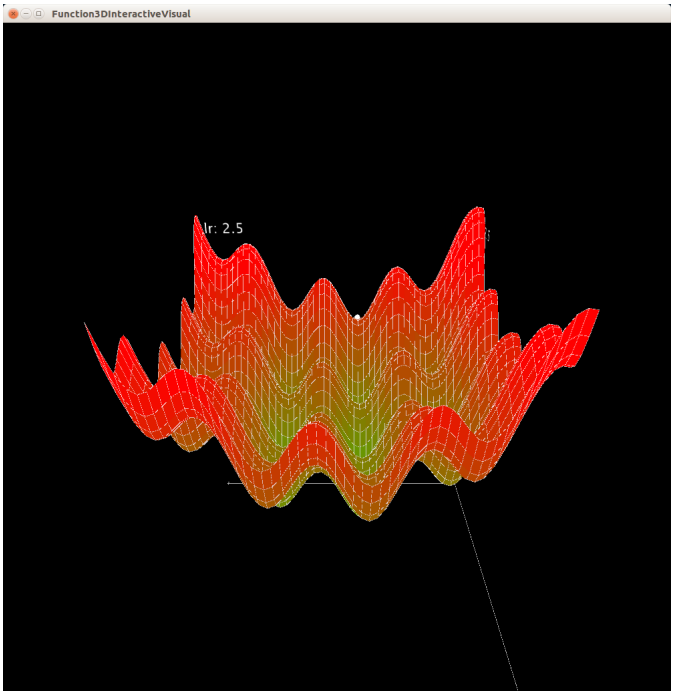
Slika 3.7: Decepcijska funkcija



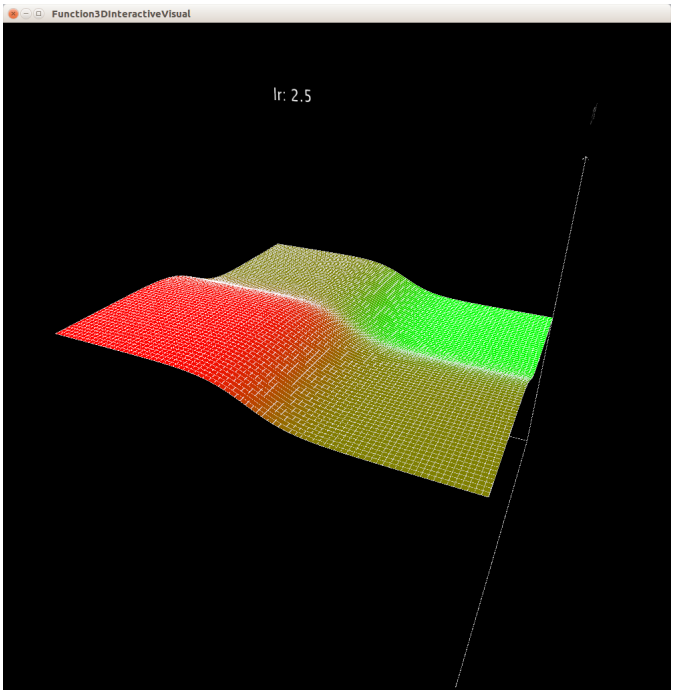
Slika 3.8: Elipsoidna funkcija



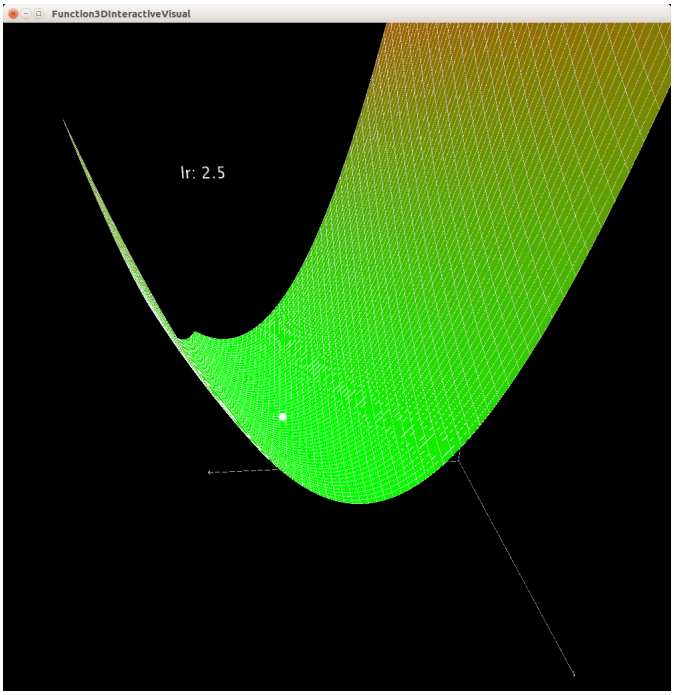
Slika 3.9: Matyas funkcija



Slika 3.10: Funkcija s više lokalnih optimuma



Slika 3.11: Sigmoidalna funkcija



Slika 3.12: Funkcija kvadratne pogreške

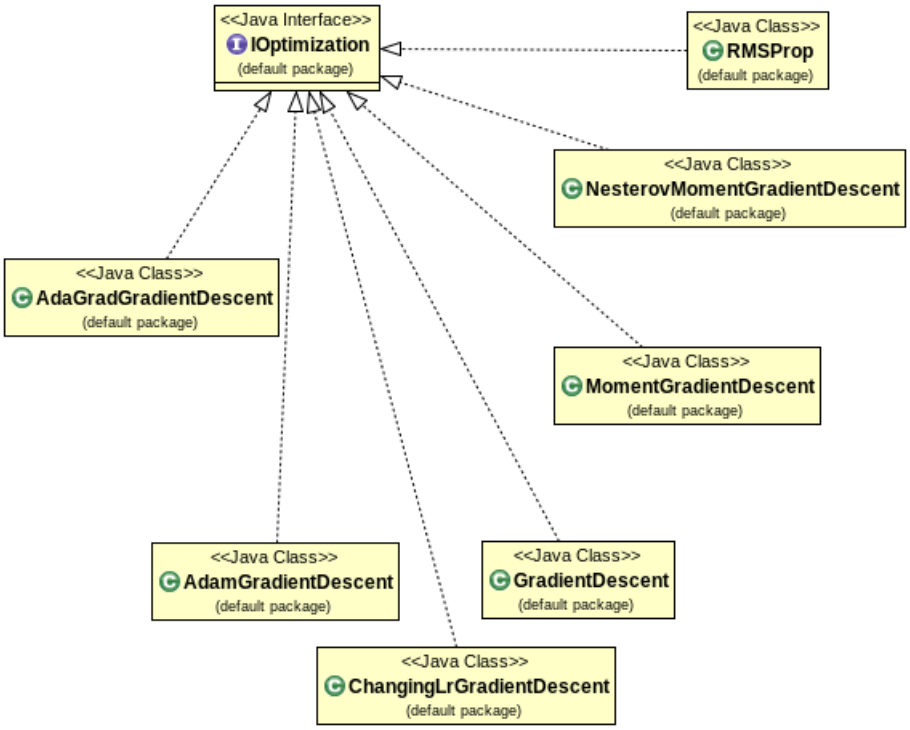
3.3. Optimizacijski postupci

Kako bi se dodao optimizacijski postupak potrebno je implementirati sučelje *IOptimization*. Sučelje propisuje 3 funkcije:

- optimize - prima funkciju i trenutno rješenje te nad njim vrši jednu iteraciju optimizacije
- getInfo - vraća informacije o postupku koje će biti prikazane iznad grafa
- adjust - prima znak tipke stisnute na tipovnici čime se omogućuje definiranje interakcije s korisnikom

U alatu su podržani idući optimizacijski postupci:

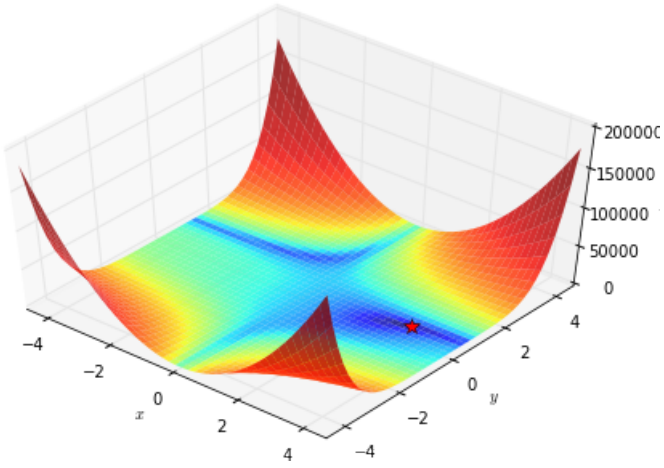
- Adagrad
- Adam
- Gradijentni spust s promjenjivom stopom učenja
- Gradijentni spust s fiksnom stopom učenja
- Gradijentni spust s momentom
- Gradijentni spust s Nesterovljevim momentom
- RMSProp



Slika 3.13: Podržani optimizacijski postupci

4. Slični alati

Opis ostvarenja slične vizualizacije dostupan je na stranici http://louistiao.me/notes/visualizing-and-animating-optimization-algorithms-with-matplotlib/)visualizing-and-animating-optimization-algorithms-with-matplotlib. Ova vizualizacija ostvarena je korištenjem matplotlib biblioteke programskog jezika Python. Osnovna razlika u usporedbi s ovim alatom je u tome što ovo ostvarenje ne nudi mogućnosti interakcije za vrijeme optimizacije. S druge strane vizualizacija ostvarena u Pythonu ima bolje skaliranje osi. Primjer vizualizacije ostvarene pomoću Python alata vidljiv je na idućoj slici:



Slika 4.1: Vizualizacija ostvarena pomoću Python alata preuzeta s (2)

5. Zaključak

Osnovni cilj ovog rada bio je vizualizacijom poboljšati razumijevanje razlika između različitih gradijentnih optimizacijskih postupaka. Taj cilj je svakako ostvaren, ali razvojem ovog alata otvorila su se vrata budućem radu i nadogradnji istog. Budući rad mogao bi uključivati:

- Refaktoriranje PApplet klase prvenstveno u pogledu bolje interakcije s vizualizacijom.
- Skaliranje različitih dimenzija.
- Nadogradnja alata kako bi ga mogli koristiti i ljudi bez računalnih vještina.
- Lakša promjena stopa učenja tijekom vizualizacije.

Želja autora rada je da će ovaj alat poslužiti kao pomoć nastavi budućim generacijama kako bi lakše savladali postupak gradijentne optimizacije i utjecaj stope učenja na postupak.

6. Literatura

[1] Casey Reas, Ben Fry *Processing*, link, 2001

[2] Louis Tiao, *Visualizing and Animating Optimization Algorithms with Matplotlib*, link, 2016