Seminarski rad

za kolegij *Matematika 3*

voditeljica kolegija:
dr.sc. Marija Maksimović,

student:
Tin Švagelj
Jednopredmetna Informatika

Fakultet Informatike i Digitalnih Tehnologija
Rijeka
5. veljače 2023.

Sadržaj

Poglavlje 1

Uvod

Potrebno je odrediti ekstreme i nacrtati funkciju:

$$f(x, y) = 2x^3 + 2y^3 - 36xy + 430 \tag{1.1}$$

Bitno je odrediti **domenu funkcije** na početku kako bismo uspješno i smisleno odredili ekstreme funkcije jer mogu postojati samo unutar njene domene.

Kako bismo mogli odrediti ekstreme funkcije, potrebno je odrediti **nulte točke gradijenta** (∇) funkcije f [1]:

$$\nabla f(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}, \tag{1.2}$$

rješavajući sustav jednadžbi:

$$\begin{cases} \frac{\partial f}{\partial x} = 0 \\ \frac{\partial f}{\partial y} = 0, \end{cases} \tag{1.3}$$

gdje su rješenje nulte točke funkcije gradijenta ($\nabla f(\mathbf{x}, \mathbf{y}) = 0$), tj. ekstremi funkcije f.

Za određivanje parcijalne derivacije prvog reda funkcije koristimo **pravila deriviranja** [2]:

$$(c)' = 0 \text{ i} \tag{1.4}$$

$$(x^a)' = ax^{a-1} \tag{1.5}$$

$$(f + g)' = f' + g'. \tag{1.6}$$

Crtanje će biti izvedeno koristeći `NumPy` i `Matplotlib` biblioteke u Python programskom jeziku.

Poglavlje 2

Razrada

S obzirom da je zadana funkcija f (1.1) racionalna, važeća je za svaki $x \in \mathbb{R}$ i $y \in \mathbb{R}$ [vidi 2, stranica 119]. Dakle domena funkcije je skup svih realnih brojeva:

$$D(f) = \mathbb{R} \times \mathbb{R} = \mathbb{R}^2$$

2.1 Parcijalne derivacije prvog reda

Kako bismo odredili gradijent funkcije f, trebamo prvo odrediti parcijalne derivacije te funkcije po x i y, pri čemu se služimo pravilima 1.4, 1.5 i 1.6:

$$\begin{aligned}
\frac{\partial f}{\partial x} &= \frac{\partial}{\partial x}(2x^3 + 2y^3 - 36xy + 430) \\
&= \frac{\partial}{\partial x}2x^3 + \frac{\partial}{\partial x}2y^3 - \frac{\partial}{\partial x}36xy + \frac{\partial}{\partial x}430 \\
&= 6x^2 + 0 - 36y + 0 \\
&= 6x^2 - 36y
\end{aligned}$$

$$\begin{aligned}
\frac{\partial f}{\partial y} &= \frac{\partial}{\partial y}(2x^3 + 2y^3 - 36xy + 430) \\
&= \frac{\partial}{\partial y}2x^3 + \frac{\partial}{\partial y}2y^3 - \frac{\partial}{\partial y}36xy + \frac{\partial}{\partial y}430 \\
&= 0 + 6y^2 - 36x + 0 \\
&= 6y^2 - 36x.
\end{aligned}$$

S obzirom da je domena zadane funkcije f skup svih realnih brojeva, ne trebamo isključiti rješenja iz dobivenih izraza.

2.2 Crtanje grafa u Pythonu

Za crtanje, kao što je već navedeno u uvodu, koristimo `NumPy` i `Matplotlib`.

`NumPy`eva dokumentacija[3] i `Matplotlib`ova specifikacija sučelja za programiranje aplikacija (engl. API Specification) [4], te upute za korištenje [5], koji su dostupni putem interneta.

`NumPy` omogućava podršku za rad s velikim, višedimenzionalnim poljima i matricama, zajedno sa širokim spektrom visoko-razinskih matematičkih funkcija.

Na početku svakog programa moramo uključiti potrebne module sa sljedećim kodom[6, naslov 5.4.2. Submodules]:

```
import numpy
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

Često se koriste aliasi za sažimanje duljih imena modula koji se učestalo pojavljuju u kodu:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

2.2.1 NumPy

, izvođenjem tih linija koda će Python omogučiti korištenje vanjskog koda u našim programima.

Stvaramo jednodimenzionalne nizove (polja) od 100 uniformno razmaknutih vrijednosti između -10 i 10:

```
x = np.linspace(-10, 10, 100)
y = np.linspace(-10, 10, 100)
```

S obzirom da su ekstremi funkcije (rješenja) dobiveni od parametara koji se nalaze u intervalu $[-10, 10]$, uneseni argumenti funkcije će biti zadovoljavajući.

Nakon izvršavanja tog koda imamo `x` i `y` nizove sa 100 elemenata nalik na:

$$x = y = [-10, -9.8, -9.6, \ldots, 9.8, 10]$$

Zatim od ta dva jednodimenzionalna niza stvaramo dvodimenzionalni niz vrijednosti:

```
X, Y = np.meshgrid(x, y)
```

U našem slučaju se radi o 100×100 nizevima nalik na:

$$\begin{aligned} X = Y^T = [&[-10, -9.8, \ldots, 9.8, 10], \\ &[-10, -9.8, \ldots, 9.8, 10], \\ &\vdots \\ &[-10, -9.8, \ldots, 9.8, 10]] \end{aligned}$$

`NumPy` nam dozvoljava simboličko izražavanje funkcija, pa za računanje `Z` vrijednosti grafa, tj. rezultata funkcije f koji su nam potrebni za crtanje grafa, možemo koristiti:

```
Z = 2*X**3 + 2*Y**3 - 36*X*Y + 430
```

, što će generirati novi dvodimenzionalni niz (dimenzija 100×100) koji zadovoljava jednadžbu funkcije 1.1.

To je jedina linija koda koju moramo mjenjati za crtanje različitih grafova u ovom slučaju.

`NumPy` u pozadini računa sve vrijednosti rezultirajućeg dvodimenzionalnog niza za sve odgovarajuće parove[1] `X` i `Y` vrijednosti, tj. za sve kombinacije `x` i `y` vrijednosti.

Time dobivamo dvodimenzionalni niz svih vrijednosti koje funkcija može poprimiti za sve kombinacije x i y u intervalima $[-10, 10]$:

$$f(x, y) \mid x \in [-10, 10] \mid y \in [-10, 10]$$

Razlog zašto možemo na tako prirodan način izraziti operacije nad multidimenzionalnim nizovima pomoću `NumPy`a je zato što su `X` i `Y` apstraktne reprezentacije nizeva za koje su definirani[2] svi matematički operatori koje Python podržava za normalne brojeve.

Tako da nam `X**3` prvo daje novi niz gdje je svaki element/broj iz niza `X` eksponenciran brojem 3. Zatim je izraz `2*X**3` evaluiran i zbog množenja rezultata sa 2, `NumPy` množi sve elemente niza sa 2. Na kraju vrši binarne operacije nad samim nizovima gdje dobivamo rješenja polinoma za cijeli izraz, te će taj rezultatski niz biti pohranjen u `Z`.

Sada kada imamo sve vrijednosti koje funkcija poprima, možemo koristiti `Matplotlib` za crtanje grafa. Za početak stvaramo

```
Z = 2*X**3 + 2*Y**3 - 36*X*Y + 430
```

2.2.2 Matplotlib

`Matplotlib` pruža funkcije za crtanje širokog raspona statičkih, animiranih i interaktivnih vizualizacija. Može se koristiti za stvaranje dijagrama stupčastih grafova, linijskih grafova, grafova raspršenja, pogrešnih traka, histograma, dijagrama kružnica, dijagrama kutije i mnogih drugih vrsta vizualizacija.

U napisanom kodu stvaramo novu figuru [3] za crtanje te ju pohranjujemo u `fig` varijablu:

```
fig = plt.figure()
```

Stvaramo 3D podgraf unutar figure i pohranjujemo ga u `ax` varijablu:

```
fig.add_subplot(111, projection='3d')
```

Prvi argument (`111`) određuje poziciju podgrafa unutar figure:

- prvi broj određuje broj redaka,
- drugi broj stupaca,

[1] Parovi u rezultatu su usklađeni na osnovu stupca i retka

[2] Definirani kao preoptrećenja operatora u Pythonu

[3] Figura (engl. Figure) je u Matplotlib biblioteci cijelokupni prostor za crtanje koji može sadržavati jedan ili više grafova.

- a treći broj indeks podgrafa.

U ovom slučaju, `111` stvara jedan podgraf koji se proteže cijelom figurom. Ovaj argument nije obavezan jer smo koristili zadanu vrijednost, no naveden je u svrhu boljeg objašnjavanja figura.

Argument (`projection`) specificira projekciju grafa, s obzirom da se radi o 3D grafu koristimo vrijednost `'3d'`. Potrebno je navesti projekciju jer je zadana vrijednost `'rectilinear'` koja je zapravo ortogonalna projekcija usmjerena u ravninu s rješenjima $z = 0$ i koristi se za crtanje dvodimenzionalnih grafova. Argumentom `'3d'` navodimo da očekujemo ortogonalnu projekciju trodimenzionalnog grafa, koja gleda u smjeru:

$$\vec{v} = (\textit{azimut}, \textit{elevacija}) = (-60°, 30°)$$

U slučaju da želimo izmjeniti kut projekcije, to možemo postignuti korištenjem `mpl_toolkits.mplot3d.axes3d.Axes3D.view_init()` funkcije:

```
ax.view_init(elev=zeljenja_elevacija, azim=zeljeni_azimut)
```

, no rezultati dobiveni zadanim vrijednostima su dovoljno jasni pa to nije potrebno u ovom slučaju.

2.3 Gradijent funkcije

Određujemo gradijent funkcije f uz dobivene parcijalne derivacije na osnovu formule 1.2:

$$\nabla f(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} 6x^2 - 36y \\ 6y^2 - 36x \end{bmatrix}, \tag{2.1}$$

2.4 Ekstremi

Gradijent 2.1 izjednačujemo s nulom kako bi mu odredili nulte točke tj. ekstreme funkcije. To možemo izraziti kao sustav (1.3):

$$\begin{cases} 6x^2 - 36y = 0 \\ 6y^2 - 36x = 0. \end{cases}$$

Izrazimo y iz prvog izraza,

$$\begin{aligned} 6x^2 - 36y &= 0 \\ 36y &= 6x^2 \\ y &= \frac{6x^2}{36} = \frac{x^2}{6} \end{aligned}$$

te ga uvrštavamo u drugi kako bismo dobili jednadžbu za $x$eve nultih točaka gradijenta:

$$\begin{aligned} 6(\frac{x^2}{6})^2 - 36x &= 0 \\ x^4 - 36x &= 0. \end{aligned} \tag{2.2}$$

Jedno rješenje (x_1) uočavamo nakon izlučivanja xa iz izraza:

$$\begin{aligned} x(x^3 - 36) &= 0 \\ x_1 &= 0. \end{aligned}$$

Zaključujemo da je jedno rješenje $x_1 = 0$ jer ako je x jednak nuli, onda će jednadžba 2.2 vrijediti.

Drugi član umnoška nam daje drugo rješenje:

$$\begin{aligned} x^3 - 36 &= 0 \\ x^3 &= 36 \\ x &= \sqrt[3]{36}, \end{aligned}$$

ali zbog duplog pojavljivanja xa u zadanom polinomu se radi o funkciji simetričnoj s obzirom na ishodište te je zbog toga dobiveno rješenje zapravo dvojno rješenje:

$$x_{2,3} = \pm\sqrt[3]{36}.$$

Time dobivamo 3 rješenja sustava za x:

$$x_1 = 0 \qquad x_2 = \sqrt[3]{36} \qquad x_3 = -\sqrt[3]{36}$$

Za $x_1 = 0$, y_1 će biti 0. y_2 i y_3 računamo uvrštavajući dobivene vrijednosti za x sa jednom od parcijalnih derivacija:

$$\begin{aligned} 6x^2 - 36y &= 0 \\ 6(\sqrt[3]{36})^2 - 36y &= 0 \\ 36y &= 6(\sqrt[3]{36^2}) \\ 6y &= \sqrt[3]{36^2} \\ y &= \frac{\sqrt[3]{6^4}}{6} \\ y &= 6^{\frac{4}{3}} * 6^{-1} = 6^{\frac{4}{3}} * 6^{-\frac{3}{3}} = 6^{\frac{4}{3}-\frac{3}{3}} \\ y &= \sqrt[3]{6} \end{aligned}$$

Time utvrđujemo da su ekstremi funkcije f:

x	y
0	0
$\sqrt[3]{36}$	$\sqrt[3]{6}$
$-\sqrt[3]{36}$	$-\sqrt[3]{6}$

Poglavlje 3

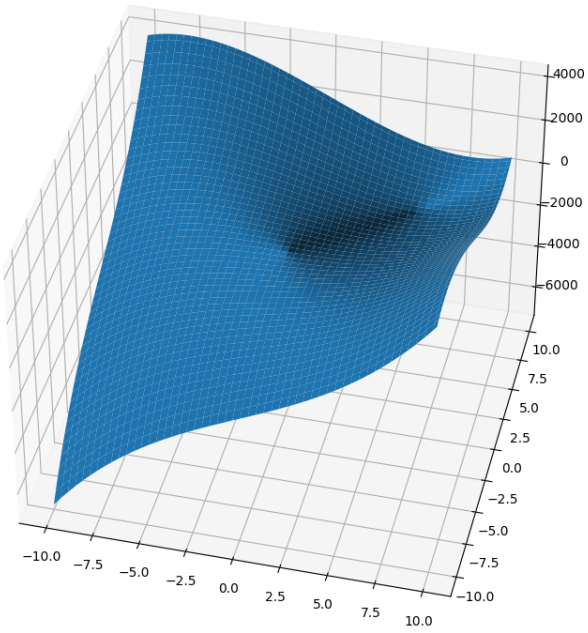
Zaključak

Korištenjem `NumPy` i `Matplotlib` biblioteka u Pythonu, pomoću sljedećeg koda možemo dobiti 3D prikaz cijele multivarijatne funkcije:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

x = np.linspace(-10, 10, 100)
y = np.linspace(-10, 10, 100)
X, Y = np.meshgrid(x, y)
Z = 2*X**3 + 2*Y**3 - 36*X*Y + 430
print(Z)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z)

plt.savefig("figures/graf_fje.png")
```

Na grafu vidimo da dobivene točke jesu ekstremi funkcije.

Za algebarsko određivanje o kakvim se ekstremima radi bi trebali odrediti Hessovu matricu i parcijalne derivacije višeg reda [1, poglavlje 6.3], no to nije traženo u sklopu ovog zadatka.



Slika 3.1: Graf zadane funkcije f

Literatura

[1] Ken Binmore i Joan Davies. *Calculus: Concepts and Methods*. Cambridge University Press, 2002. DOI: 10.1017/CBO9780511802997.

[2] T. Hunjak B. Divjak. *Matematika za informatičare*. Varaždin: Fakultet organizacije i informatike, tiskara TIVA, 2004.

[3] *NumPy v1.24 Manual*. [Internet; pristupljeno 4. Feb. 2023]. Prosinac 2022. URL: https://numpy.org/doc/stable.

[4] *Matplotlib 3.6.3 API Reference*. [Internet; pristupljeno 4. Feb. 2023]. Siječanj 2023. URL: https://matplotlib.org/stable/api/index.

[5] *Matplotlib 3.6.3 Users guide*. [Internet; pristupljeno 4. Feb. 2023]. Siječanj 2023. URL: https://matplotlib.org/stable/users/index.

[6] *The Python Language Reference*. [Internet; pristupljeno 5. Feb. 2023]. Veljača 2023. URL: https://docs.python.org/3/reference.