VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

Netiesinis programavimas

Trečio laboratorinio darbo ataskaita

Atliko: Paulina Podgaiska

VU el. p.: paulina.podgaiska@mif.stud.vu.lt

Vertino: Vyr. M. Darbuot., Dr. (HP), Julius Žilinskas

Vilnius

2025

UŽDUOTIS

Trečio laboratorinio darbo užduotis pateikta lentelėje 1

1lentelė. Trečio laboratorinio darbo užduotis

Netiesinis programavimas
1. Aprašykite tikslo funkciją $f(X)$, lygybinio ir nelygybinių apribojimų funkcijas $g_i(X)$ ir $h_i(X)$ taip, kad optimizavimo uždavinys būtų formuluojamas min $f(X)$, $g_i(X)=0$, $h_i(X)\leq 0$.
2. Apskaičiuokite funkcijų $f(X)$, $g_i(X)$, $h_i(X)$ reikšmes taškuose $X_0=(0,0,0)$, $X_1=(1,1,1)$ ir $X_m=(a/10,b/10,c/10)$, čia a,b,c – studento knygelės numerio “1x1xabc” skaitmenys.
3. Aprašykite kvadratinę baudos funkciją, apimančią tikslo funkciją ir apribojimus.
4. Patyrinėkite baudos daugiklio įtaką baudos funkcijos reikšmėms.
5. Minimizuokite baudos funkciją praeitame laboratoriniame darbe sukurtu optimizavimo be apribojimų algoritmu sprendžiant optimizavimo uždavinių seką su mažėjančia parametro r seka, kai pirmasis sekos uždavinys optimizuojamas pradedant iš taškų X_0, X_1 ir X_m, o kiekvieno paskesnio uždavinio pradinis taškas yra ankstesnio uždavinio sprendinys.
6. Palyginkite rezultatus: gauti sprendiniai, rastas funkcijos minimumo įvertis, atliktų žingsnių ir funkcijų skaičiavimų skaičius priklausomai nuo pradinio taško.

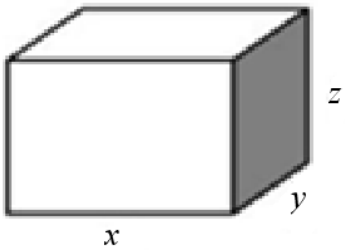
ATASKAITA

1. DARBO EIGA

Užduoties klausimas:

Kokia turėtų būti stačiakampio gretasienio formos dėžė, kad vienetiniam paviršiaus plotui jos tūris būtų maksimalus?

Laikant stačiakampio gretasienio briaunas x, y, z (1 pav.)



1 pav. Stačiakampio gretasienio briaunų žymėjimas

Čia tūris: V=xyz. Viso paviršiausiaus plotas yra vienetinis, t.y. 2xy+2xz+2yz=1. Čia tikslo funkcija $f(X)$ yra

$$f(X) = x * y * z$$

Lygybės funkcija $g(X)$ yra

$$g(X) = 2xy + 2yz + 2xz - 1$$

Nelygybės funkcija $h(X)$ yra

$$g(X) = 2xy + 2yz + 2xz - 1$$

Baudos funkcija yra

$$B(X, r) = f(X) + \frac{1}{r} b(X)$$

$$b(X) = g(X)^2 + \sum_{i=1}^{3} \max(0, h_i(X))^2$$

Čia r – (10.0, 1.0, 0.5, 0.1, 0.01, 0.001).

Apskaičiuotos funkcijos reikšmės buvo įvertintos taškuose: X_0= (0;0), X_1=(1;1) ir X_m= (0,9; 0,9). Gauti rezultatai, bei programos kodas, atliktas naudojant „Python“ programavimo kalbą, aprašyti žemiau (programos kodas pateiktas priede).

1.1. Optimizavimo metodas

Optimizavimui buvo taikytas gradientinio nusileidimo algoritmas, paremtas antrojo laboratorinio darbo metu realizuotu metodu. Šiame darbe jis buvo pritaikytas baudos funkcijos $B(x, r)$ minimizavimui. Metodo esmė – kiekviename žingsnyje judėti priešinga kryptimi nei funkcijos gradientas, t. y. antigradiento kryptimi, kuria funkcijos reikšmė greičiausiai mažėja.

Paieška pradedama nuo pasirinkto pradinio taško (X_0, X_1, X_m). Kiekviename iteracijos žingsnyje skaičiuojamas baudos funkcijos gradientas ir atliekamas žingsnis pagal formulę:

$$X_{i+1} = X_i - \alpha * \nabla B(X_i)$$

Čia X_i – pradinis taškas, α –žingsnio dydis, $\nabla B(x_i)$ – baudos funkcijos gradientas. Naudojamas backtracking line search – tai reiškia, kad žingsnis α sumažinamas, jei funkcijos reikšmė nesumažėja pakankamai.

Baudos funkcija buvo minimizuojama kiekvienai reikšmei

$$r \in \{10.0, 1.0, 0.5, 0.1, 0.01, 0.001\}$$

Kiekvienam r skaičiui minimizuotas $B(X, r)$, o sekos pradinis taškas imamas iš ankstesnio sprendinio – jei r_i yra baudos parametras, tai sprendžiame uždavinį su $B(X, r_i)$ pradedant nuo ankstesnio X_{i-1}.

Lentelė su pradiniais duomenimis pavaizduota 2 pav.

Taškas	x	y	z	f(X)	g(X)	h(X)
X0	0	0	0	-0	-1	[-0.0, -0.0, -0.0]
X1	1	1	1	-1	5	[-1.0, -1.0, -1.0]
Xm	0.9	0.9	0.9	-0.729	3.86	[-0.9, -0.9, -0.9]

2 pav. Pradiniai taškai X_0, X_1, X_m ir funkcijų reikšmės taškuose

1.2. Rezultatai

Eksperimentų metu kiekvienam pradiniam taškui (X_0, X_1, X_m) buvo atlikta optimizacija su skirtingomis baudos parametro r reikšmėmis:

$$r \in \{10.0, 1.0, 0.5, 0.1, 0.01, 0.001\}$$

Rezultatai rodo, kad mažėjant baudos parametrui r, reikšmė $g(X)$ artėja prie nulio – tai reiškia, kad paviršiaus ploto apribojimas vis geriau tenkinamas. Tuo pačiu, kai r tampa labai mažas, išauga iteracijų skaičius – optimizavimas tampa sudėtingesnis, bet gauname tikslesnius rezultatus. Rezultatai pateikti 3–5 pav., kuriuose matoma, kaip keičiasi sprendiniai priklausomai nuo pradinio taško ir r reikšmės.

r	x	y	z	f(X)	g(X)	Iteracijos	Skaičiavimai
10	0.5255	0.5255	0.5255	-0.1451	0.6569	1188	1188
1	0.4188	0.4188	0.4188	-0.0735	0.0523	462	462
0.5	0.4135	0.4135	0.4135	-0.0707	0.0258	144	144
0.1	0.4093	0.4093	0.4093	-0.0686	0.0051	150	150
0.01	0.4084	0.4084	0.4084	-0.0681	0.0005	364	364
0.001	0.4083	0.4083	0.4083	-0.068	0	15000	14999

3 pav. Rasti sprendiniai nuo pradinio taško X_0 su skirtingais r

r	x	y	z	f(X)	g(X)	Iteracijos	Skaičiavimai
10	0.5255	0.5255	0.5255	-0.1451	0.6569	1084	1084
1	0.4188	0.4188	0.4188	-0.0735	0.0523	462	462
0.5	0.4135	0.4135	0.4135	-0.0707	0.0258	144	144
0.1	0.4093	0.4093	0.4093	-0.0686	0.0051	150	150
0.01	0.4084	0.4084	0.4084	-0.0681	0.0005	1109	1109
0.001	0.4083	0.4083	0.4083	-0.068	0	6269	6269

4 pav. Rasti sprendiniai nuo pradinio taško X_1 su skirtingais r

r	x	y	z	f(X)	g(X)	Iteracijos	Skaičiavimai
10	0.5255	0.5255	0.5255	-0.1451	0.6569	805	805
1	0.4188	0.4188	0.4188	-0.0735	0.0523	462	462
0.5	0.4135	0.4135	0.4135	-0.0707	0.0258	144	144
0.1	0.4093	0.4093	0.4093	-0.0686	0.0051	150	150
0.01	0.4084	0.4084	0.4084	-0.0681	0.0005	1109	1109
0.001	0.4083	0.4083	0.4083	-0.068	0	1427	1427

5 pav. Rasti sprendiniai nuo pradinio taško X_m su skirtingais r

1.3. Išvados

Uždavinys išsprendžiamas naudojant baudos funkcijų metodą kartu su gradientinio nusileidimo algoritmu. Gradientinis nusileidimas su backtracking linijine paieška užtikrino stabilų konvergavimą net ir esant griežtiems apribojimams. Kuo mažesnė baudos parametro reikšmė r, tuo tiksliau tenkinami apribojimai, tačiau tuo pačiu daugėja iteracijų ir funkcijos kvietimų skaičius. Pradinis taškas X_m buvo efektyviausias, lyginant su X_0 ir X_1.

Rezultatų apibendrinimui sukurta 2 lentelė, kurioje pateikti geriausi rezultatai.

2 lentelė. Įgyvendinto metodo rezultatų palyginimas nuo skirtingų pradinių taškų

Pradinis taškas	Rasto minimumo taško f(x)	r reikšmė	Rastas minimumo taškas	$g(X)$	Iteracijų skaičius	Funkcijos įvertinimų skaičius
X_0= (0;0;0)	-0,068	0,001	(0,4083;0,4083;0,4083)	0	15000	14999
X_1=(1;1;1)	-0,068	0,001	(0,4083;0,4083;0,4083)	0	6269	6269
X_m= (0,9; 0,9;0,9)	-0,068	0,001	(0,4083;0,4083;0,4083)	0	1427	1427

2. PRIEDAS

Žemiau pateikiamas kodas su anksčiau aprašytais įgyvendintais algoritmais.

3_laboratorinis.py

```
import numpy as np
from tabulate import tabulate # type: ignore

X0 = np.array([0.0, 0.0, 0.0])
X1 = np.array([1.0, 1.0, 1.0])
Xm = np.array([9 / 10, 9 / 10, 9 / 10])

def f(X):
    x, y, z = X
    return -x * y * z

def g(X):
    x, y, z = X
    return 2 * (x * y + y * z + x * z) - 1

def h(X):
    x, y, z = X
    return [-x, -y, -z]

def B(X, r):
    bauda_kv = g(X)**2 + sum(max(0, hi)**2 for hi in h(X))
    return f(X) + (1.0 / max(r, 1e-4)) * bauda_kv

def grad_B(X, r, h_step=1e-5):
    grad = np.zeros(3)
    fx = B(X, r)
    for i in range(3):
        X_step = X.copy()
        X_step[i] += h_step
        grad[i] = (B(X_step, r) - fx) / h_step
    return grad

def gradient_nusileidimas(f, grad_f, X0, zings=0.01, epsilon=1e-6, max_iter=15000):
    X = np.array(X0, dtype=float) + 1e-4
    kelias = [X.copy()]
    i = 1
    kvietimai = 0
    while i < max_iter:
        grad = grad_f(X)
        kvietimai += 1
        grad_norm = np.linalg.norm(grad)
        if grad_norm <= epsilon:
            break
        grad = grad / grad_norm

        alpha = zings
        while f(X - alpha * grad) > f(X) - 1e-4 * alpha * grad_norm**2 and alpha > 1e-
8:
            alpha *= 0.7

        X = X - alpha * grad
        X = np.clip(X, 1e-6, 1e+2)
        kelias.append(X.copy())
        i += 1
    return X, f(X), i, kvietimai, np.array(kelias)

def funkciju_reiks():
```

```
    print(" Pradiniai taskai ir funkciju reiksmes ")
    duom = []
    for name, X in zip(["X0", "X1", "Xm"], [X0, X1, Xm]):
        gx = g(X)
        hx = h(X)
        duom.append([
            name,
            *[round(x, 4) for x in X],
            round(f(X), 3),
            round(gx, 3),
            "[" + ", ".join(f"{hi:.1f}" for hi in hx) + "]"
        ])
    antrast = ["Taškas", "x", "y", "z", "f(X)", "g(X)", "h(X)"]
    print(tabulate(duom, headers=antrast, tablefmt="fancy_grid"))

def sprendiniai(X_start):
    r_visi = [10.0, 1.0, 0.5, 0.1, 0.01, 0.001]
    rez = []
    X_dabartinis = np.array(X_start)
    for r in r_visi:
        grad_f_r = lambda X: grad_B(X, r)
        f_r = lambda X: B(X, r)
        X_opt, f_val, iters, kv, _ = gradient_nusileidimas(f_r, grad_f_r,
X_dabartinis, zings = 0.001 if r > 0.01 else 0.01)
        X_dabartinis = X_opt
        rez.append([
            f"{r:.1e}",
            *[round(xi, 4) for xi in X_opt],
            round(f(X_opt), 4),
            round(g(X_opt),4),
            iters,
            kv
        ])
    return rez

def visi_sprend():
    pradiniai_task = {"X0": X0, "X1": X1, "Xm": Xm}
    for label, X in pradiniai_task.items():
        print(f"\n--- Lentele su pradiniu tasku {label} ---")
        lentel = sprendiniai(X)
        antrast = ["r", "x", "y", "z", "f(X)", "g(X)", "Iteracijos", "Skaičiavimai"]
        print(tabulate(lentel, headers=antrast, tablefmt="fancy_grid"))

if __name__ == "__main__":
    funkciju_reiks()
    visi_sprend()
```