Vilniaus universitetas

Matematikos ir informatikos fakultetas

Programų sistemų studijų programa

Optimizavimo metodų trečiojo laboratorinio darbo ataskaita

Ataskaitą tikrino: Prof. Dr. Pranas Katauskis

Ataskaitą parengė: Dominykas Daunoravičius

Vilnius

2022

Įvadas

Laboratorinio darbo formulavimas: Apsirašyti tikslo funkciją f(X), lygybinio ir nelygybinių apribojimų funkcijas $g_i(X)$ ir $h_j(X)$ taip, kad optimizavimo uždavinys būtų formuluojamas min f(X), $g_i(X)=0$, $h_i(X)<=0$. Apskaičiuoti funkcijų f(X), $g_i(X)$, $h_j(X)$ reikšmes taškuose $X_0=(0,0,0)$, $X_1=(1,1,1)$, $X_m=(a/10,b/10,c/10)$, kur a = 0, b = 0.3, c = 0.9. Taip pat reikia minimizuoti baudos funkciją praeitame laboratoriniame darbe sukurtu optimizavimo be apribojimų algoritmu, sprendžiant optimizavimo uždavinių seką su mažėjančia parametro r seka.

Laboratorinio darbo tikslas: Rasti kokie turėtų būti stačiakampio gretasienio matmenys, kad vienetiniam paviršiaus plotui jos tūris būtų maksimalus.

Darbo eiga

Laboratoriniui darbui atlikti, kaip pagrindinę funkciją naudojau tūrio formulę, lygybinį apribojimą (g_1) suformulavau iš paviršiaus ploto atėmęs vienetą, o nelygybiniai apribojimai (h_j) tokie, kad kintamieji turi būti neneigiami.

a, b ,c – kraštinių ilgiai, X = (a,b,c)

Tūrio funkcija:
f(X) = a * b * c

Lygybinis apribojimas, jog paviršiaus plotas turi būti lygus 1:
$g_1(X) = 2ab + 2ac + 2bc – 1$

Nelygybiniai apribojimai, jog kraštinės turi būti neneigiamos:
$h_1(X) = -a$
$h_2(X) = -b$
$h_3(X) = -c$

Kadangi reikia ieškoti funkcijos minimumo, mums reikia f(X) padauginti iš -1 ir kaip tikslo funkciją gauname:
f(X) = -1 * a * b * c

Apsirašius funkcijas, lygybinius bei nelygybinius apribojimus reikėjo apskaičiuoti šių funkcijų reikšmes taškuose $X_0 = (0, 0, 0)$, $X_1 = (1, 1, 1)$ ir $X_m = (0, 0.3, 0.9)$.

1 lentelė. Funkcijų reikšmės taškuose X_0, X_1, X_m.

Funkcija	X_0	X_1	X_m
f(X)	0	-1	0
$g_1(X)$	-1	5	-0.4599999999999999
$h_1(X)$	0	-1	0

$h_2(X)$	0	-1	-0.3
$h_3(X)$	0	-1	-0.9

Vėliau reikėjo apsirašyti baudos metodo funkciją, kurią reikės minimizuoti:
$B(X, r) = f(X) + 1/r * b(X)$, kur $r > 0$, $b(X) = \sum[g_i(X)]^2 + \sum[max(0, h_j(X))]^2$

Apsirašius visas reikalingas funkcijas galima buvo pradėti minimizuoti baudos funkciją. Šios funkcijos minimizavimui pasirinkau naudoti simplekso metodą, kurį suprogramavau praeitame laboratoriniame darbe.

Pagrindinės iteracijų nutraukimo sąlygos:

1. Simplekso metode – jei pasiekiamas norimas tikslumas epsilon (atstumas tarp blogiausio ir geriausio taško) arba pasiektas maksimalus iteracijų kiekis, kuris yra 1000.
2. Pagrindiniam optimizavimo metode – jei pasiektas norimas tikslumas epsilon (atstumas tarp dabartinės ir praeitos interacijos taško) arba pasiektas maksimalus iteracijų kiekis, kuris yra 100.

Visus skaičiavimus atliksime iki tikslumo epsilon = 0.0001, o deformuojamo simplekso metodo parametrai bus alpha=0.5, gama=3, beta=0.5, niu=0.5.

Skaičiuojant iš pradinių taškų X_0 ir X_1 naudosime pradinį r = 4 ir kiekvienos iteracijos metu r dauginsime iš 0.5 (baudos daugiklį toliau žymėsime raide q). Skaičiuojant iš taško X_m patyrinėsim r ir q įtaką rezultatam.

Skaičiuojame minimumą taške X_0. Prireikė 15 iteracijų ir 1320 funkcijų skaičiavimų.

2 lentelė. Gauti rezultatai po pirmų ir paskutinių trijų iteracijų iš pradinio taško X_0, r =4, q = 0.5.

Iteracija	X	B(X, r)	f(X)	r
1	(0.4690946517791426, 0.5316785551929821, 0.328599340045153)	-0.07583038051024814	-0.08195516182208569	4
2	(0.4312846082959807, 0.4292374753005549, 0.42576356370152163)	-0.07351384019215565	-0.0788188480678279	2
3	(0.41876795226736097, 0.4188233408550708, 0.41880913932836134)	-0.07071313586158155	-0.07345484817442798	1
13	(0.4089371713203966, 0.40812929782477647, 0.40770920697856383)	-0.06804384537872707	-0.06804635702441582	0.0009765625
14	(0.40830719274899957, 0.4081562978675768, 0.40829725486317126)	-0.06804264986258296	-0.06804402455148484	0.00048828125

15	(0.4082663373112877, 0.40816534564300444, 0.40832035160957214)	-0.06804201381476643	-0.0680425730855883	0.000244140625

Skaičiuojame minimumą taške X_1. Taip pat prireikė 15 iteracijų ir 1320 funkcijų skaičiavimų.

3 lentelė. Gauti rezultatai po pirmų ir paskutinių trijų iteracijų iš pradinio taško X_1, r =4, q = 0.5.

Iteracija	X	B(X, r)	f(X)	r
1	(0.4690946517791426, 0.5316785551929821, 0.328599340045153)	-0.07583038051024814	-0.08195516182208569	4
2	(0.4312846082959807, 0.4292374753005549, 0.42576356370152163)	-0.07351384019215565	-0.0788188480678279	2
3	(0.41876795226736097, 0.4188233408550708, 0.41880913932836134)	-0.07071313586158155	-0.07345484817442798	1
13	(0.4089371713203966, 0.40812929782477647, 0.40770920697856383)	-0.06804384537872707	-0.06804635702441582	0.0009765625
14	(0.40830719274899957, 0.4081562978675768, 0.40829725486317126)	-0.06804264986258296	-0.06804402455148484	0.00048828125
15	(0.4082663373112877, 0.40816534564300444, 0.40832035160957214)	-0.06804201381476643	-0.0680425730855883	0.000244140625

Skaičiuojame minimumą taške X_m. Metodui prireikė 15 iteracijų ir 1289 funkcijų skaičiavimų.

4 lentelė. Gauti rezultatai po pirmų ir paskutinių trijų iteracijų iš pradinio taško X_m, r =4, q = 0.5.

Iteracija	X	B(X, r)	f(X)	r
1	(0.4955309255077497, 0.36946672015547993, 0.46164950808765276)	-0.0777287351671542	-0.08451980100631469	4
2	(0.4217102940175068, 0.43122458697772437, 0.43451344464810304)	-0.07351223063298569	-0.07901707261286373	2
3	(0.42283030313589376, 0.4205592405672262, 0.4139411042070947)	-0.07070636232717999	-0.07360915599108277	1
13	(0.4084943297378064, 0.40797915559756204, 0.40830028904353366)	-0.06804390324496762	-0.06804617138152319	0.0009765625
14	(0.4082934216943052, 0.4083601109184728, 0.4081066333139173)	-0.06804264979961057	-0.06804392381599202	0.00048828125

15	(0.40825951383821235, 0.40833434471028074, 0.4081587537098548)	-0.06804201583239787	-0.06804266872683354	0.000244140625

Metodas atlieka gan daug iteracijų, todėl bandome r mažinti staigiau po kiekvienos iteracijos ir q keičiame į 0.25. Metodui prireikė 12 iteracijų ir 1054 funkcijų skaičiavimų.

5 lentelė. Gauti rezultatai po pirmų ir paskutinių trijų iteracijų iš pradinio taško X_m, r =4, q = 0.25.

Iteracija	X	B(X, r)	f(X)	r
1	(0.4955309255077497, 0.36946672015547993, 0.46164950808765276)	-0.0777287351671542	-0.08451980100631469	4
2	(0.41439671429241937, 0.42098249278786437, 0.4202056664811341)	-0.07070855131652025	-0.07330645924139483	1
3	(0.4071241156762574, 0.4091270660052475, 0.4145280233674291)	-0.06865805672077817	-0.069046065381435	0.25
10	(0.40811025107297494, 0.4083503754405561, 0.40828457994725686)	-0.06804141382816344	-0.06804143130273128	0.00001525878
11	(0.40806869978285076, 0.40846443791777287, 0.40821194450580156)	-0.06804138222571174	-0.06804140049134819	0.000003814697
12	(0.40811441514765046, 0.4083961431183335, 0.40823448333930723)	-0.06804134256596948	-0.06804140195362074	0.0000009536743

Pabandome q pakeisti į 0.75 ir gauname, jog prireikia net 32 iteracijų ir 2450 funkcijų skaičiavimų.

6 lentelė. Gauti rezultatai po pirmų ir paskutinių trijų iteracijų iš pradinio taško X_m, r =4, q = 0.75.

Iteracija	X	B(X, r)	f(X)	r
1	(0.4955309255077497, 0.36946672015547993, 0.46164950808765276)	-0.0777287351671542	-0.0845198010063146	4
2	(0.44069793269754853, 0.4406980474601202, 0.4406516815312511)	-0.0764832934022555	-0.0855810422673064	3
3	(0.4325118243412226, 0.43236204170422987, 0.43215726716340286)	-0.0742502854285099	-0.0808141416534177	2.25
30	(0.4083262321507324, 0.40833436937370415, 0.408113787400889)	-0.068043858719596	-0.0680462950645470	0.00095243781440
31	(0.4082264429601952, 0.4083840617701118, 0.4081585893818691)	-0.068043226035052	-0.0680454134814434	0.00071432836080

32	(0.408223103042044, 0.4083833070143146, 0.4081552750718499)	-0.0680427741223619	-0.0680441784748097	0.0005357462705

Pabandome q dar labiau sumažinti ir keičiame į 0.05. Gauname, jog prireikia 8 iteracijų ir 872 funkcijų skaičiavimų.

7 lentelė. Gauti rezultatai po pirmų ir paskutinių trijų iteracijų iš pradinio taško X_m, r =4, q = 0.05.

Iteracija	X	B(X, r)	f(X)	r
1	(0.4955309255077497, 0.36946672015547993, 0.46164950808765276)	-0.0777287351671542	-0.0845198010063146	4
2	(0.6264268652746205, 0.3304330697057323, 0.312234789553447)	-0.0639672641513194	-0.0646301510310598	0.2
3	(0.5193505212461842, 0.32886041809543143, 0.3879714665996278)	0.06625692961739885	-0.0662631325386591	0.01
6	(0.5037385483383519, 0.3516680985964369, 0.3774221895283356)	-0.0668498213926519	-0.0668598794702329	0.00000125
7	(0.504966266345466, 0.3523649345414769, 0.375662974754564)	-0.0668426148887896	-0.0668426167127178	0.0000000625
8	(0.5049176944041807, 0.35235340167370377, 0.3757160691321094)	-0.0668433930399080	-0.0668434456734725	0.000000003125

Matome, jog nukentėjo metodo tikslumas, bandom q padidinti ir keičiam į q=0.15. Gauname, jog prireikė 10 iteracijų ir 877 funkcijų skaičiaivimų.

8 lentelė. Gauti rezultatai po pirmų ir paskutinių trijų iteracijų iš pradinio taško X_m, r =4, q = 0.15.

Iteracija	X	B(X, r)	f(X)	r
1	(0.4955309255077497, 0.36946672015547993, 0.46164950808765276)	-0.0777287351671542	-0.0845198010063146	4
2	(0.4110812024920635, 0.41610810870441894, 0.41712050297337266)	-0.0696239068639467	-0.0713502229882657	0.6
3	(0.41353101163588946, 0.40570056486461337, 0.4079625313930827)	-0.0682685593691690	-0.0684437780245861	0.09
8	(0.4083011651096837, 0.40818687821869815, 0.40825709051172704)	-0.0680413980803392	-0.0680414241290526	0.0000068
9	(0.4081201638668812,	-0.0680413773622598	-0.0680413774955461	0.000001025

	0.40826083856887774, 0.40836388012891367)			
10	(0.4081201638668812, 0.40826083856887774, 0.40836388012891367)	-0.0680413766069707	-0.0680413774955461	0.0000001537

Toliau bandome nagrinėti r. iš pradžių didiname, keičiame į r=8. Prireikė 9 iteracijų ir 790 funkcijų skaičiavimų.

9 lentelė. Gauti rezultatai po pirmų ir paskutinių trijų iteracijų iš pradinio taško X_m, r = 8, q = 0.15.

Iteracija	X	B(X, r)	f(X)	r
1	(0.5016288389807951, 0.4912949101068922, 0.5020380609778876)	-0.0937299846615048	-0.1237261231080387	8
2	(0.42124779967246206, 0.42076370746298486, 0.42078387861139943)	-0.0712640234379045	-0.0745821692799076	1.2
3	(0.4119501767319784, 0.4051754886493776, 0.4129501653214429)	-0.0685069878101966	-0.0689263851351099	0.18
7	(0.4084709110283293, 0.40747077544594784, 0.4088065497612886)	-0.0680415202263474	-0.0680417453255315	0.0000911
8	(0.40862792765440475, 0.4078849247773568, 0.4082326017851651)	-0.0680413890953874	-0.0680414224628013	0.00001366
9	(0.40862792765440475, 0.4078849247773568, 0.4082326017851651)	-0.0680412000133755	-0.0680414224628013	0.00000205

Bandome r dar labiau padidinti ir keičiame į 16. Gauname, jog prireikė 9 iteracijų ir 868 funkcijų skaičiavimų.

10 lentelė. Gauti rezultatai po pirmų ir paskutinių trijų iteracijų iš pradinio taško X_m, r = 16, q = 0.15.

Iteracija	X	B(X, r)	f(X)	r
1	(0.6076834873822281, 0.6074019053742244, 0.6082441914957679)	-0.132038125890349	-0.2245078627860717	16
2	(0.4460789171880777, 0.4778623374606544, 0.39090735576502367)	-0.0741168350961834	-0.0833274983524299	2.4
3	(0.41200120935828405, 0.413969485342336, 0.41015919664881884)	-0.0689867516564723	-0.069955082657641	0.36
7	(0.40943239422588307, 0.40683949277694154,	-0.068041504423284	-0.068042005496387	0.0001822499

	0.4084809434279336)			
8	(0.40826554418595074, 0.40737435319975024, 0.40910666793251516)	-0.0680412991242766	-0.0680413576877117	0.0000273374
9	(0.40829852373026254, 0.40742160922880355, 0.40902543975533445)	-0.0680412302008451	-0.0680412353020085	0.0000041006

Toliau pabandome r sumažinti, keičiame į 1. Gauname, jog reikia 9 iteracijų ir 784 funkcijų skaičiavimų.

11 lentelė. Gauti rezultatai po pirmų ir paskutinių trijų iteracijų iš pradinio taško X_m, r = 1, q = 0.15.

Iteracija	X	B(X, r)	f(X)	r
1	(0.4173502077023937, 0.4231585688620354, 0.41521753865842514)	-0.0707083278169758	-0.0733296248749766	1
2	(0.41073116433673185, 0.40834987896746167, 0.4103755580022863)	-0.068433163623669	-0.0688290180577165	0.15
3	(0.4081785719995832, 0.4065393670967711, 0.41071687673549656)	-0.0680990636752861	-0.068154628909926	0.0225
7	(0.40825144748639886, 0.40823461735103833, 0.4082592617860916)	-0.06804140907643	-0.0680414575512422	0.000001139
8	(0.40846477331031866, 0.4083714451461333, 0.4079085579283239)	-0.0680412649354151	-0.068041329678603	0.00000170859
9	(0.40846477331031866, 0.4083714451461333, 0.4079085579283239)	-0.068040898057346	-0.0680413296786037	0.0000002562

Bandome r dar labiau sumažinti ir keičiam į 0.4. Gauname, jog prireikė tik 6 iteracijų ir 625 funkcijų skaičiavimų.

12 lentelė. Gauti rezultatai iš pradinio taško X_m, r =0.4, q = 0.15.

Iteracija	X	B(X, r)	f(X)	r
1	(0.4287619950030064, 0.4363549495571414, 0.37650612494837365)	-0.0687993922787167	-0.07044144157254	0.4
2	(0.4066964939356286, 0.4157642524231994, 0.40420013331649723)	-0.0681903695002631	-0.0683461454759915	0.06
3	(0.4089132278622798, 0.4084713287814702,	-0.0680647385057294	-0.068087912165380	0.009

	0.40764045663354814)			
4	(0.40823133725884714, 0.40914789400174845, 0.40740782801913655)	-0.06804474114545	-0.06804810399257	0.001349
5	(0.408233520355444, 0.40828211403400383, 0.40823592456568725)	-0.068041907244829	-0.068042496037767	0.0002024
6	(0.4082417675598088, 0.408269555417893, 0.4082345804953098)	-0.0680414601792334	-0.06804155361840	0.00003037

Bandome r dar labiau sumažinti ir keičiam į 0.1. Gauname, jog prireikė tik 5 iteracijų ir 621 funkcijų skaičiavimų.

13 lentelė. Gauti rezultatai iš pradinio taško X_m, r =0.1, q = 0.15.

Iteracija	X	B(X, r)	f(X)	r
1	(0.32700524555240706, 0.5403243529629471, 0.37659320328955825)	-0.06609912003604	-0.066539837977546	0.1
2	(0.3992393408552677, 0.4172806111921611, 0.40828534592643084)	-0.0680179566867625	-0.068018230312792	0.015
3	(0.40174185800927775, 0.4104875938922019, 0.41262595210537156)	-0.068040389925426	-0.0680461658400802	0.00225
4	(0.40208535226257125, 0.4108879751758847, 0.41181815403363675)	-0.0680363101983511	-0.0680373157881039	0.0003374
5	(0.40209451940343566, 0.4108591121560227, 0.4118299449393297)	-0.068036035455112	-0.0680353952219591	0.0000506249

Tačiau iš rezultatų matome, jog sumažėjo metodo tikslumas.

Rezultatų palyginimai

14 lentelė. Rrezultatų palyginimai iš taškų X_0, X_1, X_m su r=4, q=0.5

	X_0	X_1	X_m
Iteracijų skaičius	15	15	15
Minimumo X_{min} reikšmė	(0.4082663373112877, 0.40816534564300444, 0.40832035160957214)	(0.4082663373112877, 0.40816534564300444, 0.40832035160957214)	(0.40825951383821235, 0.40833434471028074, 0.4081587537098548)

Funkcijos $f(X_{min})$ reikšmė	-0.0680425730855883	-0.0680425730855883	-0.06804266872683354
Skaičiuotų funkcijų kiekis	1320	1320	1289

Iš lentelės matosi, jog iš visų taškų gauti rezultatai labai panašūs, iš X_m, taško prireikė nežymiai mažiau funkcijų skaičiavimų.

15 lentelė. Rezultatų palyginimai iš taško X_m, r=4.

	q=0.5	q=0.25	q=0.75
Iteracijų skaičius	15	12	32
Minimumo X_{min} reikšmė	(0.40825951383821235, 0.40833434471028074, 0.4081587537098548)	(0.40811441514765046, 0.4083961431183335, 0.40823448333930723)	(0.408223103042044, 0.4083833070143146, 0.4081552750718499)
Funkcijos $f(X_{min})$ reikšmė	-0.06804266872683354	-0.06804140195362074	-0.0680441784748097
Skaičiuotų funkcijų kiekis	1289	1054	2450
	q=0.05	q=0.15	
Iteracijų skaičius	8	10	
Minimumo X_{min} reikšmė	(0.5049176944041807, 0.35235340167370377, 0.3757160691321094)	(0.4081201638668812, 0.40826083856887774, 0.40836388012891367)	
Funkcijos $f(X_{min})$ reikšmė	-0.0668434456734725	-0.0680413774955461	
Skaičiuotų funkcijų kiekis	872	877	

Iš 15 lentelės matome, jog nuo tinkamai pasirinkto q labai priklauso iteracijų skaičius ir skaičiuotų funkcijų kiekis. Kuo q mažesnis, tuo mažiau iteracijų ir funkcijų skaičiavimų prireiks, tačiau, jei q pasirinksime per maža, pradeda kentėti metodo tikslumas, pvz.: su q=0.05 funkcijos riekšmė minimumo taške yra gana netiksli lyginant su kitais gautais rezultatais.

16 lentelė. Rezultatų palyginimai iš taško X_m, q=0.15.

	r=4	r=8	r=16
Iteracijų skaičius	10	9	9
Minimumo X_{min} reikšmė	(0.4081201638668812, 0.40826083856887774, 0.40836388012891367)	(0.40862792765440475, 0.4078849247773568, 0.4082326017851651)	(0.40829852373026254, 0.40742160922880355, 0.40902543975533445)
Funkcijos $f(X_{min})$ reikšmė	-0.0680413774955461	-0.0680414224628013	-0.0680412353020085

Skaičiuotų funkcijų kiekis	877	790	868
	r=1	r=0.4	r=0.1
Iteracijų skaičius	9	6	5
Minimumo X_{min} reikšmė	(0.40846477331031866, 0.4083714451461333, 0.4079085579283239)	(0.4082417675598088, 0.408269555417893, 0.4082345804953098)	(0.40209451940343566, 0.4108591121560227, 0.4118299449393297)
Funkcijos $f(X_{min})$ reikšmė	-0.0680413296786037	-0.06804155361840	-0.0680353952219591
Skaičiuotų funkcijų kiekis	784	625	621

Iš 16 lentelės matome, jog nuo tinkamai pasirinkto r priklauso iteracijų skaičius ir skaičiuotų funkcijų kiekis. Kuo r mažesnis, tuo mažiau iteracijų ir funkcijų skaičiavimų prireiks, tačiau, jei r pasirinksime per maža, pradeda kentėti metodo tikslumas, pvz.: su r=0.1 funkcijos reikšmė minimumo taške yra gana netiksli lyginant su kitais gautais rezultatais.

Lyginant 15 ir 16 lentelės matosi, jog r ir q veikia panašiai – kuo mažesnis, tuo mažiau iteracijų ir funkcijų skaičiavimų, o pasirenkant per mažus – nukenčia tikslumas. Tačiau taip pat iš lentelių matosi, jog q įtaka metodui yra didesnė nei r.

Išvados

Apibendrinant galima būtų teigti, jog iš visų taškų metodas tinkamai atliko savo darbą – surado funkcijos minimumą, o iteracijų skaičiui ir skaičiuotų funkcijų kiekiui įtaką turi r ir q parametrai: kuo mažesnis r, tuo didesnę įtaka funkcija b(x) turi funkcijai B(X, r), kas matosi ir iš formulės $B(X, r) = f(X) + 1/r * b(X)$, o kuo r staigiau mažėja, t.y. kuo q mažesnis, tuo kiekvienoj iteracijoj funkcija b(X) turi vis didesnę įtaką funkcijai B(X, r), dėl ko greičiau atrandamas minimumas. Tačiau pradedant su per mažu r arba q, funkcija b(X) turi per didelę įtaka funkcijai B(X, r), dėl ko tikslo funkcijos f(X) įtaka tampa per maža ir sumažėja metodo tikslumas. Naudojant tuos pačius parametrus q ir r iš skirtingų pradžios taškų metodui prireikė vienodai iteracijų bei panašiai funkcijų skaičiavimų, o gauti X_{min} ir $f(X_{min})$ labai panašūs. Taip pat buvo naudotas deformuojamo simplekso metodas baudos funkcijos minimizavimui, kurio parametrų šiame laboratoriniame darbe nekeičiau ir nenagrinėjau, tačiau iš praeito laboratorinio darbo rezultatų žinoma, jog metodo parametrai taip pat turi didelę įtaką minimumui rasti, juos pakoregavus veikiausiai būtų galima minimumą rasti dar greičiau.

Priedas

```
from operator import itemgetter
import math
import numpy as np


def getPoint(arg, value):
    return {"arg": arg, "value": value}


def getModVector(x):
    return math.sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2])


def simplex_method(func, args, epsilon=0.001, alpha=0.5, gama=3, beta=0.5,
niu=0.5):
    simplex = [getPoint(args, func(args))]
    max_iterations = 1000
    counter = 1

    for i in range(0, len(args)):
        argList = list(args)
        argList[i] += alpha
        simplex.append(getPoint(argList, func(argList)))
        counter += 1

    for i in range(0, max_iterations):
        # 1. Sort
        simplex.sort(key=itemgetter('value'))

        # 6. Check convergence
        if getModVector(np.array((simplex[0]['arg']) - np.array(simplex[-
1]['arg']))) < epsilon:
            break

        centroid = [0] * len(args)
        for j in range(0, len(args)):
            for k in range(0, len(simplex) - 1):
                centroid[j] += simplex[k]['arg'][j]
            centroid[j] /= (len(simplex) - 1)

        # 2. Reflect
        reflection = [0] * len(args)
        for j in range(0, len(args)):
            reflection[j] = centroid[j] + alpha * (centroid[j] - simplex[-
1]['arg'][j])
        reflection_value = func(reflection)
        counter += 1

        # 3. Evaluate or Extend
        if simplex[0]['value'] <= reflection_value < simplex[-2]['value']:
            simplex[-1] = getPoint(reflection, reflection_value)
            continue
        elif reflection_value < simplex[0]['value']:
            extend = [0] * len(args)
            for j in range(0, len(args)):
                extend[j] = centroid[j] + gama * (reflection[j] - centroid[j])
            extended_value = func(extend)
            counter += 1
            if extended_value < simplex[0]['value']:
```

```
                simplex[-1] = getPoint(extend, extended_value)
            else:
                simplex[-1] = getPoint(reflection, reflection_value)
            continue

        # 4. Contract
        contraction = [0] * len(args)
        for j in range(0, len(args)):
            contraction[j] = centroid[j] + niu * (simplex[-1]['arg'][j] -
centroid[j])
        contraction_value = func(contraction)
        counter += 1
        if contraction_value < simplex[-1]['value']:
            simplex[-1] = getPoint(contraction, contraction_value)
            continue

        # 5. Reduce
        for j in range(1, len(simplex)):
            reduce = [0] * len(args)
            for k in range(0, len(args)):
                reduce[k] = simplex[0]['arg'][k] + beta * (simplex[j]['arg'][k]
- simplex[0]['arg'][k])
            reduce_value = func(reduce)
            counter += 1
            simplex[j] = getPoint(reduce, reduce_value)

    return simplex[0]['arg'], counter


def optimization(func, constraints, args, epsilon, penaltyQuantifier=1,
rdiv=0.5):
    penaltyQuantifier = penaltyQuantifier / rdiv
    equals = []
    inequals = []

    for con in constraints:
        if con.get("type") == "eq":
            equals.append(con.get("func"))
        elif con.get("type") == "ineq":
            inequals.append(con.get("func"))

    penaltyFunction = lambda x: sum(eq(x) ** 2 for eq in equals) + sum(
        max(0, iq(x)) ** 2 for iq in inequals)

    r = lambda x: x * rdiv

    bfunc = lambda x: func(x) + 1 / r(penaltyQuantifier) * penaltyFunction(x)

    counter = 0
    maxIterations = 100
    for i in range(1, maxIterations):
        newargs, c = simplex_method(bfunc, args, epsilon)
        counter += c

        print(i)
        print("r: ", r(penaltyQuantifier))
        print("Baudos funkcija:", bfunc(newargs))
        print("X:", "(" + str(newargs[0]) + ", " + str(newargs[1]) + ", " +
str(newargs[2]) + ")")
        print("F(X): ", func(newargs))
        print('*************')
```

```
        if getModVector(np.array(newargs) - np.array(args)) < epsilon:
            break
        else:
            args = newargs
            penaltyQuantifier = r(penaltyQuantifier)

    print("X:", newargs, ". f(X):", func(newargs), ". iterations:", i)
    print("counter: ", counter)


def main():
    # funkcija yra Turis, gi yra (pavirsiaus plotas - 1)
    func = lambda x: -1 * x[0] * x[1] * x[2]

    g1 = lambda x: 2 * (x[0] * x[1] + x[0] * x[2] + x[1] * x[2]) - 1

    h1 = lambda x: -x[0]
    h2 = lambda x: -x[1]
    h3 = lambda x: -x[2]

    constraints = (
        {"type": "eq", "func": g1},
        {"type": "ineq", "func": h1},
        {"type": "ineq", "func": h2},
        {"type": "ineq", "func": h3})

    # print("Funkcija taske 0,0,0:", func([0, 0, 0]))
    # print("Funkcija taske 1,1,1:", func([1, 1, 1]))
    # print("Funkcija taske 0,0.3,0.9:", func([0, 0.3, 0.9]))
    #
    # print("g(X) taske 0,0,0:", g1([0, 0, 0]))
    # print("g(X) taske 1, 1, 1:", g1([1, 1, 1]))
    # print("g(X) taske 0,0.3,0.9:", g1([0, 0.3, 0.9]))
    #
    # print("h1(X) taske 0,0,0:", h1([0, 0, 0]))
    # print("h2(X) taske 0,0,0:", h2([0, 0, 0]))
    # print("h3(X) taske 0,0,0:", h3([0, 0, 0]))
    #
    # print("h1(X) taske 1,1,1:", h1([1, 1, 1]))
    # print("h2(X) taske 1,1,1:", h2([1, 1, 1]))
    # print("h3(X) taske 1,1,1:", h3([1, 1, 1]))
    #
    # print("h1(X) taske 0,0.3,0.9:", h1([0, 0.3, 0.9]))
    # print("h2(X) taske 0,0.3,0.9:", h2([0, 0.3, 0.9]))
    # print("h3(X) taske 0,0.3,0.9:", h3([0, 0.3, 0.9]))

    # optimization(func, constraints, [0, 0, 0], 0.0001, penaltyQuantifier=4,
rdiv=0.5)
    # optimization(func, constraints, [1, 1, 1], 0.0001, penaltyQuantifier=4,
rdiv=0.5)
    optimization(func, constraints, [0, 0.3, 0.9], 0.0001, penaltyQuantifier=4,
rdiv=0.5)

if __name__ == "__main__":
    main()
```