VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

Optimizavimas be apribojimų

Laboratorinio darbo ataskaita

Atliko: Monika Mirbakaitė

VU el. p.: monika.mirbakaite@mif.stud.vu.lt,

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

Vilnius

2024

TURINYS

ĮVADAS

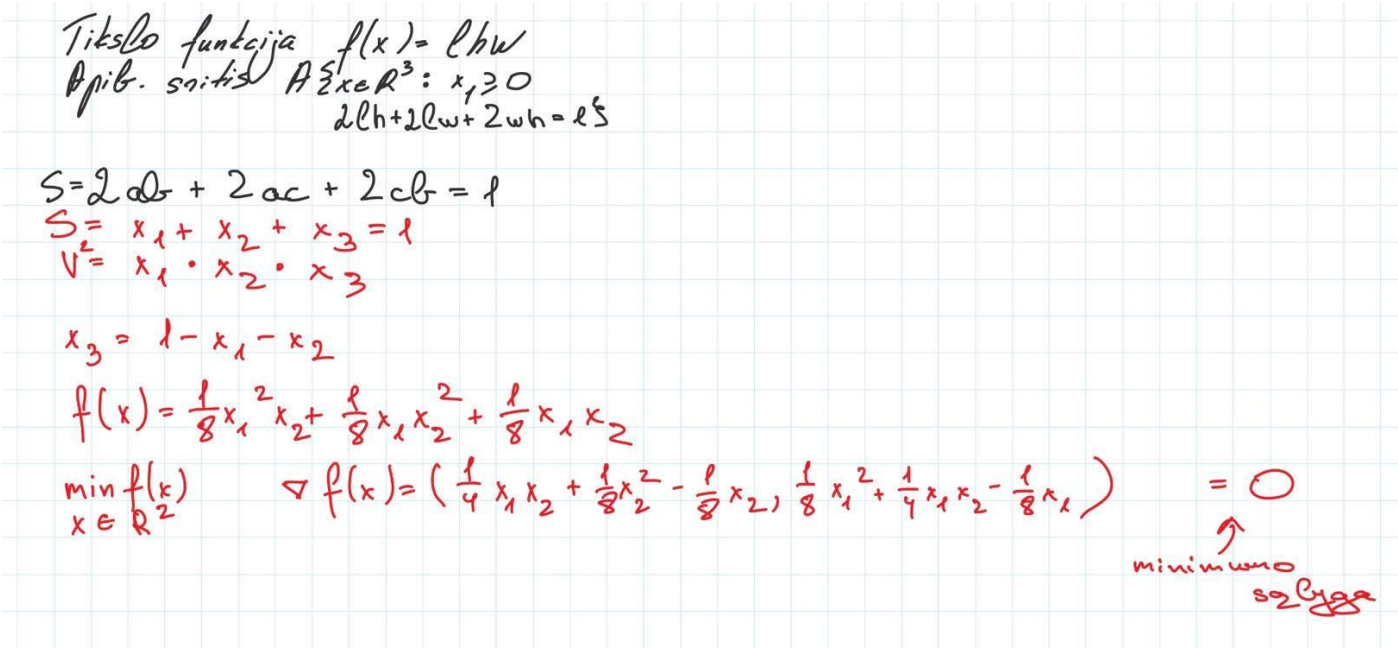
Šiame laboratoriniame darbe bandoma atsakyti į klausimą „Kokia turėtų būti stačiakampio gretasienio formos dėžė, kad vienetiniam paviršiaus plotui jos tūris būtų maksimalus?“

Kad tai atlikti:

- programuojami gradientinio nusileidimo, greičiausiojo nusileidimo ir deformuojamo simplekso algoritmai
- laikant kintamaisiais dėžės priekinės ir galinės sienų plotų sumą, šoninių sienų plotų sumą, viršutinės ir apatinės sienų plotų sumą, aprašomi vienetinio dėžės paviršiaus ploto reikalavimas ir dėžės tūrio pakelto kvadratu funkcija
- iš vienetinio paviršiaus ploto reikalavimo išvedami vieno iš kintamojo išraišką per kitus.
- aprašoma tikslo funkcija f(X) taip, kad optimizavimo uždavinys būtų formuluojamas be apribojimų: min f(X)
- išvedama ir aprašoma tikslo funkcijos gradiento funkcija
- apskaičiuojama tikslo ir gradiento funkcijų reikšmes taškuose X_0= (0,0), X_1=(1,1) ir X_m= (a/10, b/10), čia a ir b – studento knygelės numerio “1x1xxab” skaitmenys
- minimizuojamas suformuluotas uždavinys naudojant suprogramuotus optimizavimo algoritmus pradedant iš taškų X_0, X_1 ir X_m
- palyginami rezultatus: gauti sprendiniai, rastas funkcijos minimumo įvertis, atliktų žingsnių ir funkcijų skaičiavimų skaičius priklausomai nuo pradinio taško
- vizualizuojama tikslo funkciją ir bandymo taškai

SKAIČIAVIMAI

1 pav. pateikiami reikalingi skaičiavimai.



1 pav. Skaičiavimai.

REZULTATŲ PALYGINIMAS

1.1. Rezultatų lentelė

1 lentelė. vaizduojami rezultatai įvairiais metodais, kuri buvo sugeneruota naudojant MATLAB. Methods – pasirinktas metodas, starting_point – pradžios taškai, x_values – x reikšmės, y_values – y reikšmės, iter_count – iteracijų skaičius, fja-count – tikslo funkcijos kvietimų skaičius, func_values – funkcijos reikšmė.

1 lentelė. Rezultatai.

Methods	starting_point	x_values	y_values	iter_count	fja_count	func_values
Gradiento nusileidimo algoritmas	(0, 0)	0	0	1	1	0
Gradiento nusileidimo algoritmas	(1, 1)	0,333192412	0,333192412	11	11	-0,004629627
Gradiento nusileidimo algoritmas	(8/10, 4/10)	0,333993159	0,332902539	7	7	-0,004629616
Greiciausio nusileidimo algoritmas	(0, 0)	0	0	1	1	0
Greiciausio nusileidimo algoritmas	(1, 1)	0,334021919	0,334021919	42	42	-0,00462957
Greiciausio nusileidimo algoritmas	(8/10, 4/10)	0,33555512	0,331139617	97	97	-0,004629427
Deformuojamo simplekso algoritmas	(0, 0)	0,333439878	0,33314493	29	136	-0,004629629
Deformuojamo simplekso algoritmas	(1, 1)	0,333211919	0,333391903	38	173	-0,004629629
Deformuojamo simplekso algoritmas	(8/10, 4/10)	0,333736293	0,333906648	30	134	-0,0046296

1.2. Rezultatai

2 pav. (gradientinio nusileidimo metodas), 3 pav. (greičiausio nusileidimo metodas), 4 pav. (deformuojamo simplekso metodas).

1.2.1. Rezultatai naudojant gradientinio nusileidimo metodą

Methods	starting_point	x_values	y_values	iter_count	fja_count	func_values
"Gradiento nusileidimo algoritmas"	"(0, 0)"	0	0	1	1	0
"Gradiento nusileidimo algoritmas"	"(1, 1)"	0.33319	0.33319	11	11	-0.0046296
"Gradiento nusileidimo algoritmas"	"(8/10, 4/10)"	0.33336	0.33333	11	11	-0.0046296
"Greiciausio nusileidimo algoritmas"	"(0, 0)"	0	0	1	1	0
"Greiciausio nusileidimo algoritmas"	"(1, 1)"	0.33402	0.33402	42	42	-0.0046296
"Greiciausio nusileidimo algoritmas"	"(8/10, 4/10)"	0.33853	0.32828	100	100	-0.0046285
"Deformuojamo simplekso algoritmas"	"(0, 0)"	0.33344	0.33314	29	136	-0.0046296
"Deformuojamo simplekso algoritmas"	"(1, 1)"	0.33321	0.33339	38	173	-0.0046296
"Deformuojamo simplekso algoritmas"	"(8/10, 4/10)"	0.33374	0.33391	30	134	-0.0046296

2 pav. rezultatai. Gradientinis nusileidimas.

1.2.2. Rezultatai naudojant greičiausio nusileidimo metodą

Methods	starting_point	x_values	y_values	iter_count	fja_count	func_values
"Gradiento nusileidimo algoritmas"	"(0, 0)"	0	0	1	1	0
"Gradiento nusileidimo algoritmas"	"(1, 1)"	0.33319	0.33319	11	11	-0.0046296
"Gradiento nusileidimo algoritmas"	"(8/10, 4/10)"	0.33336	0.33333	11	11	-0.0046296
"Greiciausio nusileidimo algoritmas"	"(0, 0)"	0	0	1	1	0
"Greiciausio nusileidimo algoritmas"	"(1, 1)"	0.33402	0.33402	42	42	-0.0046296
"Greiciausio nusileidimo algoritmas"	"(8/10, 4/10)"	0.33853	0.32828	100	100	-0.0046285
"Deformuojamo simplekso algoritmas"	"(0, 0)"	0.33344	0.33314	29	136	-0.0046296
"Deformuojamo simplekso algoritmas"	"(1, 1)"	0.33321	0.33339	38	173	-0.0046296
"Deformuojamo simplekso algoritmas"	"(8/10, 4/10)"	0.33374	0.33391	30	134	-0.0046296

3 pav. rezultatai. Greičiausias nusileidimas.

1.2.3. Rezultatai naudojant deformuoto simplekso metodą

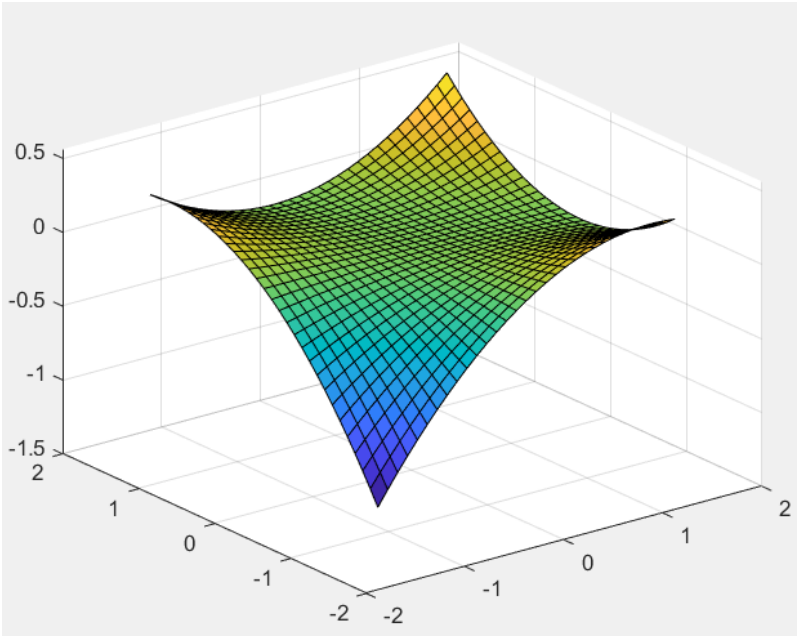
Methods	starting_point	x_values	y_values	iter_count	fja_count	func_values
"Gradiento nusileidimo algoritmas"	"(0, 0)"	0	0	1	1	0
"Gradiento nusileidimo algoritmas"	"(1, 1)"	0.33319	0.33319	11	11	-0.0046296
"Gradiento nusileidimo algoritmas"	"(8/10, 4/10)"	0.33336	0.33333	11	11	-0.0046296
"Greiciausio nusileidimo algoritmas"	"(0, 0)"	0	0	1	1	0
"Greiciausio nusileidimo algoritmas"	"(1, 1)"	0.33402	0.33402	42	42	-0.0046296
"Greiciausio nusileidimo algoritmas"	"(8/10, 4/10)"	0.33853	0.32828	100	100	-0.0046285
"Deformuojamo simplekso algoritmas"	"(0, 0)"	0.33344	0.33314	29	136	-0.0046296
"Deformuojamo simplekso algoritmas"	"(1, 1)"	0.33321	0.33339	38	173	-0.0046296
"Deformuojamo simplekso algoritmas"	"(8/10, 4/10)"	0.33374	0.33391	30	134	-0.0046296

4 pav. rezultatai. Deformuojamas simpleksas.

VIZUALIZUOTOS TIKSLO FUNKCIJOS IR JŲ BANDYMO TAŠKAI

5 pav. vaizduojama tikslo funkcija. 6 pav. (gradientinis nusileidimas), 9 pav. (greičiausias nusileidimas) vaizduojama tikslo funkcijos ir jos bandymo taškų vizualizacijos x_0 taške (0,0). 7 pav. (gradientinis nusileidimas), 10 pav. (greičiausias nusileidimas) vaizduojama tikslo funkcijos ir jos bandymo taškų vizualizacijos x_1 taške (1,1). 8 pav. (gradientinis nusileidimas), 11 pav. (greičiausias nusileidimas) vaizduojama tikslo funkcijos ir jos bandymo taškų vizualizacijos x_m taške (8/10,4/10).

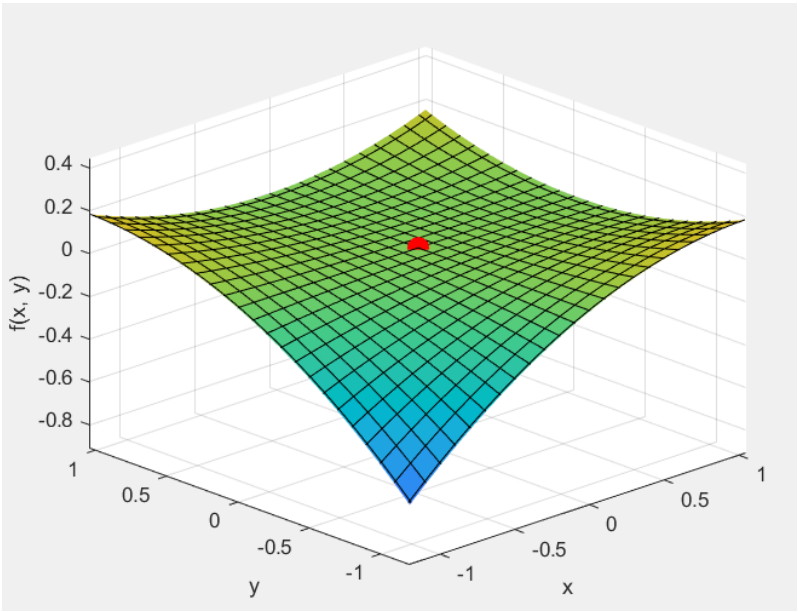
Raudona spalva žymimi bandymo taškai.



5 pav. grafikas. Tikslo funkcija.

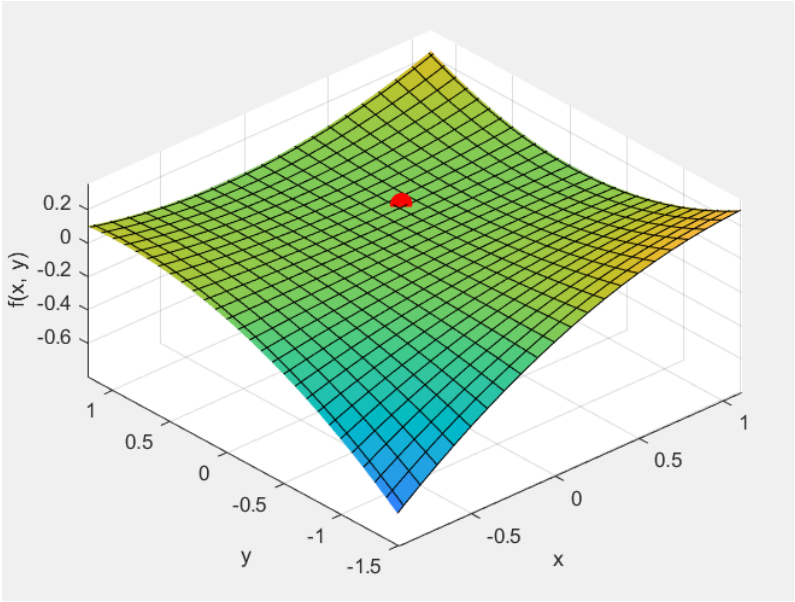
1.3. Gradientinio nusileidimo algoritmo tikslo funkcijos ir jos bandymo taškų vizualizacija

1.3.1. Gradientinio nusileidimo algoritmo grafikas (x_0 taške (0,0))



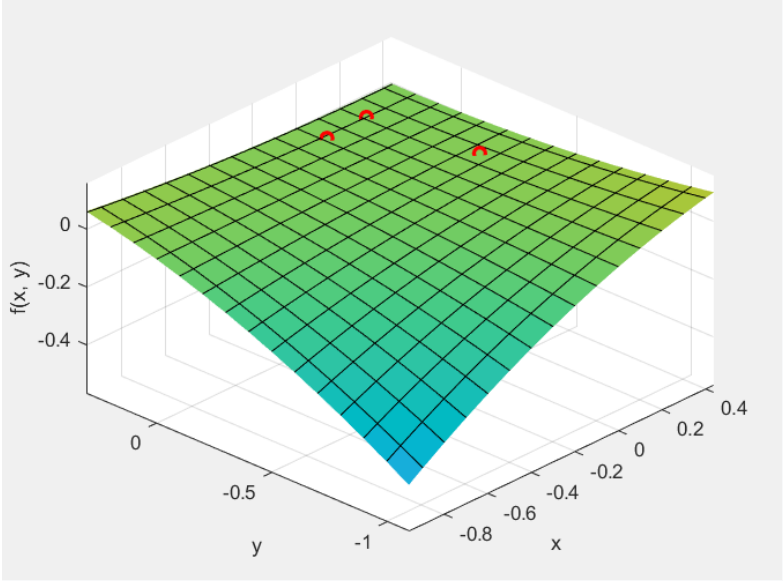
6 pav. grafikas. Gradientinis nusileidimas (1).

1.3.2. Gradientinio nusileidimo algoritmo grafikas (x_1 taške (1,1))



7 pav. grafikas. Gradientinis nusileidimas (2).

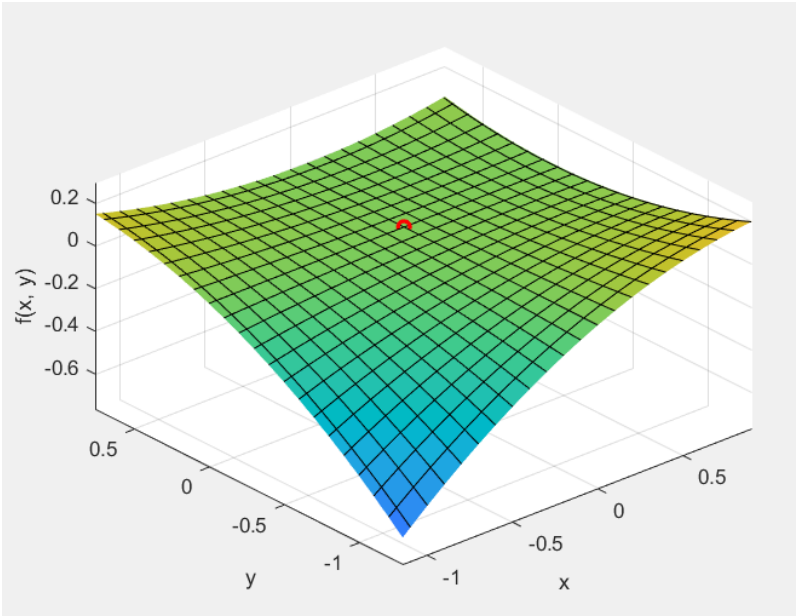
1.3.3. Gradientinio nusileidimo algoritmo grafikas (x_m taške (8/10,4/10))



8 pav. grafikas. Gradientinis nusileidimas (3).

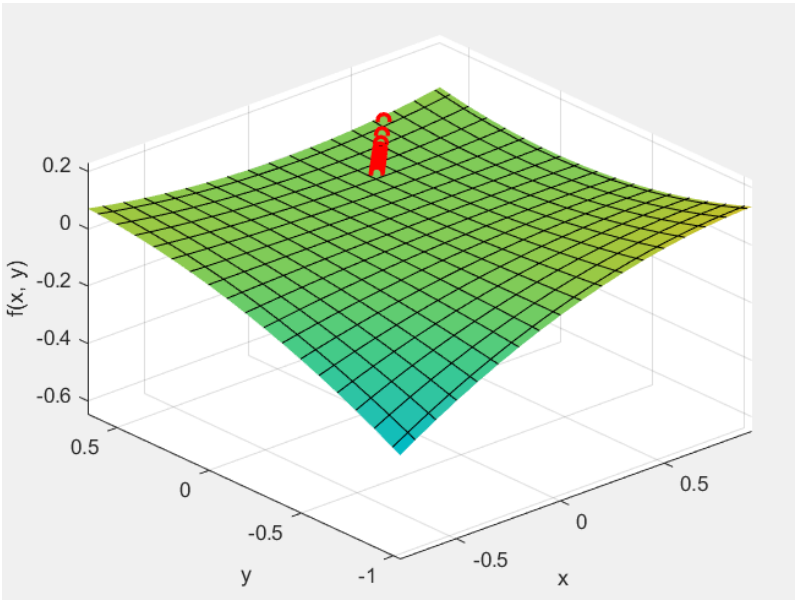
1.4. Greičiausio algoritmo tikslo funkcijos ir jos bandymo taškų vizualizacija

1.4.1. Greičiausio nusileidimo algoritmo grafikas (x_0 taške (0,0))



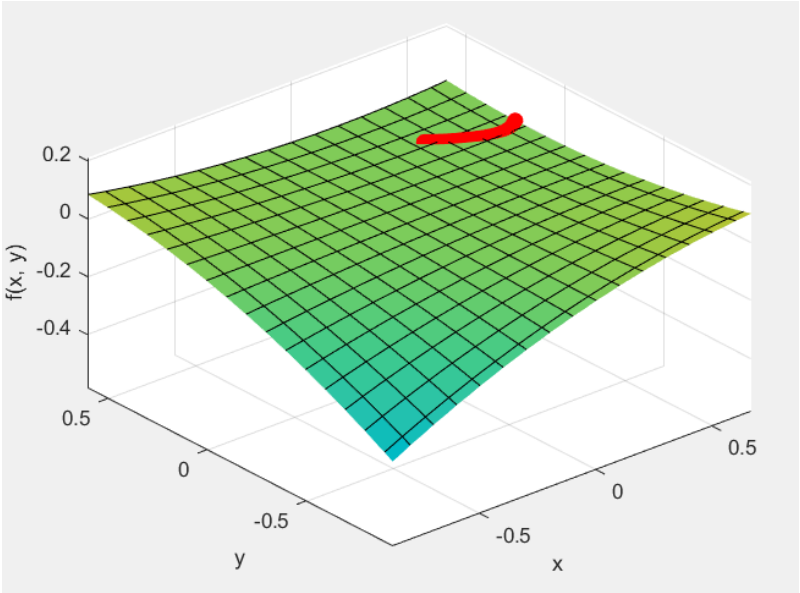
9 pav. grafikas. Greičiausias nusileidimas (1).

1.4.2. Greičiausio nusileidimo algoritmo grafikas (x_1 taške (1,1))



10 pav. grafikas. Greičiausias nusileidimas (2).

1.4.3. Greičiausio nusileidimo algoritmo grafikas (x_m taške (8/10,4/10))



11 pav. grafikas. Greičiausias nusileidimas (3).

1.5. Deformuojamo simplekso algoritmo tikslo funkcijos ir jos bandymo taškų vizualizacija

Deja, šio metodo grafiškai pavaizduoti nepavyko, nes naudotas grafinio atvaizdavimo metodas tiko tik gradientinio bei greičiausio nusileidimo metodams.

IŠVADOS

Realizavau minėtus minimizavimo algoritmus ir gavau šias išvadas:

1. Palyginus visus 3 algoritmus, matome, kad gradietinio nusileidimo metodas rado sprendimą per mažiausiai žingsnių, tikslo funkcijos kvietimų.
2. Palyginus visus 3 algoritmus, matome, kad deformuojamo simplekso metodo iteracijų skaičius ir tikslo funkcijos kvietimų skaičius skiriasi labiausiai.

PRIEDAI

Toliau pateikiamas kodas, kuriuo buvo gauti aukščiau esantys rezultatai.

1.6. Pagrindinio kodo metodo realizacija

```
%tikslo f-ja
f = @(x, y) (1/8.*x.^2.*y+1/8.*x.*y.^2-1/8.*x.*y);
%gradiento formule
g = @(x,y) [1/4.*x.*y+1/8.*y.^2-1/8.*y; 1/8.*x.^2+1/4.*x.*y-1/8.*x];
%Gradiento nusileidimo algoritmas (0,0)
[x_0_0, y_0_0, iter_0, g_sk_0, L_0, X_0, Y_0, r_0] = gradientinis (f, g, 0, 0, 3, 0.0001, 50);
%Gradiento nusileidimo algoritmas (1,1)
[x_0_1, y_0_1, iter_1, g_sk_1, L_1, X_1, Y_1, r_1] = gradientinis (f, g, 1, 1, 2.7, 0.0001, 50);
%Gradiento nusileidimo algoritmas (8/10,4/10)
[x_0_2, y_0_2, iter_2, g_sk_2, L_2, X_2, Y_2, r_2] = gradientinis (f, g, 8/10, 4/10, 10.8,
0.0001, 50);
%Greiciausio nusileidimo algoritmas (0,0)
[x_0_3, y_0_3, iter_3, g_sk_3, L_3, X_3, Y_3, r_3] = greiciausias (g, 0, 0, 0.0001, 100, f);
%Greiciausio nusileidimo algoritmas (1,1)
[x_0_4, y_0_4, iter_4, g_sk_4, L_4, X_4, Y_4, r_4] = greiciausias (g, 1, 1, 0.0001, 100, f);
%Greiciausio nusileidimo algoritmas (8/10,4/10)
[x_0_5, y_0_5, iter_5, g_sk_5, L_5, X_5, Y_5, r_5] = greiciausias(g, 8/10, 4/10, 0.0001, 100,
f);
% Deformuojamo simplekso algoritmas (0,0)
[triangle_1, iter_6, g_sk_6, X_6, Y_6, r_6, x_0_6, y_0_6] =  simpleksas(f, 0, 0, 100, 0.7);
% Deformuojamo simplekso algoritmas (1,1)
[triangle_2, iter_7, g_sk_7, X_7, Y_7, r_7, x_0_7, y_0_7] =  simpleksas(f, 1, 1, 100, 0.5);
% Deformuojamo simplekso algoritmas (8/10,4/10)
[triangle_3, iter_8, g_sk_8, X_8, Y_8, r_8, x_0_8, y_0_8] =  simpleksas(f, 8/10, 4/10, 100,
0.8);

Methods = ["Gradiento nusileidimo algoritmas"; "Gradiento nusileidimo algoritmas"; "Gradiento
nusileidimo algoritmas"; "Greiciausio nusileidimo algoritmas"; "Greiciausio nusileidimo
algoritmas"; "Greiciausio nusileidimo algoritmas"; "Deformuojamo simplekso algoritmas";
"Deformuojamo simplekso algoritmas"; "Deformuojamo simplekso algoritmas"];
x_values = [x_0_0; x_0_1; x_0_2; x_0_3; x_0_4; x_0_5; x_0_6; x_0_7; x_0_8];
y_values = [y_0_0; y_0_1; y_0_2; y_0_3; y_0_4; y_0_5; y_0_6; y_0_7; y_0_8];
iter_count = [iter_0; iter_1; iter_2; iter_3; iter_4; iter_5; iter_6; iter_7; iter_8];
fja_count = [g_sk_0; g_sk_1; g_sk_2; g_sk_3; g_sk_4; g_sk_5; g_sk_6; g_sk_7; g_sk_8];
starting_point = ["(0, 0)"; "(1, 1)"; "(8/10, 4/10)"; "(0, 0)"; "(1, 1)"; "(8/10, 4/10)"; "(0,
0)"; "(1, 1)"; "(8/10, 4/10)"];
func_values = [r_0; r_1; r_2; r_3; r_4; r_5; r_6; r_7; r_8];
tbl = table(Methods, starting_point, x_values, y_values, iter_count, fja_count, func_values);
display(tbl)
writetable(tbl, "results.xlsx")

[X, Y] = meshgrid(-1.5:0.1:1.5, -1.5:0.1:1.5);
figure;
surf(X, Y, f(X, Y));
hold on;
plot3(X_4, Y_4, f(X_4, Y_4), 'ro', 5);
    hold off
    xlabel('x');
    ylabel('y');
    zlabel('f(x, y)');
```

1.7. Gradientinio nusileidimo metodo realizacija

```
function [x_0, y_0, iter, g_sk, L, x_arr, y_arr, value] = gradientinis(f, g, x0, y0, gamma,
epsilon, max_it)
L=100;
iter=0;
g_sk=0;
x_arr = zeros (max_it, 1);
```

```
y_arr = zeros (max_it, 1);
while L>=epsilon && iter<max_it
gradient = g(x0, y0);
g_sk=g_sk+1;
x_0=x0-gamma*gradient(1);
y_0=y0-gamma*gradient(2);
L=abs(x0-x_0);
x0=x_0;
y0=y_0;
iter=iter+1;
x_arr(iter) = x0;
y_arr(iter) = y0;
end
x_arr = x_arr(1:iter);
y_arr = y_arr(1:iter);
value=f(x0, y0);
end
```

1.8. Greičiausio nusileidimo realizacija

```
function [x_0, y_0, iter, g_sk, L, x_arr, y_arr, value] =greiciausias (g, x0, y0, epsilon,
max_it, f)
L=100;
iter=0;
g_sk=0;
x_arr = zeros (max_it, 1);
y_arr = zeros (max_it, 1);
while L> epsilon && iter < max_it
    gradient = g(x0, y0);
    g_sk = g_sk+1;
    %perskaiciuojami x_0 ir y_0
    [x, L, it, x_arr_gold, x1_arr_gold, x2_arr_gold, call]=auksinis(f, 0.0001, 100, 0, 6,
gradient, x0, y0);
    x_0=x0-x*gradient(1);
    y_0=y0-x*gradient(2);
    L=abs(x0-x_0);
    x0=x_0;
    y0=y_0;
    iter=iter+1;
    x_arr(iter) = x0;
    y_arr(iter) = y0;
end
x_arr = x_arr(1:iter);
y_arr = y_arr(1:iter);
value=f(x0, y0);
end
```

1.9. Auksinio pjūvio realizacija

```
function [x, L, it, x_arr_gold, x1_arr_gold, x2_arr_gold, call] = auksinis(f, epsilon, max_it,
l, r, gradient, x0, y0)
t = (1+sqrt(5))/2-1;
L= r-l;
x1 = r-t*L;
x2 = 1 + t*L;
it = 0;
x_arr_gold = [];
x1_arr_gold = x1;
x2_arr_gold = x2;
fx1 = f(x0-x1*gradient(1), y0-x1*gradient(2));
fx2 = f(x0-x2*gradient(1), y0-x2*gradient(2));
call = 2;
if fx1<fx2
    x = x1;
else
    x = x2;
```

```
end
x_arr_gold (it + 1) = x;
while L > epsilon && it < max_it
    if fx2 < fx1
        l = x1;
        L = abs(r-1);
        x1 = x2;
        x2 = 1+t*L;
        fx1 = fx2;
        fx2 = f(x0-x2*gradient(1), y0-x2*gradient(2));
        call = call + 1;
    else
        r = x2;
        L = abs(r-1);
        x1 = r-t*L;
        fx2 = fx1;
        fx1 = f(x0-x1*gradient(1), x0-x1*gradient(2));
        call = call + 1;
        it = it +1;
    end
    it = it +1;
    if fx1<fx2
        x = x1;
    else
        x = x2;
    end
    x_arr_gold (it + 1) = x;
    x1_arr_gold(it+1)=x1;
    x2_arr_gold(it+1)=x2;
end
```

1.10. Deformuojamo simplekso realizacija

```
function [triangle, iter, f_sk, X, Y, values, x_0, y_0] = simpleksas (f, x0, y0, max_it, alfa)
gamma=1;
beta=0.5;
epsilon=-0.5;
iter=0;
delta1=((sqrt(3)+1)/(2*sqrt(2)))*alfa;
delta2=((sqrt(3)-1)/(2*sqrt(2)))*alfa;
L=100;
x1 = [x0, y0];
x2 = [delta1, delta2];
x3 = [delta2, delta1];
triangle = [x1; x2; x3];
f_sk=0;
X=[x0, delta1, delta2];
Y=[y0, delta2, delta1];
while iter < max_it && L>0.0001
    iter = iter + 1;
    values=[f(triangle (1, 1), triangle (1, 2)); f(triangle (2, 1), triangle (2, 2)); f(triangle
(3, 1), triangle (3, 2))];
    [values, index] = sort(values);
    triangle = triangle (index,:);
    f_sk=f_sk+3;
    L=abs(triangle (1,1)-triangle (2,1));
    center = mean (triangle (1:end-1,:));
    x_new = center + alfa * (center - triangle (end-1:end));
    f_new = f(x_new (1), x_new (2));
    f_sk = f_sk + 1;
    if (values (1) <= f_new) && (f_new < values (end))
        triangle (3,1)=x_new(1);
        triangle (end)=x_new (2);
        X=[X, x_new (1)];
        Y=[Y, x_new(2)];
    elseif f_new < values (1)
```

```
        x=center+gamma*(x_new-center);
        fx=f(x(1), x(2));
        f_sk=f_sk+1;
        if fx < f_new
            triangle (3,1)=x(1);
            X=[X, x(1)];
            Y=[Y, x(2)];
            triangle (end)=x(2);
        else
            triangle (3,1)=x_new(1);
            triangle (end)=x_new(2);
            X=[X, x_new(1)];
            Y=[Y, x_new(2)];
        end
    else
        xe=center + beta * (triangle (end-1:end)-center); fe=f(xe(1), xe(2));
        f_sk=f_sk+1;
        if fe < values (end)
            triangle (3, 1)=xe(1);
            triangle (end)=xe(2);
            X=[X, x(1)];
            Y=[Y, x(2)];
        else
            triangle (2,1)=triangle (2,1)+epsilon* (triangle (2,1)-triangle (1,1));
            triangle (2,2)=triangle (2,2)+epsilon* (triangle (2,2)-triangle (1,2));
            triangle (3,1)=triangle (3,1)+epsilon* (triangle (3,1)-triangle (3,1)); triangle
(3,2)=triangle (3,2)+epsilon* (triangle (3,2)-triangle (3,2));
            X=[X, triangle(2,1)];
            Y=[Y, triangle(2,2)];
            X=[X, triangle(3,1)];
            Y=[Y, triangle(3,2)];
        end
    end
end
values=[f(triangle (1, 1), triangle(1, 2))];
x_0=triangle(1,1);
y_0=triangle(1,2);
```