



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ «Информатика и системы управления»

КАФЕДРА _____ «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 3
по курсу «Методы оптимизации»
«Симплексный алгоритм и алгоритм Нельдера-Мида»

Студент группы ИУ9-81Б Окутин Д.А.

Преподаватель Посевин Д. П.

Москва 2025

1 Задание

1. Реализовать симплексный метод поиска минимума.
2. Реализовать метод Нelderа-Мида для поиска минимума.

2 Реализация

Исходный код программы представлен в листинге 1.

Листинг 1: code

```
1
2 using Plots
3 using LinearAlgebra
4
5 plotly()
6
7 function simplex_method(x0)
8     xs = []
9     x1 = x0
10    x2 = [(sqrt(3)+1)/(2*sqrt(2)), (sqrt(3)-1)/(2*sqrt(2))]
11    x3 = [(sqrt(3)-1)/(2*sqrt(2)), (sqrt(3)+1)/(2*sqrt(2))]
12    points = [x1,x2,x3]
13    points[1] = x1
14    points[2] = x2
15    points[3] = x3
16    push!(xs, x1)
17    push!(xs, x2)
18    push!(xs, x3)
19    while true
20        if(norm(points[1]- points[2])) < 0.001
21            return points[1], xs
22        end
23        if(f(points[2]) >= f(points[1]) && f(points[2]) >= f(points[3]))
24            temp = points[1]
25            points[1] = points[2]
26            points[2] = temp
27        elseif (f(points[3]) >= f(points[1]) && f(points[3]) >= f(points
28            [2]))
29            temp = points[1]
30            points[1] = points[3]
31            points[3] = temp
32        end
33        x4 = points[2] + points[3] - points[1]
34        if(f(x4) >= f(points[2]) && f(x4) >= f(points[3]))
```

```

34         points[1] = x4
35         points[2] = x4+(points[2] - x4)/2
36         points[3] = x4+(points[3] - x4)/2
37         push!(xs, x4)
38         push!(xs, points[2])
39         push!(xs, points[3])
40     else
41         points[1] = x4
42         push!(xs, x4)
43         push!(xs, points[2])
44         push!(xs, points[3])
45     end
46 end
47 end
48
49 function nelder_meed(x0)
50     x1 = x0
51     x2 = [(sqrt(3)+1)/(2*sqrt(2)), (sqrt(3)-1)/(2*sqrt(2))]
52     x3 = [(sqrt(3)-1)/(2*sqrt(2)), (sqrt(3)+1)/(2*sqrt(2))]
53     points = [x1,x2,x3]
54     points[1] = x1
55     points[2] = x2
56     points[3] = x3
57     xs = []
58     center = x0
59     beta = 2.0
60     while true
61         points = sort(points, by=x -> f(x), rev=true)
62         push!(xs, points[3])
63         push!(xs, points[1])
64         push!(xs, points[2])
65         center = (points[2]+points[3])/2.0
66         if (sqrt(((f(points[1]) - f(center))^2 +((f(points[2]) - f(
center))^2) + ((f(points[2]) - f(center))^2))/(3.0)) < 0.001)
67             return points[3],xs
68         end
69         x4 = points[2]+points[3]-points[1]
70         beta = 2.0
71         y_min = f(points[3])
72         if (f(x4) < y_min)
73             beta = 2.0
74             x5 = beta*x4 + (1-beta)*center
75             if (f(x5)< f(x4) && f(x5)< f(points[3]) && f(x5)< f(points
[2]))
76                 points[1] = x5
77             else

```

```

78         if (f(x5) > f(x4))
79             points[1] = x4
80         end
81     end
82 else
83     if f(points[3]) < f(x4) < f(points[2])
84         points[1] = x4
85     else
86         if f(points[2]) < f(x4) < f(points[1])
87             points[1] = x4
88         end
89         points = sort(points, by=x -> f(x), rev=true)
90         beta = 0.5
91         x5 = beta * points[1] + (1 - beta) * center
92         if f(x5) < f(points[1])
93             points[1] = x5
94         else
95             #c
96             points[1] = points[3] + 0.5 * (points[1] - points
97 [3])
98             points[2] = points[3] + 0.5 * (points[2] - points
99 [3])
100         end
101     end
102 end
103
104 f(x) = (1.0 - x[1])^2 + 100.0*(x[2] - x[1]^2)^2
105
106 x0 = [-1.0, -1.0]
107
108 x = -2.5:0.1:2.5
109 y = -2.5:0.1:2.5
110
111
112 plt = surface(x, y, (x, y) -> f([x, y]), color=:thermal, alpha=0.5,
113             legend=true)
114 println(" ")
115 r, xs = simplex_method(x0)
116 println(xs)
117 println(" : ", r)
118
119 triangles_simplex = [xs[i:i+2] for i in 1:3:length(xs)]
120 x_coords_simplex = [x[1] for x in xs]

```

```

121 y_coords_simplex = [y[2] for y in xs]
122
123
124 res, xs = nelder_meed(x0)
125 println("          -          : ", res)
126
127 triangles_nm = [xs[i:i+2] for i in 1:3:length(xs)]
128 x_coords_nm = [x[1] for x in xs]
129 y_coords_nm = [y[2] for y in xs]
130
131 println()
132
133 x = -2.5:0.1:2.5
134 y = -2.5:0.1:2.5
135
136 plt5 = surface(x, y, (x, y) -> f([x, y]), color=:thermal, alpha=0.5,
               legend=true)
137
138 Z = map((a,b) -> f([a,b]), x_coords_nm, y_coords_nm)
139
140 for (i, triangle) in enumerate(triangles_nm)
141     x1 = [triangle[1][1], triangle[2][1], triangle[3][1], triangle
           [1][1]] #
           x
142     y1 = [triangle[1][2], triangle[2][2], triangle[3][2], triangle
           [1][2]] #
           y
143     z1 = [f([x1[1], y1[1]]), f([x1[2], y1[2]]), f([x1[3], y1[3]]), f([x1[4],
           y1[4]])]
144     plot!(plt5, x1, y1, z1, linecolor=:blue)
145 end
146
147 scatter3d!(plt5, [x0[1]], [x0[2]], [f(x0)], color=:green, label="Start")
148 scatter3d!(plt5, [res[1]], [res[2]], [f(res)], color=:blue, label="End")
149
150 xlabel!(plt5, "x")
151 ylabel!(plt5, "y")
152 zlabel!(plt5, "f(x,y)")
153
154
155 plt6 = surface(x, y, (x, y) -> f([x, y]), color=:thermal, alpha=0.5,
               legend=true)
156
157 Z = map((a,b) -> f([a,b]), x_coords_simplex, y_coords_simplex)
158
159
160 scatter3d!(plt6, [x0[1]], [x0[2]], [f(x0)], color=:green, label="Start")
161 scatter3d!(plt6, [r[1]], [r[2]], [f(r)], color=:blue, label="End")

```



```

202     levels=20,
203     xlabel=" x ",
204     ylabel=" x ",
205     title=" (2D) ",
206     legend=:topleft ,
207     size=(700, 500)
208 )
209
210 colors = [:red, :green, :blue, :orange, :purple, :yellow]
211
212 scatter!(plt2,
213     [x0[1], r[1]],
214     [x0[2], r[2]],
215     color=[:green :blue],
216     markersize=8,
217     label=[" " " " ""])
218 )
219
220 for (i, triangle) in enumerate(triangles_simplex)
221     x1 = [triangle[1][1], triangle[2][1], triangle[3][1], triangle
222           [1][1]] # x
223     y1 = [triangle[1][2], triangle[2][2], triangle[3][2], triangle
224           [1][2]] # y
225     plt2 = plot!(x1, y1, color=colors[i % length(colors) + 1], linewidth
226                  =2, label="Triangle $i")
227 end
228
229 display(plt6)
230 display(plt2)
231 display(plt5)
232 display(plt1)
233
234 f(x) = 20 + x[1]^2 - 10*cos(2*pi*x[1]) + x[2]^2 - 10*cos(2*pi*x[2])
235
236 x0 = [0.5, 0.5]
237
238 x = -2.5:0.1:2.5
239 y = -2.5:0.1:2.5
240
241 plt = surface(x, y, (x, y) -> f([x, y]), color=:thermal, alpha=0.5,
242              legend=true)
243
244 println(" ")
245 r, xs = simplex_method(x0)
246 println(" : ", r)

```

```

244 triangles_simplex = [xs[i:i+2] for i in 1:3:length(xs)]
245 x_coords_simplex = [x[1] for x in xs]
246 y_coords_simplex = [y[2] for y in xs]
247
248 res, xs = nelder_meed(x0)
249 println("          -          : ", res)
250
251 triangles_nm = [xs[i:i+2] for i in 1:3:length(xs)]
252 x_coords_nm = [x[1] for x in xs]
253 y_coords_nm = [y[2] for y in xs]
254
255 println()
256
257 f(x) = 418.9829*2 - (x[1]*sin(sqrt(abs(x[1])))) + x[2]*sin(sqrt(abs(x[2]))
    ))
258
259 x0 = [0.0, 0.0]
260
261 x = -6:0.1:6
262 y = -6:0.1:6
263
264
265 plt = surface(x, y, (x, y) -> f([x, y]), color=:thermal, alpha=0.5,
    legend=true)
266 display(plt)
267
268
269 println("          ")
270 r, xs = simplex_method(x0)
271 println("          : ", r)
272 triangles_simplex = [xs[i:i+2] for i in 1:3:length(xs)]
273 x_coords_simplex = [x[1] for x in xs]
274 y_coords_simplex = [y[2] for y in xs]
275
276 res, xs = nelder_meed(x0)
277 println("          -          : ", res)
278
279 triangles_nm = [xs[i:i+2] for i in 1:3:length(xs)]
280 x_coords_nm = [x[1] for x in xs]
281 y_coords_nm = [y[2] for y in xs]
282
283 println()

```


3 Результаты

Результаты запуска представлены на рисунках 1 - ??.

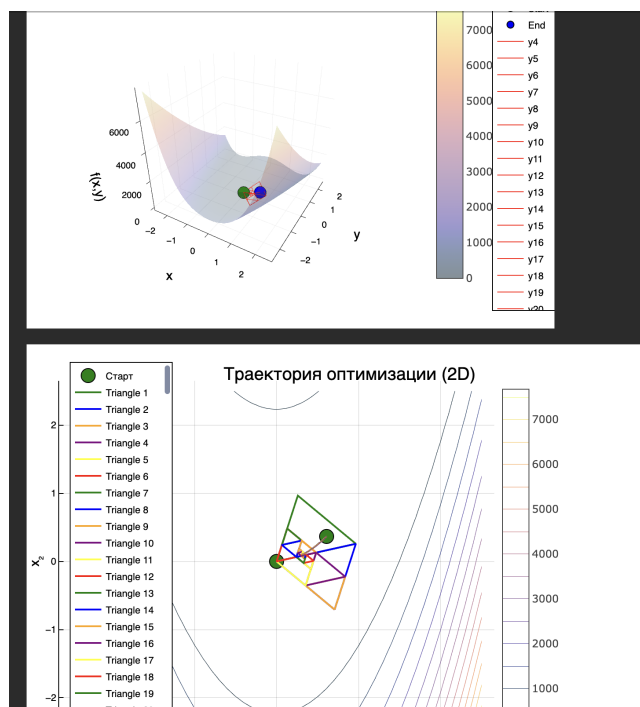


Рис. 1 — Симплексный метод (Розенброк)

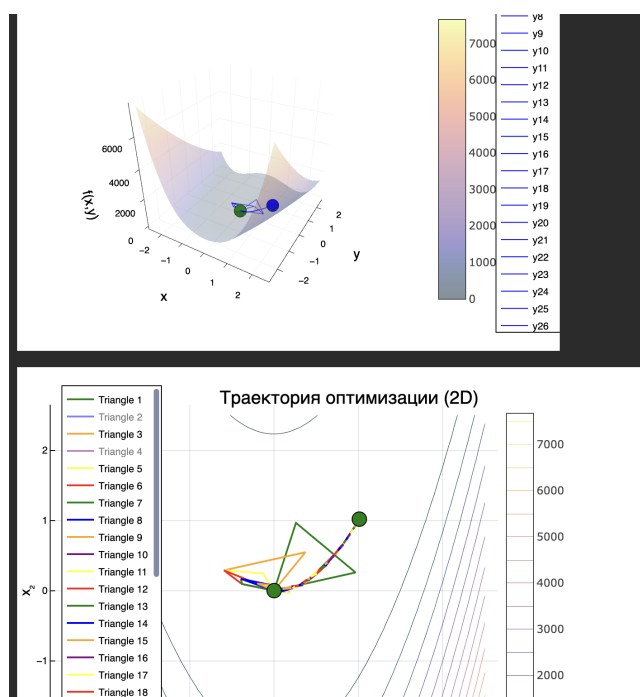


Рис. 2 — Метод Нельдера-Мида (Розенброк)

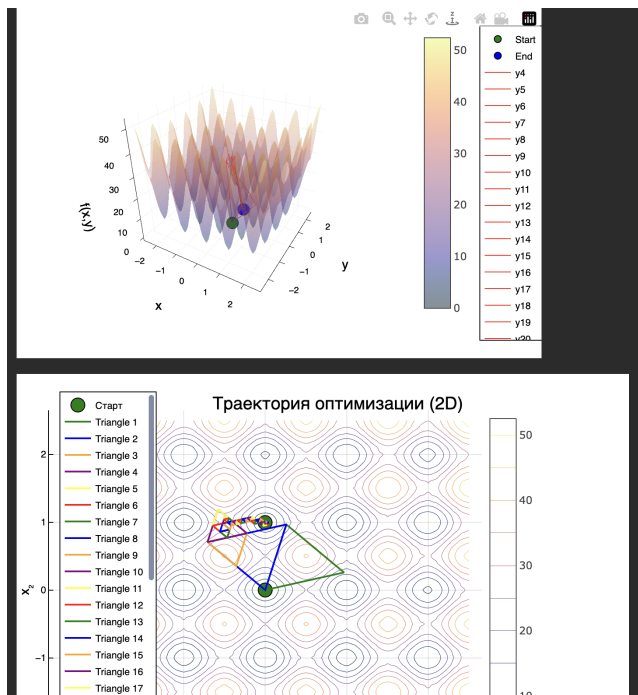


Рис. 3 — Симплексный метод (Растрингин)

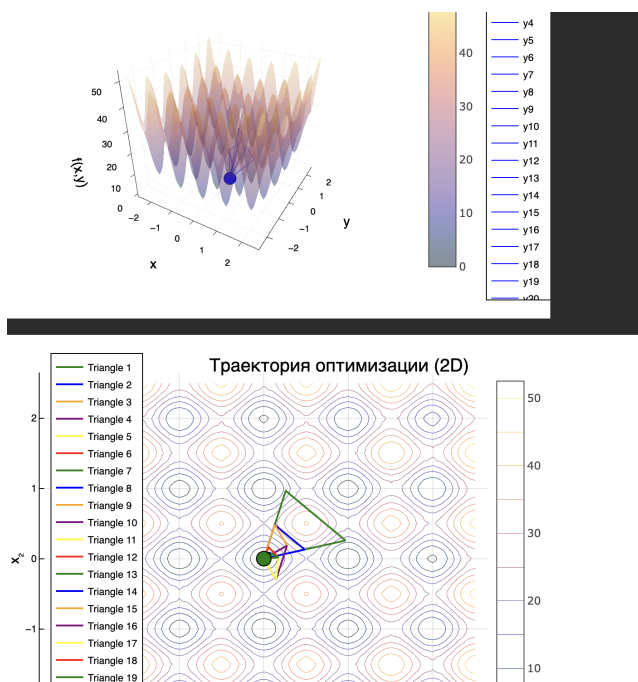


Рис. 4 — Метод Нельдера-Мида (Растрингин)

4 Выводы

В ходе выполнения лабораторной работы были реализованы и сравнены методы оптимизации многомерных функций. Поиск экстремума был реализован с помощью следующих методов: метод Нельдера-Мида, симплексный алгоритм.

Наилучший результат показал метод Нельдера-Мида, который использует растяжение и редукцию симплексов.