



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 № 2
по курсу «Методы оптимизации»
«Численные методы минимизации функций нескольких
переменных»

Студент группы ИУ9-81Б Окутин Д.А.

Преподаватель Посевин Д. П.

Moskva 2025

1 Задание

1. Реализовать метод покоординатного спуска.
2. Реализовать метод Гусса-Зейделя.
3. Реализовать метод Хука-Дживса.

2 Реализация

Исходный код программы представлен в листинге 1.

Листинг 1: code

```
1
2 using Plots
3 using LinearAlgebra
4
5 #                                     (
6 f(x) = (1.0 - x[1])^2 + 100.0*(x[2] - x[1]^2)^2
7 # f(x) = x[1]^2+x[2]^2 + 2*x[1]*x[2]
8
9 #
10 x0 = [-2.0, -2.0]
11
12 function coordinate_descent1(f, x0, eps1=1e-6, eps2=1e-10)
13     x = x0
14     n = length(x)
15     trajectory = [x]
16     alpha = 0.1
17
18     while true
19         x_new = copy(x)
20         min_f = f(x)
21         changed = false
22         while true
23             for i in 1:n
24                 for j in [-1, 1]
25                     xi = copy(x)
26                     xi[i] += alpha * j
27                     if f(xi) <= min_f
28                         x_new = xi
29                         min_f = f(xi)
30                         changed = true
31                     end
32                 end
33             end
34         end
35     end
36 end
```

```

33         end
34         if changed
35             break
36         end
37         alpha /= 2
38     end
39
40     if norm(x - x_new) < eps1 || norm(f(x) - f(x_new)) < eps2
41         break
42     end
43
44     x = copy(x_new)
45     push!(trajectory, x)
46 end
47
48 return x, trajectory
49 end
50
51
52 result_coord, history_coord = coordinate_descent1(f, x0)
53
54 println(result_coord)
55 println(" - : ", length(history_coord))
56
57 # 3D
58 x = range(-2.0, 2.0, length=100)
59 y = range(-2.0, 2.0, length=100)
60 X = repeat(reshape(x, 1, :), length(y), 1)
61 Y = repeat(y, 1, length(x))
62 Z = map((a,b) -> f([a,b]), X, Y)
63
64 # 3D
65 plt = surface(x, y, Z, color=:thermal, alpha=0.5, legend=true)
66 # scatter3d!(plt,
67 #     [p[1] for p in history_coord],
68 #     [p[2] for p in history_coord],
69 #     [f(p) for p in history_coord],
70 #     markersize=3,
71 #     color=:red,
72 #     label="Search Path"
73 # )
74
75 #
76 plot!(plt,
77     [p[1] for p in history_coord],
78     [p[2] for p in history_coord],

```

```

79      [f(p) for p in history_coord] ,
80      linecolor=:blue ,
81      label="Connecting Lines"
82 )
83
84 scatter3d!( plt , [x0[1]] , [x0[2]] , [f(x0)] , color=:green , label="Start")
85 scatter3d!( plt , [result_coord[1]] , [result_coord[2]] , [f(result_coord)] ,
86             color=:blue , label="End")
87 title !( plt , "
88 xlabel!( plt , "x")
89 ylabel!( plt , "y")
90 zlabel!( plt , "f(x,y)")
91
92 #
93 plt = contour(x, y, Z,
94                 color=:thermal ,
95                 levels=20,
96                 xlabel=" x   " ,
97                 ylabel=" x   " ,
98                 title="                                     (2D) " ,
99                 legend=:topleft
100 )
101
102 #
103 threshold = 0.1
104 filtered_x = []
105 filtered_y = []
106
107 last_x = nothing
108 last_y = nothing
109
110 for p in history_coord
111     if last_x === nothing || abs(p[1] - last_x) > threshold || abs(p[2]
112         - last_y) > threshold
113         push!(filtered_x , p[1])
114         push!(filtered_y , p[2])
115         last_x = p[1]
116         last_y = p[2]
117     end
118 end
119 plot !( plt ,
120         filtered_x ,
121         filtered_y ,

```

```

122      color=:red ,
123      linewidth=1,
124      label="",
125      marker=:circle ,
126      markersize=4
127 )
128
129 print(x0[1],x0[2])
130
131 #
132 scatter!(plt ,
133     [x0[1], result_coord[1]] ,
134     [x0[2], result_coord[2]] ,
135     color=[:green :blue] ,
136     markersize=8,
137     label=[" " ""])
138 )
139
140 #
141 plot!(plt , colorbar=true)
142
143 #
144
145 annotate!(plt ,
146     [(x0[1], x0[2], text("f = $(round(f(x0), digits=3))", 8, :left)),
147     (result_coord[1], result_coord[2], text("f = $(round(f(result_coord),
148     digits=3))", 8, :right))])
149 using Plots
150 using LinearAlgebra
151
152 function swann_method(f , x0 , h=0.1)
153     first = x0
154     second = x0 + h
155     if f(second) > f(first)
156         h = -h
157         first , second = second , second + h
158     end
159     last = second + h
160
161     while f(last) < f(second)
162         h *= 2
163         first , second , last = second , last , last + h
164     end
165     if second > last

```

```

166      first , second , last = last , second , first
167  end
168
169  return first , last
170end
171
172function golden_section(f , a , b)
173    k = (sqrt(5) - 1) / 2
174    x1 = a + (1 - k) * (b - a)
175    x2 = a + k * (b - a)
176
177    a = Float64(a)
178    b = Float64(b)
179
180    while abs(x1 - x2) > 1e-6
181        if f(x1) <= f(x2)
182            b = x2
183            x2 = x1
184            x1 = a + b - x1
185        else
186            a = x1
187            x1 = x2
188            x2 = a + b - x2
189        end
190    end
191    return (a + b) / 2
192end
193
194function gauss_zeidel(f , x0 , eps1=1e-6 , eps2=1e-10)
195    x = x0
196    n = length(x)
197    trajectory = [x]
198
199    while true
200        x_new = copy(x)
201        min_f = f(x)
202        for i in 1:n
203            for j in [-1 , 1]
204                function g(alpha)
205                    xi = copy(x)
206                    xi[i] += alpha * j
207                    return f(xi)
208                end
209                l , r = swann_method(g , 0.5)
210                alpha = golden_section(g , l , r)
211                xi = copy(x)

```

```

212         xi[ i ] += alpha * j
213         if f( xi ) < min_f
214             x_new = xi
215             min_f = f( xi )
216         end
217     end
218 end
219
220 if norm( x - x_new ) < eps1 || norm( f( x ) - f( x_new ) ) < eps2
221     break
222 end
223
224 x = copy( x_new )
225 push!( trajectory , x )
226 end
227
228 return x, trajectory , length( trajectory )
229 end
230
231 result_gauss , history_gauss ,iters_count_gauss = gauss_zeidel(f, x0)
232
233 println(result_gauss)
234 println("      -
235
236 #                                     3D
237 x = range( -2.0 , 2.0 , length=100)
238 y = range( -2.0 , 2.0 , length=100)
239 X = repeat( reshape(x, 1, :) , length(y) , 1)
240 Y = repeat(y, 1, length(x))
241 Z = map((a,b) -> f([a,b]) , X, Y)
242
243 #                                     3D
244 plt = surface(x, y, Z, color=:thermal , alpha=0.5, legend=true)
245 # scatter3d!( plt ,
246 #     [p[1] for p in history_gauss] ,
247 #     [p[2] for p in history_gauss] ,
248 #     [f(p) for p in history_gauss] ,
249 #     markersize=3,
250 #     color=:red ,
251 #     label="Search Path"
252 # )
253
254 #
255 plot!( plt ,
256     [p[1] for p in history_gauss] ,
257     [p[2] for p in history_gauss] ,

```

```

258 [ f(p) for p in history_gauss] ,
259 linecolor=:blue ,
260 label="Connecting Lines"
261 )
262 scatter3d!( plt , [x0[1]] , [x0[2]] , [f(x0)] , color=:green , label="Start")
263 scatter3d!( plt , [result_gauss[1]] , [result_gauss[2]] , [f(result_gauss)] ,
264 color=:blue , label="End")
265 title!( plt , " - ")
266 xlabel!( plt , "x")
267 ylabel!( plt , "y")
268 zlabel!( plt , "f(x,y)")
269
270 using Plots
271
272 #
273 plt = contour(x, y, Z,
274 color=:thermal ,
275 levels=20,
276 xlabel=" x " ,
277 ylabel=" x " ,
278 title=" (2D) " ,
279 legend=:topleft
280 )
281
282 threshold = 0.1
283 filtered_x = []
284 filtered_y = []
285
286 last_x = nothing
287 last_y = nothing
288
289 for p in history_gauss
290 if last_x === nothing || abs(p[1] - last_x) > threshold || abs(p[2]
291 - last_y) > threshold
292 push!( filtered_x , p[1])
293 push!( filtered_y , p[2])
294 last_x = p[1]
295 last_y = p[2]
296 end
297 end
298 plot!( plt ,
299 filtered_x ,
300 filtered_y ,
301 color=:red ,

```

```

302     linewidth=1,
303     label="",
304     marker=:circle,
305     markersize=4
306 )
307
308 print(x0[1],x0[2])
309
310 #
311 scatter!(plt,
312     [x0[1], result_gauss[1]],
313     [x0[2], result_gauss[2]],
314     color=[:green :blue],
315     markersize=8,
316     label=[" " ""])
317 )
318
319 #
320 plot!(plt, colorbar=true)
321
322 #

323 annotate!(plt,
324     [(x0[1], x0[2], text("f = $(round(f(x0), digits=3))", 8, :left)),
325      (result_gauss[1], result_gauss[2], text("f = $(round(f(result_gauss),
326      digits=3))", 8, :right))])
327
328 function exploratory_search(f, x, delta)
329     n = length(x)
330     x_new = copy(x)
331     for i in 1:n
332         start_val = f(x_new)
333         temp = x_new[i]
334         x_new[i] = temp + delta[i]
335         f_plus = f(x_new)
336         if f_plus >= start_val
337             x_new[i] = temp - delta[i]
338             f_minus = f(x_new)
339             if f_minus >= start_val
340                 x_new[i] = temp
341             end
342         end
343     end
344     return x_new
345 end

```

```

346
347 function hooke_jeeves(f , x0 , eps=1e-6 , delta0=0.1)
348     n = length(x0)
349     delta = fill(delta0 , n)
350     x = copy(x0)
351     trajectory = [x]
352
353     while maximum(delta) > eps
354         x_exp = exploratory_search(f , x , delta)
355
356         if f(x_exp) >= f(x)
357             delta = delta / 2
358         end
359         while f(x_exp) < f(x)
360             direction = x_exp - x
361             x = copy(x_exp)
362             push!(trajectory , x)
363             x_pattern = x + direction
364             x_exp = exploratory_search(f , x_pattern , delta)
365         end
366     end
367
368     return x , trajectory
369 end
370
371 result_hooke , history_hooke = hooke_jeeves(f , x0)
372 println(result_hooke)
373 println("           - : " , length(history_hooke))
374
375 #                                     3D
376 x = range(-2.0 , 2.0 , length=100)
377 y = range(-2.0 , 2.0 , length=100)
378 X = repeat(reshape(x , 1 , :) , length(y) , 1)
379 Y = repeat(y , 1 , length(x))
380 Z = map((a,b) -> f([a,b]) , X , Y)
381
382 #                                     3D
383 plt = surface(x , y , Z , color=:thermal , alpha=0.5 , legend=true)
384 # scatter3d!(plt ,
385 #     [p[1] for p in history_hooke] ,
386 #     [p[2] for p in history_hooke] ,
387 #     [f(p) for p in history_hooke] ,
388 #     markersize=3,
389 #     color=:red ,
390 #     label="Search Path"
391 # )

```

```

392#
393# plot!( plt ,
394    [p[1] for p in history_hooke],
395    [p[2] for p in history_hooke],
396    [f(p) for p in history_hooke],
397    linecolor=:blue ,
398    label="Connecting Lines"
399)
400)
401
402 scatter3d!( plt , [x0[1]] , [x0[2]] , [f(x0)] , color=:green , label="Start")
403 scatter3d!( plt , [result_hooke[1]] , [result_hooke[2]] , [f(result_hooke)] ,
404     color=:blue , label="End")
405 title!( plt , "
406 xlabel!( plt , "x")
407 ylabel!( plt , "y")
408 zlabel!( plt , "f(x,y)")
409
410 using Plots
411
412#
413 plt = contour(x, y, Z,
414     color=:thermal ,
415     levels=20,
416     xlabel=" x " ,
417     ylabel=" x " ,
418     title=" (2D) " ,
419     legend=:topleft
420)
421
422#
423 threshold = 0.1
424 filtered_x = []
425 filtered_y = []
426
427 last_x = nothing
428 last_y = nothing
429
430 for p in history_hooke
431     if last_x === nothing || abs(p[1] - last_x) > threshold || abs(p[2]
432         - last_y) > threshold
433         push!(filtered_x , p[1])
434         push!(filtered_y , p[2])
435         last_x = p[1]

```

```

435         last_y = p[2]
436     end
437 end
438
439 plot!( plt ,
440         filtered_x ,
441         filtered_y ,
442         color=:red ,
443         linewidth=1,
444         label="" ,
445         marker=:circle ,
446         markersize=4
447 )
448
449 #
450 scatter!( plt ,
451         [x0[1], result_hooke[1]] ,
452         [x0[2], result_hooke[2]] ,
453         color=[:green :blue] ,
454         markersize=8,
455         label=[" " " "]
456 )
457
458 #
459 plot!( plt , colorbar=true)
460
461 #
462 annotate!( plt ,
463         [(x0[1], x0[2], text("f = $(round(f(x0), digits=3))", 8, :left)),
464         (result_hooke[1], result_hooke[2], text("f = $(round(f(result_gauss),
465         digits=3))", 8, :right))]
```

3 Результаты

Результаты запуска представлены на рисунках 1 - 6.

[0.9975219726562554, 0.9950439453125073]
Кол-во итераций: 5030
Оптимизация методом координатного спуска

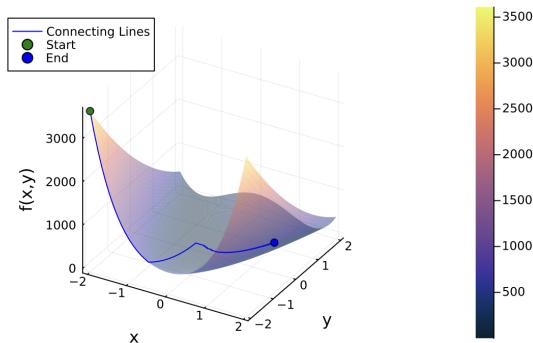


Рис. 1 — Метод покоординатного спуска 3D

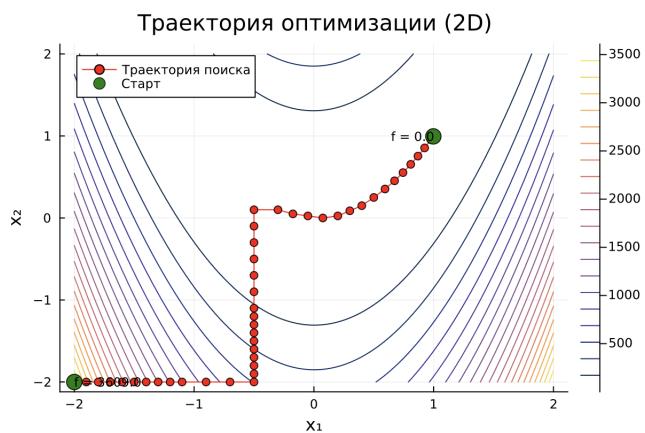


Рис. 2 — Метод покоординатного спуска 2D

[0.9960386598874171, 0.9920930183191654]
Кол-во итераций: 3947
Оптимизация Гаусса-Зейделя

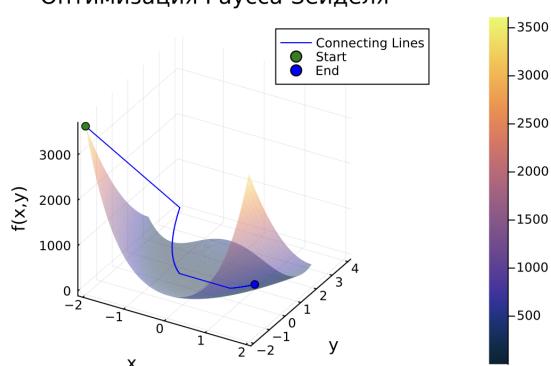


Рис. 3 — Метод Гаусса-Зейделя 3D

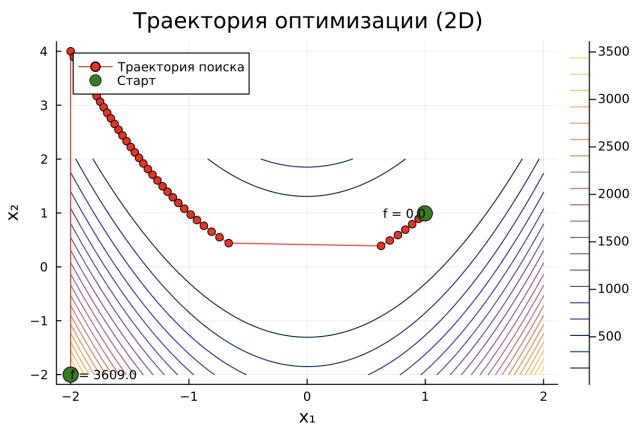


Рис. 4 — Метод Гаусса-Зейделя 2D

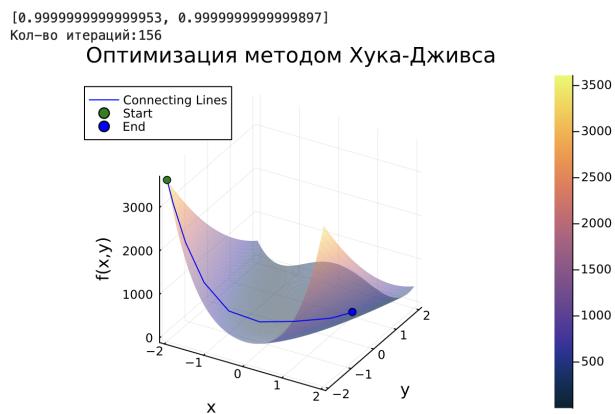


Рис. 5 — Метод Хука-Дживса 3D

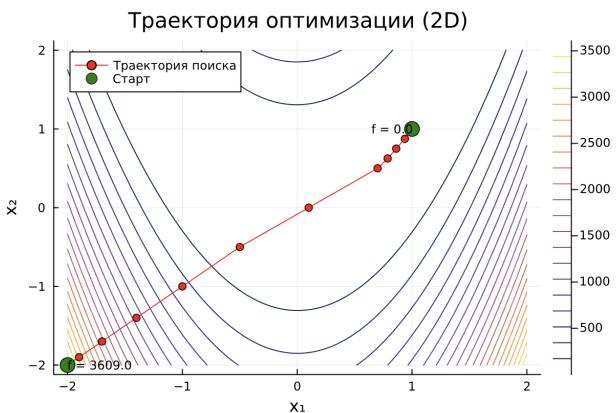


Рис. 6 — Метод Хука-Дживса 2D

4 Выводы

В ходе выполнения лабораторной работы были реализованы различные методы поиска безусловного минимума функций. На графиках была наглядна продемонстрирована работа этих методов. Для первых двух методов мы кажды

раз продвигаемся по 1 координате, тогда как в последнем можем совершать шаги вдоль нескольких координат, тем самым наблюдая лучшую сходимость на овражных функциях.