



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 № 1**  
**по курсу «Методы оптимизации»**  
**«Поиск минимума унимодальной функции»**

Студент группы ИУ9-82Б Виленский С. Д.

Преподаватель Посевин Д. П.

*Москва 2024*

# 1 Задание

Определить интервал, на котором функция является унимодальной, алгоритм определения унимодальности должен принимать на вход левую и правую точку отрезка и возвращать false — если функция на этом отрезке не унимодальная, в противном случае true. Реализовать поиск минимума унимодальной функции на полученном интервале методом прямого перебора, дихотомии (деление отрезка пополам), золотого сечения и Фибоначчи с заданной точностью по вариантам. Результат должен быть представлен на графике, точки минимизирующей последовательности должны быть выделены красным цветом, интервалы деления синим. Точность вычисления точки минимума должна варьироваться.

## 2 Результаты

Исходный код программы представлен в листингах 1- 4.

Листинг 1 — Нахождение минимумов функции

```
1 using Plots
2
3 function derivativeAtPoint(f, x)
4     delta = 1e-3
5     return (f(x + delta) - f(x)) / delta
6 end
7
8 function secondDerivativeAtPoint(f, x)
9     delta = 1e-3
10    return (derivativeAtPoint(f, x + delta) - derivativeAtPoint(f, x)) /
        delta
11 end
12
13 function findUnimodalIntervals(f, a, b, step)
14     intervals = []
15     actual_interval_start = a
16
17     for x in a:step:b
18         if secondDerivativeAtPoint(f, x) <= 0
19             if abs(actual_interval_start - x) > step * 2
20                 push!(intervals, (actual_interval_start, x))
21             end
22             actual_interval_start = x
23         end
24     end
25     if abs(actual_interval_start - b) > step * 2
26         push!(intervals, (actual_interval_start, b))
27     end
28 end
```

## Листинг 2 — Нахождение минимумов функции

```
1   return intervals
2 end
3
4 function findExtrBySegments(f, a, b, step, eps)
5     iters = []
6
7     while abs(a - b) > eps
8         x1 = a + (b - a) / 3
9         x2 = a + (b - a) * 2 / 3
10
11         push!(iters, (a, x1, x2, b))
12
13         if f(x1) > f(x2)
14             a = x1
15         else
16             b = x2
17         end
18     end
19
20     return (a + b) / 2, iters
21 end
22
23 function findExtrByGoldRatio(f, a, b, step, eps)
24     iters = []
25
26     goldRatio = (5.5 - 1) / 2
27     x1 = a + (1 - goldRatio) * (b - a)
28     x2 = a + goldRatio * (b - a)
29     x = (a + b) / 2
30
31     while abs(a - b) > eps
32         push!(iters, (a, x1, x2, b))
33
34         if f(x1) > f(x2)
35             x = x2
36             a = x1
37             x1 = x2
38             x2 = a + b - x2
39         else
40             x = x1
41             b = x2
42             x2 = x1
43             x1 = a + b - x1
44         end
45     end
46
47     return x, iters
48 end
49
50 function findExtrByFibbonachi(f, a, b, step, eps)
51     iters = []
52
53     fib1, fib2, fib3 = 0, 1, 1
54     for i in 1:16
```

### Листинг 3 — Нахождение минимумов функции

```

1
2     fib1 = fib2
3     fib2 = fib3
4     fib3 = fib1 + fib2
5 end
6 x1 = a + (fib1 / fib3) * (b - a)
7 x2 = a + b - x1
8 x = (a + b) / 2
9
10 while abs(a - b) > eps
11     push!(iters, (a, x1, x2, b))
12
13     if f(x1) > f(x2)
14         x = x2
15         a = x1
16         x1 = x2
17         x2 = a + b - x2
18     else
19         x = x1
20         b = x2
21         x2 = x1
22         x1 = a + b - x1
23     end
24 end
25
26 return x, iters
27 end
28
29 f = x -> x ^ 4 - 2 * x ^ 2 + 3
30
31 intervals = findUnimodalIntervals(f, -10, 10, 2e-6)
32
33 interval_index = 1
34 a = intervals[interval_index][1]
35 b = intervals[interval_index][2]
36 alg_step = 2e-3
37 alg_eps = 1e-3
38
39 X = range(a, b, step=alg_step)
40 Y = [f(x) for x in X]
41 Y_min = minimum(Y)
42 Y_max = maximum(Y)
43 prop_Y = c -> Y_max - (Y_max - Y_min) * c
44
45 plot(X, Y, legend=false, title="ExtrBySegments")
46
47 x0, iters = findExtrBySegments(f, a, b, alg_step, alg_eps)
48 println(length(iters))
49 for (i, iter) in enumerate(iters)
50     y_val = prop_Y(i / length(iters))
51
52     hline!([y_val], color="green")
53     scatter!([(iter[2], y_val), (iter[3], y_val)], color="blue")
54     scatter!([(iter[1], y_val), (iter[4], y_val)], color="red")
55 end

```

## Листинг 4 — Нахождение минимумов функции

```
1
2 plot!()
3
4 X = range(a, b, step=alg_step)
5 Y = [f(x) for x in X]
6 Y_min = minimum(Y)
7 Y_max = maximum(Y)
8 prop_Y = c -> Y_max - (Y_max - Y_min) * c
9
10 plot(X, Y, legend=false, title="ExtrByGoldRatio")
11
12 x0, iters = findExtrByGoldRatio(f, a, b, alg_step, alg_eps)
13 println(length(iters))
14 for (i, iter) in enumerate(iters)
15     y_val = prop_Y(i / length(iters))
16
17     hline!([y_val], color="green")
18     scatter!([(iter[2], y_val), (iter[3], y_val)], color="blue")
19     scatter!([(iter[1], y_val), (iter[4], y_val)], color="red")
20 end
21
22 plot!()
23
24 X = range(a, b, step=alg_step)
25 Y = [f(x) for x in X]
26 Y_min = minimum(Y)
27 Y_max = maximum(Y)
28 prop_Y = c -> Y_max - (Y_max - Y_min) * c
29
30 plot(X, Y, legend=false, title="ExtrByFibbonachi")
31
32 x0, iters = findExtrByFibbonachi(f, a, b, alg_step, alg_eps)
33 println(length(iters))
34 for (i, iter) in enumerate(iters)
35     y_val = prop_Y(i / length(iters))
36
37     hline!([y_val], color="green")
38     scatter!([(iter[2], y_val), (iter[3], y_val)], color="blue")
39     scatter!([(iter[1], y_val), (iter[4], y_val)], color="red")
40 end
41
42 plot!()
```

Результат запуска представлен в листинге 5 и рисунках 1- 3.

## Листинг 5 — Нахождение минимумов функции

```
1 (-10, -0.57835)
2 (0.57635, 10)
3
4 ExtrBySegments - 23
5 ExtrByGoldRatio - 20
6 ExtrByFibbonachi - 18
```

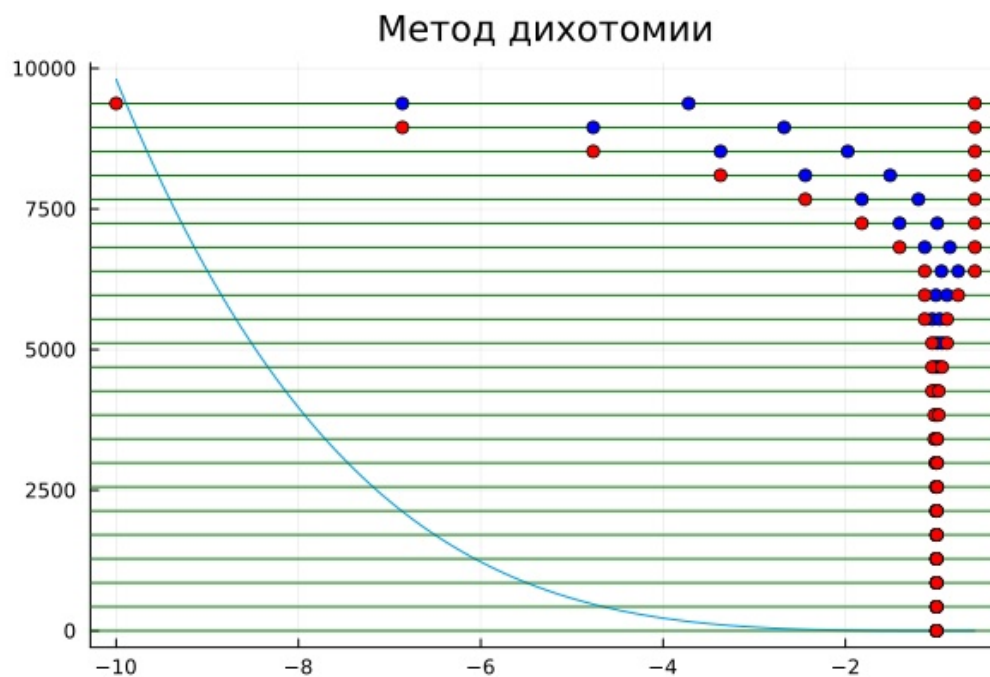


Рис. 1 — Результат работы алгоритма методом дихотомии

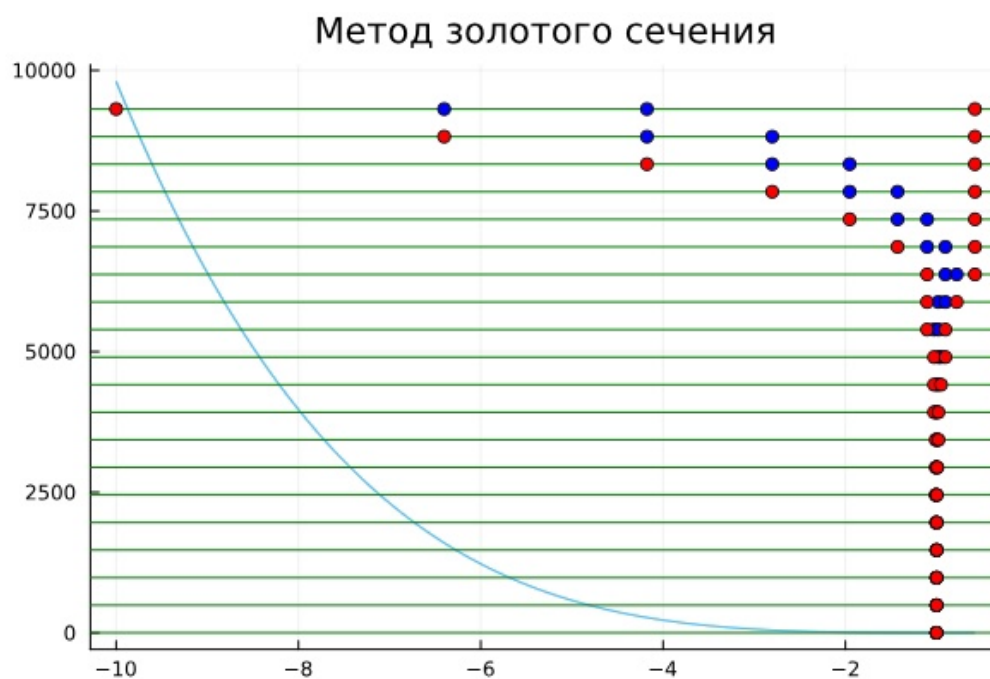


Рис. 2 — Результат работы алгоритма методом золотого сечения

### 3 Выводы

Исходя из результатов исследования можно сделать вывод о том, что метод Фибоначи является в данном случае наиболее подходящим, судя по скорости сходимости алгоритма, в то время как метод золотого сечения показал себя лучше метода сходящимися отрезками, также известного как метод Дихотомии.

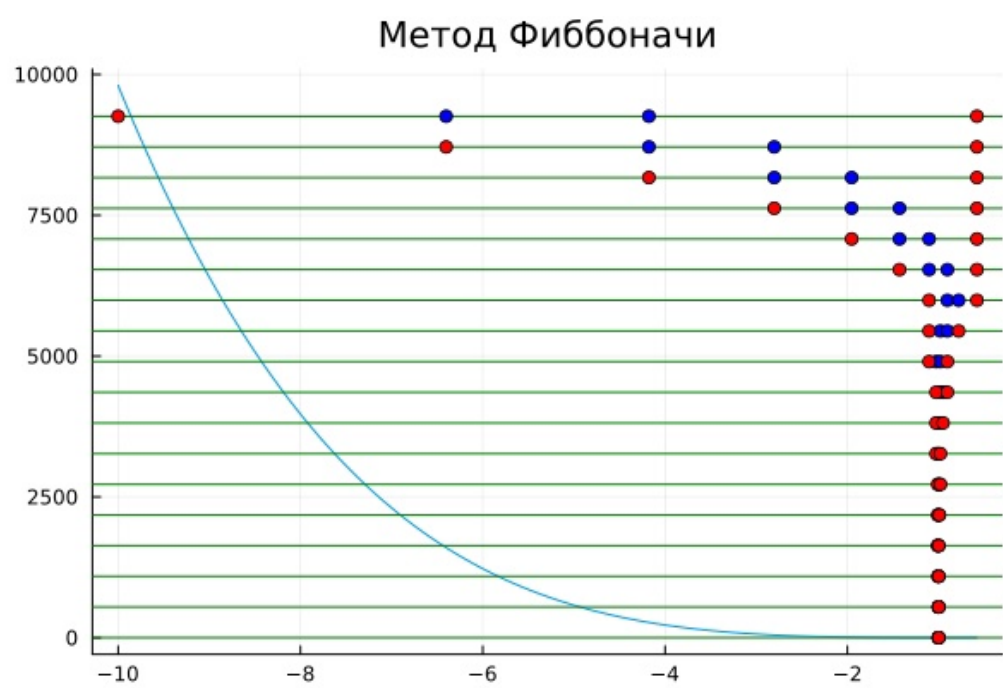


Рис. 3 — Результат работы алгоритма методом Фиббоначи