МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Новосибирский государственный технический университет»

НГТУ НЭТИ

Кафедра теоретических и прикладных приколов

Лабораторная работа №1
по дисциплине «Методы оптимизации»
Методы одномерного поиска

ФПМИ

Группа	ПМ-92
Бригада	08
Студенты	БЕГИЧЕВ АЛЕКСАНДР
	ШИШКИН НИКИТА
Преподаватель	ФИЛИППОВА Е.П.
Дата	22.02.2022

Новосибирск

Цель работы

Ознакомиться с методами одномерного поиска [3, 12], используемыми в многомерных методах минимизации функций n переменных. Сравнить различные алгоритмы по эффективности на тестовых примерах.

Задание

Вариант 8: $f(x) = (x-8)^2, x \in [-2, 20]$.

1. Реализовать методы дихотомии, золотого сечения, исследовать их сходимость и провести сравнение по числу вычислений функции для достижения заданной точности ε от 10^{-1} до 10^{-7} . Построить график зависимости количества вычислений минимизируемой функции от десятичного логарифма задаваемой точности ε.
2. Реализовать алгоритм поиска интервала, содержащего минимум функции.
3. Реализовать метод Фибоначчи, сравнить его с методами дихотомии и золотого сечения.

Исследования

Метод дихотомии

i	x_1	x_2	$f(x_1)$	$f(x_2)$	a_i	b_i	$b_i - a_i$	$\frac{b_{i-1}-a_{i-1}}{b_i-a_i}$
1	9.00000000	9.00000001	1	1	−2.00000000	9.00000000	11	2
2	3.50000000	3.50000001	20.25	20.25	3.50000000	9.00000000	5.5	2
3	6.25000000	6.25000001	3.06	3.06	6.25000000	9.00000000	2.75	2
4	7.62500000	7.62500001	0.14	0.14	7.62500000	9.00000000	1.375	2
5	8.31250000	8.31250001	$9.77 \cdot 10^{-2}$	$9.77 \cdot 10^{-2}$	7.62500000	8.31250000	0.688	2
6	7.96875000	7.96875001	$9.77 \cdot 10^{-4}$	$9.77 \cdot 10^{-4}$	7.96875000	8.31250000	0.344	2
7	8.14062500	8.14062501	$1.98 \cdot 10^{-2}$	$1.98 \cdot 10^{-2}$	7.96875000	8.14062500	0.172	2
8	8.05468750	8.05468751	$2.99 \cdot 10^{-3}$	$2.99 \cdot 10^{-3}$	7.96875000	8.05468750	$8.594 \cdot 10^{-2}$	2
9	8.01171875	8.01171876	$1.37 \cdot 10^{-4}$	$1.37 \cdot 10^{-4}$	7.96875000	8.01171875	$4.297 \cdot 10^{-2}$	2
10	7.99023437	7.99023438	$9.54 \cdot 10^{-5}$	$9.54 \cdot 10^{-5}$	7.99023438	8.01171875	$2.148 \cdot 10^{-2}$	2
11	8.00097656	8.00097657	$9.54 \cdot 10^{-7}$	$9.54 \cdot 10^{-7}$	7.99023438	8.00097656	$1.074 \cdot 10^{-2}$	2
12	7.99560546	7.99560547	$1.93 \cdot 10^{-5}$	$1.93 \cdot 10^{-5}$	7.99560547	8.00097656	$5.371 \cdot 10^{-3}$	2
13	7.99829101	7.99829102	$2.92 \cdot 10^{-6}$	$2.92 \cdot 10^{-6}$	7.99829102	8.00097656	$2.686 \cdot 10^{-3}$	2
14	7.99963378	7.99963379	$1.34 \cdot 10^{-7}$	$1.34 \cdot 10^{-7}$	7.99963379	8.00097656	$1.343 \cdot 10^{-3}$	2
15	8.00030517	8.00030518	$9.31 \cdot 10^{-8}$	$9.31 \cdot 10^{-8}$	7.99963379	8.00030518	$6.714 \cdot 10^{-4}$	2
16	7.99996948	7.99996949	$9.32 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	7.99996948	8.00030518	$3.357 \cdot 10^{-4}$	2
17	8.00013732	8.00013733	$1.89 \cdot 10^{-8}$	$1.89 \cdot 10^{-8}$	7.99996948	8.00013733	$1.678 \cdot 10^{-4}$	2
18	8.00005340	8.00005341	$2.85 \cdot 10^{-9}$	$2.85 \cdot 10^{-9}$	7.99996948	8.00005341	$8.392 \cdot 10^{-5}$	2
19	8.00001144	8.00001145	$1.31 \cdot 10^{-10}$	$1.31 \cdot 10^{-10}$	7.99996948	8.00001144	$4.196 \cdot 10^{-5}$	2
20	7.99999046	7.99999047	$9.10 \cdot 10^{-11}$	$9.09 \cdot 10^{-11}$	7.99999046	8.00001144	$2.098 \cdot 10^{-5}$	2
21	8.00000095	8.00000096	$9.00 \cdot 10^{-13}$	$9.19 \cdot 10^{-13}$	7.99999046	8.00000095	$1.049 \cdot 10^{-5}$	2
22	7.99999570	7.99999571	$1.85 \cdot 10^{-11}$	$1.84 \cdot 10^{-11}$	7.99999571	8.00000095	$5.245 \cdot 10^{-6}$	2
23	7.99999833	7.99999834	$2.80 \cdot 10^{-12}$	$2.77 \cdot 10^{-12}$	7.99999833	8.00000095	$2.623 \cdot 10^{-6}$	2
24	7.99999964	7.99999965	$1.31 \cdot 10^{-13}$	$1.24 \cdot 10^{-13}$	7.99999964	8.00000095	$1.311 \cdot 10^{-6}$	2
25	8.00000029	8.00000030	$8.59 \cdot 10^{-14}$	$9.18 \cdot 10^{-14}$	7.99999964	8.00000030	$6.557 \cdot 10^{-7}$	2
26	7.99999997	7.99999998	$1.21 \cdot 10^{-15}$	$6.15 \cdot 10^{-16}$	7.99999997	8.00000030	$3.278 \cdot 10^{-7}$	2
27	8.00000013	8.00000014	$1.67 \cdot 10^{-14}$	$1.94 \cdot 10^{-14}$	7.99999997	8.00000013	$1.639 \cdot 10^{-7}$	2
28	8.00000005	8.00000006	$2.22 \cdot 10^{-15}$	$3.27 \cdot 10^{-15}$	7.99999997	8.00000005	$8.196 \cdot 10^{-8}$	2

Таблица 1: Решение методом дихотомии с константами $\varepsilon = 10^{-7}, \delta = 10^{-8}$.

Мы действительно приближаемся к минимуму, причём длина отрезка с каждой итерацией действительно уменьшается в 2 раза.

Метод золотого сечения

i	x_1	x_2	$f(x_1)$	$f(x_2)$	a_i	b_i	$b_i - a_i$	$\frac{b_{i-1}-a_{i-1}}{b_i-a_i}$
1	3.19349550	6.40325225	23.1	2.55	-2.00000000	11.59674775	13.597	1.62
2	6.40325225	8.38699101	2.55	0.15	3.19349550	11.59674775	8.403	1.62
3	8.38699101	9.61300899	0.15	2.6	6.40325225	11.59674775	5.193	1.62
4	7.62927023	8.38699101	0.14	0.15	6.40325225	9.61300899	3.21	1.62
5	7.16097303	7.62927023	0.7	0.14	6.40325225	8.38699101	1.984	1.62
6	7.62927023	7.91869381	0.14	$6.61 \cdot 10^{-3}$	7.16097303	8.38699101	1.226	1.62
7	7.91869381	8.09756743	$6.61 \cdot 10^{-3}$	$9.52 \cdot 10^{-3}$	7.62927023	8.38699101	0.758	1.62
8	7.80814384	7.91869381	$3.68 \cdot 10^{-2}$	$6.61 \cdot 10^{-3}$	7.62927023	8.09756743	0.468	1.62
9	7.91869381	7.98701745	$6.61 \cdot 10^{-3}$	$1.69 \cdot 10^{-4}$	7.80814384	8.09756743	0.289	1.62
10	7.98701745	8.02924378	$1.69 \cdot 10^{-4}$	$8.55 \cdot 10^{-4}$	7.91869381	8.09756743	0.179	1.62
11	7.96092014	7.98701745	$1.53 \cdot 10^{-3}$	$1.69 \cdot 10^{-4}$	7.91869381	8.02924378	0.111	1.62
12	7.98701745	8.00314648	$1.69 \cdot 10^{-4}$	$9.90 \cdot 10^{-6}$	7.96092014	8.02924378	$6.832 \cdot 10^{-2}$	1.62
13	8.00314648	8.01311476	$9.90 \cdot 10^{-6}$	$1.72 \cdot 10^{-4}$	7.98701745	8.02924378	$4.223 \cdot 10^{-2}$	1.62
14	7.99698574	8.00314648	$9.09 \cdot 10^{-6}$	$9.90 \cdot 10^{-6}$	7.98701745	8.01311476	$2.610 \cdot 10^{-2}$	1.62
15	7.99317819	7.99698574	$4.65 \cdot 10^{-5}$	$9.09 \cdot 10^{-6}$	7.98701745	8.00314648	$1.613 \cdot 10^{-2}$	1.62
16	7.99698574	7.99933893	$9.09 \cdot 10^{-6}$	$4.37 \cdot 10^{-7}$	7.99317819	8.00314648	$9.968 \cdot 10^{-3}$	1.62
17	7.99933893	8.00079328	$4.37 \cdot 10^{-7}$	$6.29 \cdot 10^{-7}$	7.99698574	8.00314648	$6.161 \cdot 10^{-3}$	1.62
18	7.99844009	7.99933893	$2.43 \cdot 10^{-6}$	$4.37 \cdot 10^{-7}$	7.99698574	8.00079328	$3.808 \cdot 10^{-3}$	1.62
19	7.99933893	7.99989444	$4.37 \cdot 10^{-7}$	$1.11 \cdot 10^{-8}$	7.99844009	8.00079328	$2.353 \cdot 10^{-3}$	1.62
20	7.99989444	8.00023777	$1.11 \cdot 10^{-8}$	$5.65 \cdot 10^{-8}$	7.99933893	8.00079328	$1.454 \cdot 10^{-3}$	1.62
21	7.99968226	7.99989444	$1.01 \cdot 10^{-7}$	$1.11 \cdot 10^{-8}$	7.99933893	8.00023777	$8.988 \cdot 10^{-4}$	1.62
22	7.99989444	8.00002558	$1.11 \cdot 10^{-8}$	$6.54 \cdot 10^{-10}$	7.99968226	8.00023777	$5.555 \cdot 10^{-4}$	1.62
23	8.00002558	8.00010663	$6.54 \cdot 10^{-10}$	$1.14 \cdot 10^{-8}$	7.99989444	8.00023777	$3.433 \cdot 10^{-4}$	1.62
24	7.99997549	8.00002558	$6.01 \cdot 10^{-10}$	$6.54 \cdot 10^{-10}$	7.99989444	8.00010663	$2.122 \cdot 10^{-4}$	1.62
25	7.99994453	7.99997549	$3.08 \cdot 10^{-9}$	$6.01 \cdot 10^{-10}$	7.99989444	8.00002558	$1.311 \cdot 10^{-4}$	1.62
26	7.99997549	7.99999463	$6.01 \cdot 10^{-10}$	$2.89 \cdot 10^{-11}$	7.99994453	8.00002558	$8.105 \cdot 10^{-5}$	1.62
27	7.99999463	8.00000645	$2.89 \cdot 10^{-11}$	$4.16 \cdot 10^{-11}$	7.99997549	8.00002558	$5.009 \cdot 10^{-5}$	1.62
28	7.99998732	7.99999463	$1.61 \cdot 10^{-10}$	$2.89 \cdot 10^{-11}$	7.99997549	8.00000645	$3.096 \cdot 10^{-5}$	1.62
29	7.99999463	7.99999914	$2.89 \cdot 10^{-11}$	$7.37 \cdot 10^{-13}$	7.99998732	8.00000645	$1.913 \cdot 10^{-5}$	1.62
30	7.99999914	8.00000193	$7.37 \cdot 10^{-13}$	$3.74 \cdot 10^{-12}$	7.99999463	8.00000645	$1.182 \cdot 10^{-5}$	1.62
31	7.99999742	7.99999914	$6.67 \cdot 10^{-12}$	$7.37 \cdot 10^{-13}$	7.99999463	8.00000193	$7.308 \cdot 10^{-6}$	1.62
32	7.99999914	8.00000021	$7.37 \cdot 10^{-13}$	$4.33 \cdot 10^{-14}$	7.99999742	8.00000193	$4.517 \cdot 10^{-6}$	1.62
33	8.00000021	8.00000087	$4.33 \cdot 10^{-14}$	$7.52 \cdot 10^{-13}$	7.99999914	8.00000193	$2.791 \cdot 10^{-6}$	1.62
34	7.99999980	8.00000021	$3.97 \cdot 10^{-14}$	$4.33 \cdot 10^{-14}$	7.99999914	8.00000087	$1.725 \cdot 10^{-6}$	1.62
35	7.99999955	7.99999980	$2.03 \cdot 10^{-13}$	$3.97 \cdot 10^{-14}$	7.99999914	8.00000021	$1.066 \cdot 10^{-6}$	1.62
36	7.99999980	7.99999996	$3.97 \cdot 10^{-14}$	$1.91 \cdot 10^{-15}$	7.99999955	8.00000021	$6.590 \cdot 10^{-7}$	1.62
37	7.99999996	8.00000005	$1.91 \cdot 10^{-15}$	$2.75 \cdot 10^{-15}$	7.99999980	8.00000021	$4.073 \cdot 10^{-7}$	1.62
38	7.99999990	7.99999996	$1.06 \cdot 10^{-14}$	$1.91 \cdot 10^{-15}$	7.99999980	8.00000005	$2.517 \cdot 10^{-7}$	1.62
39	7.99999996	7.99999999	$1.91 \cdot 10^{-15}$	$4.87 \cdot 10^{-17}$	7.99999990	8.00000005	$1.556 \cdot 10^{-7}$	1.62
40	7.99999999	8.00000002	$4.87 \cdot 10^{-17}$	$2.47 \cdot 10^{-16}$	7.99999996	8.00000005	$9.614 \cdot 10^{-8}$	1.62

Таблица 2: Решение методом золотого сечения с константами $\varepsilon = 10^{-7}, \delta = 10^{-8}$.

Мы действительно приближаемся к минимуму, причём длина отрезка с каждой итерацией действительно уменьшается в 1.62 раза ($1.62 \approx (\sqrt{5}+1)/2$).

Метод Фибоначчи

i	x_1	x_2	$f(x_1)$	$f(x_2)$	a_i	b_i	$b_i - a_i$	$\frac{b_{i-1}-a_{i-1}}{b_i-a_i}$
1	3.19349550	6.40325225	23.1	2.55	-2.00000000	11.59674775	13.597	1.62
2	6.40325225	8.38699101	2.55	0.15	3.19349550	11.59674775	8.403	1.62
3	8.38699101	9.61300899	0.15	2.6	6.40325225	11.59674775	5.193	1.62
4	7.62927023	8.38699101	0.14	0.15	6.40325225	9.61300899	3.21	1.62
5	7.16097303	7.62927023	0.7	0.14	6.40325225	8.38699101	1.984	1.62
6	7.62927023	7.91869381	0.14	$6.61 \cdot 10^{-3}$	7.16097303	8.38699101	1.226	1.62
7	7.91869381	8.09756743	$6.61 \cdot 10^{-3}$	$9.52 \cdot 10^{-3}$	7.62927023	8.38699101	0.758	1.62
8	7.80814384	7.91869381	$3.68 \cdot 10^{-2}$	$6.61 \cdot 10^{-3}$	7.62927023	8.09756743	0.468	1.62
9	7.91869381	7.98701745	$6.61 \cdot 10^{-3}$	$1.69 \cdot 10^{-4}$	7.80814384	8.09756743	0.289	1.62
10	7.98701745	8.02924378	$1.69 \cdot 10^{-4}$	$8.55 \cdot 10^{-4}$	7.91869381	8.09756743	0.179	1.62
11	7.96092014	7.98701745	$1.53 \cdot 10^{-3}$	$1.69 \cdot 10^{-4}$	7.91869381	8.02924378	0.111	1.62
12	7.98701745	8.00314648	$1.69 \cdot 10^{-4}$	$9.90 \cdot 10^{-6}$	7.96092014	8.02924378	$6.832 \cdot 10^{-2}$	1.62
13	8.00314648	8.01311476	$9.90 \cdot 10^{-6}$	$1.72 \cdot 10^{-4}$	7.98701745	8.02924378	$4.223 \cdot 10^{-2}$	1.62
14	7.99698574	8.00314648	$9.09 \cdot 10^{-6}$	$9.90 \cdot 10^{-6}$	7.98701745	8.01311476	$2.610 \cdot 10^{-2}$	1.62
15	7.99317819	7.99698574	$4.65 \cdot 10^{-5}$	$9.09 \cdot 10^{-6}$	7.98701745	8.00314648	$1.613 \cdot 10^{-2}$	1.62
16	7.99698574	7.99933893	$9.09 \cdot 10^{-6}$	$4.37 \cdot 10^{-7}$	7.99317819	8.00314648	$9.968 \cdot 10^{-3}$	1.62
17	7.99933893	8.00079328	$4.37 \cdot 10^{-7}$	$6.29 \cdot 10^{-7}$	7.99698574	8.00314648	$6.161 \cdot 10^{-3}$	1.62
18	7.99844009	7.99933893	$2.43 \cdot 10^{-6}$	$4.37 \cdot 10^{-7}$	7.99698574	8.00079328	$3.808 \cdot 10^{-3}$	1.62
19	7.99933893	7.99989444	$4.37 \cdot 10^{-7}$	$1.11 \cdot 10^{-8}$	7.99844009	8.00079328	$2.353 \cdot 10^{-3}$	1.62
20	7.99989444	8.00023777	$1.11 \cdot 10^{-8}$	$5.65 \cdot 10^{-8}$	7.99933893	8.00079328	$1.454 \cdot 10^{-3}$	1.62
21	7.99968226	7.99989444	$1.01 \cdot 10^{-7}$	$1.11 \cdot 10^{-8}$	7.99933893	8.00023777	$8.988 \cdot 10^{-4}$	1.62
22	7.99989444	8.00002558	$1.11 \cdot 10^{-8}$	$6.54 \cdot 10^{-10}$	7.99968226	8.00023777	$5.555 \cdot 10^{-4}$	1.62
23	8.00002558	8.00010663	$6.54 \cdot 10^{-10}$	$1.14 \cdot 10^{-8}$	7.99989444	8.00023777	$3.433 \cdot 10^{-4}$	1.62
24	7.99997549	8.00002558	$6.01 \cdot 10^{-10}$	$6.54 \cdot 10^{-10}$	7.99989444	8.00010663	$2.122 \cdot 10^{-4}$	1.62
25	7.99994453	7.99997549	$3.08 \cdot 10^{-9}$	$6.01 \cdot 10^{-10}$	7.99989444	8.00002558	$1.311 \cdot 10^{-4}$	1.62
26	7.99997549	7.99999463	$6.01 \cdot 10^{-10}$	$2.89 \cdot 10^{-11}$	7.99994453	8.00002558	$8.105 \cdot 10^{-5}$	1.62
27	7.99999463	8.00000645	$2.89 \cdot 10^{-11}$	$4.16 \cdot 10^{-11}$	7.99997549	8.00002558	$5.009 \cdot 10^{-5}$	1.62
28	7.99998732	7.99999463	$1.61 \cdot 10^{-10}$	$2.89 \cdot 10^{-11}$	7.99997549	8.00000645	$3.096 \cdot 10^{-5}$	1.62
29	7.99999463	7.99999914	$2.89 \cdot 10^{-11}$	$7.39 \cdot 10^{-13}$	7.99998732	8.00000645	$1.913 \cdot 10^{-5}$	1.62
30	7.99999914	8.00000193	$7.39 \cdot 10^{-13}$	$3.74 \cdot 10^{-12}$	7.99999463	8.00000645	$1.182 \cdot 10^{-5}$	1.62
31	7.99999742	7.99999914	$6.66 \cdot 10^{-12}$	$7.39 \cdot 10^{-13}$	7.99999463	8.00000193	$7.309 \cdot 10^{-6}$	1.62
32	7.99999914	8.00000021	$7.39 \cdot 10^{-13}$	$4.62 \cdot 10^{-14}$	7.99999742	8.00000193	$4.515 \cdot 10^{-6}$	1.62
33	8.00000021	8.00000086	$4.62 \cdot 10^{-14}$	$7.40 \cdot 10^{-13}$	7.99999914	8.00000193	$2.795 \cdot 10^{-6}$	1.62
34	7.99999979	8.00000021	$4.62 \cdot 10^{-14}$	$4.62 \cdot 10^{-14}$	7.99999914	8.00000086	$1.720 \cdot 10^{-6}$	1.62
35	7.99999957	7.99999979	$1.85 \cdot 10^{-13}$	$4.62 \cdot 10^{-14}$	7.99999914	8.00000021	$1.075 \cdot 10^{-6}$	1.6
36	7.99999979	8.00000000	$4.62 \cdot 10^{-14}$	$2.81 \cdot 10^{-20}$	7.99999957	8.00000021	$6.450 \cdot 10^{-7}$	1.67
37	8.00000000	8.00000000	$2.81 \cdot 10^{-20}$	$4.04 \cdot 10^{-20}$	7.99999979	8.00000021	$4.299 \cdot 10^{-7}$	1.5
38	7.99999979	8.00000000	$4.60 \cdot 10^{-14}$	$2.81 \cdot 10^{-20}$	7.99999979	8.00000000	$2.151 \cdot 10^{-7}$	2
39	8.00000000	7.99999979	$2.81 \cdot 10^{-20}$	$4.59 \cdot 10^{-14}$	7.99999979	8.00000000	$2.148 \cdot 10^{-7}$	1
40	7.99999957	8.00000000	$1.84 \cdot 10^{-13}$	$2.81 \cdot 10^{-20}$	7.99999979	7.99999979	$3.685 \cdot 10^{-10}$	582.78

Таблица 3: Решение методом Фибоначчи с константами $\varepsilon = 10^{-7}, \delta = 10^{-8}$.

Мы действительно приближаемся к минимуму, но длина отрезка на последних итерациях начинает изменяться нестабильно.

Сравнение методов

Везде наблюдается логарифмическая зависимость количества итераций от задаваемой точности. Метод Дихотомии показал себя наименее эффективно в плане количества вычислений для схождения к ответу, метод Фибоначчи и метод золотого сечения примерно одинаковы. Посмотреть результаты можно на рис.1.

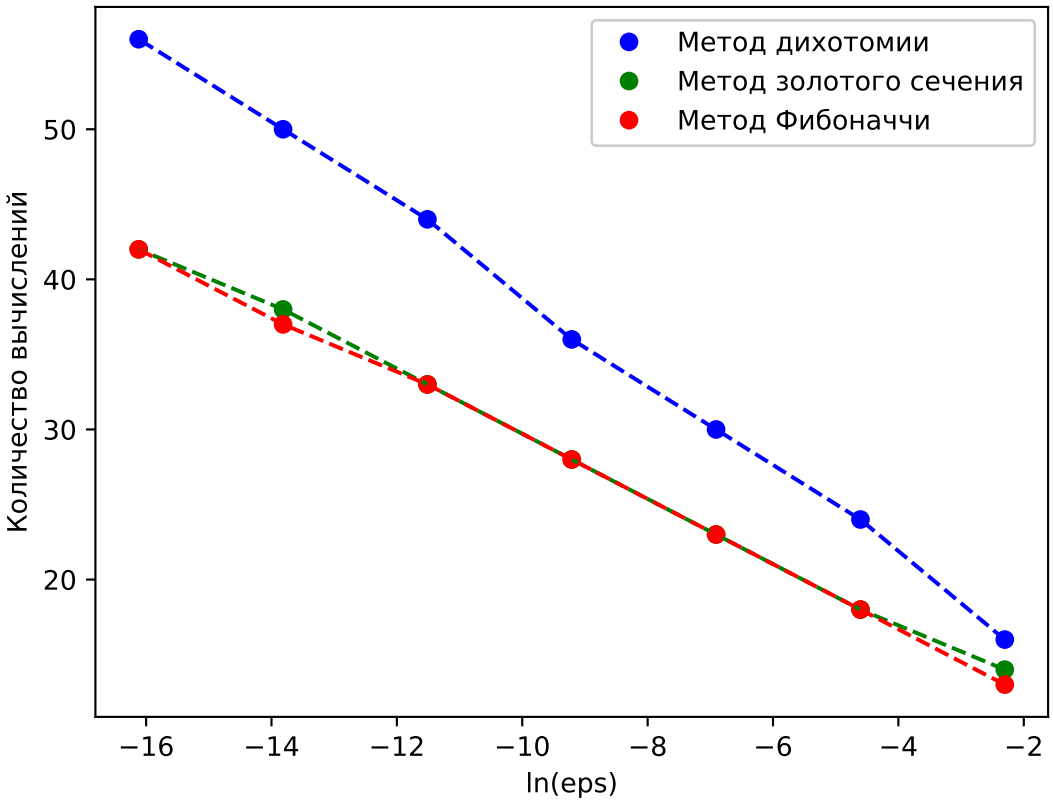

Рис. 1: График зависимости количества вычислений целевой функции от логарифма задаваемой точности ε.

Поиск интеравла с минимумом функции

i	x_i	$f(x_i)$
0	9.00000000	1.00000000
1	8.99999999	0.99999998
2	8.99999997	0.99999994
3	8.99999993	0.99999986
4	8.99999985	0.99999970
5	8.99999969	0.99999938
6	8.99999937	0.99999874
7	8.99999873	0.99999746
8	8.99999745	0.99999490

i	x_i	$f(x_i)$
9	8.99999489	0.99998978
10	8.99998977	0.99997954
11	8.99997953	0.99995906
12	8.99995905	0.99991810
13	8.99991809	0.99983619
14	8.99983617	0.99967237
15	8.99967233	0.99934477
16	8.99934465	0.99868973
17	8.99868929	0.99738030
18	8.99737857	0.99476401

i	x_i	$f(x_i)$
19	8.99475713	0.98954175
20	8.98951425	0.97913845
21	8.97902849	0.95849678
22	8.95805697	0.91787316
23	8.91611393	0.83926473
24	8.83222785	0.69260319
25	8.66445569	0.44150136
26	8.32891137	0.10818269

Таблица 4: Поиск интервала, содержащего минимум функциями с начальной точкой $x_0 = 10$ и $\delta = 10^{-8}$.

Мы выбрали начальное приближение $x_0 = 10$, в результате чего мы полчили ответ за 28 итераций. Это интервал $[9.328911, 7.315645]$.

Листинги

Сравнение методов

```
import matplotlib.pyplot as plt
from math import log, sqrt

# Our function
a0, b0 = -2., 20.
f = lambda x: (x - 8) ** 2

# Dictionary for building plot
data = {'dichotomy': {'f_count': [], 'eps': []},
        'golden_ratio': {'f_count': [], 'eps': []},
        'fibonacci': {'f_count': [], 'eps': []}}

# ------------------------------------------------------------
# Dichotomy
# ------------------------------------------------------------
print('Метод дихотомии:')
for eps in [10 ** i for i in range(-1, -8, -1)]:
    delta = eps / 10.
    ai, bi = a0, b0
    i = 0
    fc = 0 # Count function compution

    # Save data with different epsilon in different files
    with open('tests/dichotomy, {:4.0e}.csv'.format(eps), 'w') as file:
        file.write('\t'.join(['i', 'x', 'y', 'fx', 'fy', 'ai', 'bi',
        ↪  'length', 'proportion']) + '\n')

        # While length is greater then epsilon
        while bi - ai > eps:
            x1 = (ai + bi - delta) / 2.
            x2 = (ai + bi + delta) / 2.

            f_x1 = f(x1)
            f_x2 = f(x2)

            ci = (ai + bi) / 2.
            residual = bi - ai

            # Very strange if-block. Wee can't compare floats
            if f_x1 < f_x2:
                bi = ci

            elif f_x1 > f_x2:
                ai = ci

            else:
```

```
                ai = x1
                bi = x2

            i += 1
            fc += 2

            print(ai, bi)

            # Saving data
            file.write('\t'.join(map(str, [i, x1, x2, f_x1, f_x2, ai, bi,
             ↪  bi - ai, residual / (bi - ai) ])) + '\n')
        print(i, '[{}, {}], x = {}'.format(x1, x2, (x1 + x2) / 2.))

    # Saving data for plot
    data['dichotomy']['f_count'].append(fc)
    data['dichotomy']['eps'].append(log(eps))
print()


# ——————————————————————————————————————————————————————————
# Golden ratio
# ——————————————————————————————————————————————————————————
print('Метод золотого сечения:')
for eps in [10 ** i for i in range(-1, -8, -1)]:
    delta = eps / 10.
    ai, bi = a0, b0
    i = 0
    fc = 0 # Count function compution

    x1 = ai + (3. - sqrt(5.)) * (bi - ai) / 2.
    x2 = ai + (sqrt(5.) - 1.) * (bi - ai) / 2.

    f_x1 = f(x1)
    f_x2 = f(x2)
    fc += 2

    # Save data with different epsilon in different files
    with open('tests/golden_ratio, {:4.0e}.csv'.format(eps), 'w') as file:
        file.write('\t'.join(['i', 'x', 'y', 'fx', 'fy', 'ai', 'bi',
         ↪  'length', 'proportion']) + '\n')

        # While length is greater then epsilon
        while bi - ai > eps:
            residual = bi - ai

            if f_x1 > f_x2:
                ai = x1
                x1 = x2
                x2 = ai + (sqrt(5.) - 1.) * (bi - ai) / 2.
```

```
                f_x1 = f_x2
                f_x2 = f(x2)

            else:
                bi = x2
                x2 = x1
                x1 = ai + (3. - sqrt(5.)) * (bi - ai) / 2.
                f_x2 = f_x1
                f_x1 = f(x1)

            i += 1
            fc += 1

            # Saving data
            file.write('\t'.join(map(str, [i, x1, x2, f_x1, f_x2, ai, bi,
             ↪  bi - ai, residual / (bi - ai) ])) + '\n')
        print(i, '[{}, {}], x = {}'.format(x1, x2, (x1 + x2) / 2.))

    # Saving data for plot
    data['golden_ratio']['f_count'].append(fc)
    data['golden_ratio']['eps'].append(log(eps))
print()

# ——————————————————————————————————————————————————————————
# Fibonacci
# ——————————————————————————————————————————————————————————
print('Метод Фибоначчи:')
for eps in [10 ** i for i in range(-1, -8, -1)]:
    F = lambda n: (((1. + sqrt(5.)) / 2.) ** n - ((1. - sqrt(5.)) / 2.) **
     ↪  n) / sqrt(5.)
    delta = eps / 10.
    ai, bi = a0, b0
    i = 0 # Iter
    n = 0 # Max iterations
    fc = 0 # Count function compution

    while (b0 - a0) / eps ⩾ F(n + 2):
        n += 1

    x1 = ai + F(n-2) * (bi - ai) / F(n)
    x2 = ai + F(n-1) * (bi - ai) / F(n)

    f_x1 = f(x1)
    f_x2 = f(x2)
    fc += 2

    # Save data with different epsilon in different files
    with open('tests/fibonacci, {:4.0e}.csv'.format(eps), 'w') as file:
        file.write('\t'.join(['i', 'x', 'y', 'fx', 'fy', 'ai', 'bi',
         ↪  'length', 'proportion']) + '\n')
```

```
140
141         # While length is greater then epsilon
142         for k in range(n):
143             residual = bi - ai
144
145             # Algorithm from Wikipedia
146             if f_x1 > f_x2:
147                 ai = x1
148                 x1 = x2
149                 x2 = bi - (x1 - ai)
150                 f_x1 = f_x2
151                 f_x2 = f(x2)
152
153             else:
154                 bi = x2
155                 x2 = x1
156                 x1 = ai + (bi - x2)
157                 f_x2 = f_x1
158                 f_x1 = f(x1)
159
160             #if f_x1 < f_x2:
161             #    bi = x2
162             #    x2 = x1
163             #    x1 = ai + F(n-i+1) * (bi - ai) / F(n-i+3)
164             #    f_x2 = f_x1
165             #    f_x1 = f(x1)
166
167             #else:
168             #    ai = x1
169             #    x1 = x2
170             #    x2 = ai + F(n-i+2) * (bi - ai) / F(n-i+3)
171             #    f_x1 = f_x2
172             #    f_x2 = f(x2)
173
174             i += 1
175             fc += 1
176
177             # Saving data
178             file.write('\t'.join(map(str, [i, x1, x2, f_x1, f_x2, ai, bi,
                ↪  bi - ai, residual / (bi - ai) ])) + '\n')
179         print(i, '[{}, {}], x = {}'.format(x1, x2, (x1 + x2) / 2.))
180
181     # Saving data for plot
182     data['fibonacci']['f_count'].append(fc)
183     data['fibonacci']['eps'].append(log(eps))
184 print()
185
186 # Build and save plot as file
187 line1, = plt.plot(data['dichotomy']['eps'], data['dichotomy']['f_count'],
    ↪  'bo')
```

```
line2, = plt.plot(data['golden_ratio']['eps'], data['golden_ratio']['f_count'], 'go')
line3, = plt.plot(data['fibonacci']['eps'], data['fibonacci']['f_count'], 'ro')
plt.plot(data['dichotomy']['eps'], data['dichotomy']['f_count'], 'b--',
        data['golden_ratio']['eps'], data['golden_ratio']['f_count'], 'g--',
        data['fibonacci']['eps'], data['fibonacci']['f_count'], 'r--')
plt.xlabel('ln(eps)')
plt.ylabel('Количество вычислений')
plt.legend([line1, line2, line3], ['Метод дихотомии', 'Метод золотого сечения', 'Метод Фибоначчи'])

plt.savefig('plot.eps', format='eps')
plt.savefig('plot.png')

print("Done")
```

Поиск интервала

```
# Our function
f = lambda x: (x - 8) ** 2

x0 = float(input('Write x0: '))
dx = 1e-8

xk1 = x0
h = dx if f(xk1) > f(xk1 + dx) else -dx

xk0 = x0
xk2 = xk1 + h

i = 0

with open('interval_search.csv', 'w') as file:
    file.write('\t'.join(['i', 'xi', 'f_xi']) + '\n')

    f_xk1 = f(xk1)
    f_xk2 = f(xk2)

    while f_xk1 > f_xk2:
        file.write('\t'.join(map(str, [i, xk1, f(xk1)])) + '\n')

        h *= 2.
        xk0 = xk1
        xk1 = xk2
        xk2 = xk1 + h

```

```
        f_xk1 = f_xk2
        f_xk2 = f(xk2)
        i += 1

    file.write('\t'.join(map(str, [i, xk1, f(xk1)])) + '\n')


if xk0 > xk2:
    xk0, xk2 = xk2, xk0

print('Done: [%f, %f]' % (xk0, xk2))
```