

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» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

Отчёт

по лабораторной работе № 1 по дисциплине «Теория систем и системный анализ»

Тема: «Исследование методов прямого поиска экстремума унимодальной функции одного переменного»

Вариант 6

Выполнил: Калинин Д. В., студент группы ИУ8-31

Проверил: Коннова Н.С., доцент каф. ИУ8

1. Цель работы

Исследовать функционирование и провести сравнительный анализ различных алгоритмов прямого поиска экстремума (пассивный поиск, метод дихотомии, золотого сечения, Фибоначчи) на примере унимодальной функции одного переменного.

2. Условие задачи

На интервале [-5; 2] задана унимодальная функция одного переменного $f(x) = (1-x)^2 + e^x$. Используя метод Фибоначчи, найти интервал нахождения минимума f(x) с заданным количеством итераций. Провести сравнение с методом оптимального пассивного поиска. Результат, в зависимости от числа точек разбиения N, представить в виде таблицы.

3. Ход работы

Для наглядности построим график заданной функции и определим местонахождение её минимума:

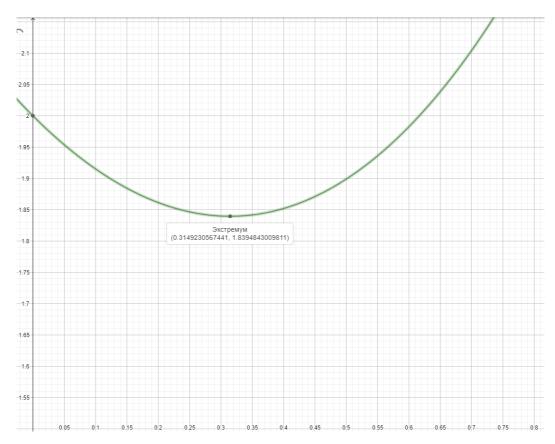


Рисунок 1 – График исследуемой функции

Как видно из графика, функция достигает своего минимума в точке $x \approx 0.3149$. Теперь проведём программный расчет при помощи методов оптимального пассивного поиска и Фибоначчи.

Результаты этого расчёта представлены в таблицах 1 и 2:

Таблица 1 – результат работы метода пассивного поиска

Convector Todes			
0	Количество точек (N)	Точка минимума	
0	າ	α 2222221 2222221	
0			
5 0.8333333±1.1666667 6 0.00000001.0000000 7 0.2500000±0.8750000 8 0.44444440.7777778 9 0.6000000±0.7000000 10 0.999091±0.6363636 11 0.2500000±0.5833333 12 0.3846154±0.5384615 13 0.5000000±0.5000000 14 0.1333333±0.4666667 15 0.2500000±0.4375000 16 0.3529412±0.4117647 17 0.4444444±3.888889 18 0.1578947±0.3684211 19 0.2500000±0.3500000 20 0.3333333±0.3333333 21 0.409099±0.3181818 22 0.1739130±0.3043478 23 0.2500000±0.2916667 24 0.3200000±0.2800000 25 0.3846154±0.2692308 26 0.1851852±0.2592593 27 0.2500000±0.2187500 28 0.3103448±0.2413793 30 0.4193548±0.2258065 31 0.2500000±0.2187500 32 0.336393±0.2121212			
6 0.00000001.00000000000000000000000000			
7 0.2500000±0.8750000 8 0.4444444410.777778 9 0.600000001 10 0.909091±0.6363636 11 0.2500000±0.5833333 12 0.3846154±0.5384615 13 0.500000±0.5000000 14 0.133333±0.4666667 15 0.2500000±0.4750000 16 0.3529412±0.4117647 17 0.4444444±0.3888889 18 0.1578947±0.3684211 19 0.250000±0.3500000 20 0.333333±0.333333 21 0.409009±0.3181818 22 0.1739130±0.3043478 23 0.250000±0.2916667 24 0.320000±0.2916667 25 0.3846154±0.2692308 26 0.1851852±0.2592593 27 0.2500000±0.2333333 30 0.4193548±0.2258065 31 0.2500000±0.2187500 28 0.3103448±0.2413793 29 0.3666667±0.2333333 30 0.4193548±0.2528065 31 0.2500000±0.2187500 32 0.30303±0.2121212 33 0.3529412±0.205824 34 0.400000±0.2000000 35 0.2502000±0.1944444 36 0.2972973±0.1891892 37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333+0.1666667 42 0.3720930±0.1555556 43 0.2502000±0.1590000 44 0.2888889±0.1555556 45 0.326870±0.1555556 46 0.3617021±0.1489362 47 0.2500000±0.1488333 48 0.2857143±0.1428571 49 0.320000±0.1440000	:	:	
8 0.444444±0.777778 9 0.600000001.70000000 10 0.909091±0.6363636 11 0.2500000±0.5833333 12 0.384615±0.5384615 13 0.5000000±0.5803033 14 0.133333±0.4666667 15 0.2500000±0.4375000 16 0.3529412±0.4117647 17 0.444444±0.3888889 18 0.1578947±0.3684211 19 0.2500000±0.3500000 20 0.333333±0.333333 21 0.409000±0.3181818 22 0.1739130±0.3383333 21 0.409000±0.3181818 22 0.1739130±0.3043478 23 0.5500000±0.2800000 25 0.3846154±0.2692308 26 0.1851852±0.2592593 27 0.2500000±0.2500000 28 0.3103448±0.2413793 30 0.4193548±0.2258065 31 0.2506000±0.2187500 32 0.30303±0.212121 33 0.3529412±0.2058824 34 0.400000±0.2000000 35 0.2500000±0.1887500 36 0.972973±0.1891892 37 0.364654±0.1794872 39 0.5500000±0.17500000 40 0.2926829±0.1707317 41 0.3333333±0.1666667 42 0.3720930±0.1555556 43 0.2500000±0.15590909 44 0.2888889±0.15555556 45 0.360000±0.1458333 46 0.2500000±0.15590909 47 0.2500000±0.1458333 48 0.25143±0.1428571 49 0.320000±0.1400000		:	
9		:	
10	:		
11	:		
12 0.3846154±0.5384615 13 0.5000000±0.50000000 14 0.133333±0.4666667 15 0.2500000±0.4375000 16 0.3529412±0.4117647 17 0.4444444±0.3888889 18 0.1578947±0.3668211 19 0.2500000±0.3500000 20 0.333333±0.333333 21 0.4090909±0.3181818 22 0.1739130±0.3043478 23 0.250000±0.2916667 24 0.3200000±0.2916667 24 0.3200000±0.292088 25 0.3846154±0.2692308 26 0.1851852±0.2592593 27 0.2500000±0.2500000 28 0.3103448±0.2413793 29 0.366667±0.2333333 30 0.4193548±0.2258065 31 0.2500000±0.2187500 32 0.30303±0.212121 33 0.3529412±0.2698824 34 0.400000±0.2000000 35 0.2500000±0.1944444 36 0.2972973±0.1891892 37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.1666667 42 0.3720930±0.1627907 43 0.2500000±0.1550500 44 0.2888889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2887143±0.1428571 49 0.3200000±0.1400000		:	
13 0.5000000±0.50000000 14 0.1333333±0.4666667 15 0.2500000±0.43750000 16 0.3529412±0.4117647 17 0.44444440.3888889 18 0.1578947±0.3684211 19 0.2500000±0.35000000 20 0.333333±0.3333333 21 0.4090000±0.31818181 22 0.1739130±0.3043478 23 0.2500000±0.2940667 24 0.3200000±0.2940667 24 0.3200000±0.2800000 25 0.3846154±0.2692308 26 0.1851852±0.2592593 27 0.2500000±0.24137500 28 0.3103448±0.2413793 29 0.366667±0.2333333 30 0.4193548±0.2258065 31 0.2500000±0.2187500 32 0.3030303±0.2121212 33 0.3529412±0.2058824 34 0.4000000±0.1944444 36 0.2972973±0.1881892 37 0.3421053±0.1842105 38 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.33333333+0.1666667 42 0.3720930±0.1627907 43 0.288889±0.1555556 45 0.3260000±0.1590909 44 0.2888889±0.15555556 45 0.3260000±0.158333 46 0.2857143±0.1428571 49 0.3200000±0.1440000		:	
14		:	
15 0.2500000±0.4375000 16 0.3529412±0.4117647 17 0.444444±0.3888889 18 0.1578947±0.3684211 19 0.2500000±0.3500000 20 0.3333333±0.33333333 21 0.4090909±0.3181818 22 0.1739130±0.3043478 23 0.2500000±0.2916667 24 0.3200000±0.2916667 24 0.3200000±0.2916667 25 0.3846154±0.2692308 26 0.1851852±0.2592593 27 0.2500000±0.25905000 28 0.3103448±0.2413793 29 0.3666667±0.2333333 30 0.4193548±0.2258065 31 0.2500000±0.2187500 32 0.303039±0.2121212 33 0.3529412±0.2658824 4 0.400000±0.2187500 35 0.2500000±0.1944444 36 0.2972973±0.1891892 37 0.3421653±0.1842105 38 0.3421653±0.1842105 38 0.3421653±0.1842105 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.1666667 42 0.3720930±0.1527907 43 0.2500000±0.1559556 45 0.3260870±0.1521739 46 0.2888889±0.1555556 47 0.2888889±0.1555556 48 0.2857143±0.14828571 49 0.3200000±0.1400000	13	0.5000000±0.5000000	
16	14	0.1333333±0.4666667	
17 0.44444440.3888889 18 0.1578947±0.3684211 19 0.2500000±0.3500000 20 0.333333±0.33333333 21 0.4090909±0.3181818 22 0.1739130±0.3043478 23 0.5500000±0.2916667 24 0.3200000±0.2800000 25 0.3846154±0.2692308 26 0.1851852±0.2592593 27 0.2500000±0.2500000 28 0.3103448±0.2413793 29 0.3666667±0.2333333 30 0.4193548±0.2258065 31 0.2500000±0.2187500 32 0.303033±0.2121212 33 0.3529412±0.2058824 34 0.400000±0.1944444 36 0.2972973±0.1891892 37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.1666667 42 0.3720930±0.1627907 43 0.2500000±0.1590909 44 0.2888889±0.1555556 45 0.3617021±0.1489362 47 0.2500000±0.1440000	15	0.2500000±0.4375000	
18	16	0.3529412±0.4117647	
19	17	0.444444±0.3888889	
20	18	0.1578947±0.3684211	
21	19	0.2500000±0.3500000	
22 0.1739130±0.3043478 23 0.2500000±0.2916667 24 0.3200000±0.2800000 25 0.3846154±0.2692308 26 0.1851852±0.2592593 27 0.2500000±0.2500000 28 0.3103448±0.2413793 29 0.3666667±0.2333333 30 0.4193548±0.2258065 31 0.2500000±0.2187500 32 0.303003±0.2121212 33 0.3529412±0.2058824 34 0.400000±0.2000000 35 0.2500000±0.1944444 36 0.2972973±0.1891892 37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.1666667 42 0.3720930±0.1627907 43 0.2500000±0.1590909 44 0.2888889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1440000	20	0.3333333±0.3333333	
23	21	0.4090909±0.3181818	
24 0.3200000±0.2800000 25 0.3846154±0.2692308 26 0.1851852±0.2592593 27 0.2500000±0.2500000 28 0.3103448±0.2413793 29 0.3666667±0.23333333 30 0.4193548±0.2258065 31 0.2500000±0.2187500 32 0.303303±0.2121212 33 0.3529412±0.2058824 34 0.4000000±0.2000000 35 0.25500000±0.19444444 36 0.2972973±0.1891892 37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.16666667 42 0.3720930±0.1627907 43 0.2500000±0.1550909 44 0.2888889±0.1555556 45 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000	22	0.1739130±0.3043478	
24 0.3200000±0.2800000 25 0.3846154±0.2692308 26 0.1851852±0.2592593 27 0.2500000±0.2500000 28 0.3103448±0.2413793 29 0.3666667±0.23333333 30 0.4193548±0.2258065 31 0.2500000±0.2187500 32 0.303303±0.2121212 33 0.3529412±0.2058824 34 0.4000000±0.2000000 35 0.25500000±0.19444444 36 0.2972973±0.1891892 37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.16666667 42 0.3720930±0.1627907 43 0.2500000±0.1550909 44 0.2888889±0.1555556 45 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000	23 İ	0.2500000±0.2916667	
25 0.3846154±0.2692308 26 0.1851852±0.2592593 27 0.2500000±0.2500000 28 0.3103448±0.2413793 29 0.3666667±0.2333333 30 0.4193548±0.2258065 31 0.2500000±0.2187500 32 0.303030±0.2121212 33 0.3529412±0.2058824 34 0.400000±0.2000000 35 0.2500000±0.1944444 36 0.2972973±0.1891892 37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1767317 41 0.3333333±0.1666667 42 0.3720930±0.1627907 43 0.2500000±0.1599099 44 0.288889±0.1555556 45 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000			
26 0.1851852±0.2592593 27 0.2500000±0.2500000 28 0.3103448±0.2413793 29 0.3666667±0.2333333 30 0.4193548±0.2258065 31 0.2500000±0.2187500 32 0.330303±0.2121213 33 0.3529412±0.2058824 34 0.400000±0.2000000 35 0.2500000±0.1944444 36 0.2972973±0.1891892 37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.1666667 42 0.3720930±0.1627907 43 0.2500000±0.1590909 44 0.288889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000		:	
27	:	:	
28		:	
29	:	:	
30		:	
31		:	
32 0.303033±0.2121212 33 0.3529412±0.2058824 34 0.400000±0.2000000 35 0.2500000±0.1944444 36 0.2972973±0.1891892 37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.16666667 42 0.3720930±0.1627907 43 0.2500000±0.1590909 44 0.2888889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.32000000±0.1400000			
33 0.3529412±0.2058824 34 0.400000±0.2000000 35 0.2500000±0.1944444 36 0.2972973±0.1891892 37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.1666667 42 0.3720930±0.1627907 43 0.2500000±0.1590909 44 0.2888889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000		:	
0.400000±0.2000000 35		!	
0.2500000±0.1944444 36			
36 0.2972973±0.1891892 37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.1666667 42 0.3720930±0.1627907 43 0.2500000±0.1590909 44 0.2888889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000			
37 0.3421053±0.1842105 38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.1666667 42 0.3720930±0.1627907 43 0.2500000±0.1590909 44 0.2888889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000			
38 0.3846154±0.1794872 39 0.2500000±0.1750000 40 0.2926829±0.1707317 41 0.3333333±0.1666667 42 0.3720930±0.1627907 43 0.2500000±0.1590909 44 0.2888889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000			
39	· · · · · · · · · · · · · · · · · · ·	:	
40 0.2926829±0.1707317 41 0.3333333±0.1666667 42 0.3720930±0.1627907 43 0.2500000±0.1590909 44 0.2888889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000	· · · · · · · · · · · · · · · · · · ·	:	
41 0.3333333±0.1666667 42 0.3720930±0.1627907 43 0.2500000±0.1590909 44 0.2888889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000		:	
42 0.3720930±0.1627907 43 0.2500000±0.1590909 44 0.2888889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000		:	
0.2500000±0.1590909		:	
44 0.2888889±0.1555556 45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000			
45 0.3260870±0.1521739 46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000			
46 0.3617021±0.1489362 47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000		0.2888889±0.1555556	
47 0.2500000±0.1458333 48 0.2857143±0.1428571 49 0.3200000±0.1400000	45	0.3260870±0.1521739	
48 0.2857143±0.1428571 49 0.3200000±0.1400000	46	0.3617021±0.1489362	
49 0.3200000±0.1400000	47	0.2500000±0.1458333	
49 0.3200000±0.1400000	48	0.2857143±0.1428571	
;	49	0.3200000±0.1400000	
·		0.3529412±0.1372549	
51 0.2500000±0.1346154			
52 0.2830189±0.1320755		:	
53 0.3148148±0.1296296		:	
54 0.3454545±0.1272727		:	
55 0.3750000±0.1250000		:	
56 0.2807018±0.1228070			

57	0.3103448±0.1206897
58	0.3389831±0.1186441
	!!
59	0.3666667±0.1166667
60	0.2786885±0.1147541
61	0.3064516±0.1129032
	!
62	0.3333333±0.1111111
63	0.3593750±0.1093750
64	0.2769231±0.1076923
:	!
65	0.3030303±0.1060606
66	0.3283582±0.1044776
67	0.3529412±0.1029412
!	!
68	0.2753623±0.1014493
69	0.3000000±0.1000000
70	0.3239437±0.0985915
	:
71	0.3472222±0.0972222
72	0.2739726±0.0958904
İ73	0.2972973±0.0945946
! -	!
74	0.3200000±0.0933333
75	0.3421053±0.0921053
İ76	0.2727273±0.0909091
·	!
77	0.2948718±0.0897436
78	0.3164557±0.0886076
İ79	0.3375000±0.0875000
:	0.2716049±0.0864198
80	!
81	0.2926829±0.0853659
82	0.3132530±0.0843373
83	0.333333±0.0833333
	:
84	0.3529412±0.0823529
85	0.2906977±0.0813953
86	0.3103448±0.0804598
:	:
87	0.3295455±0.0795455
88	0.3483146±0.0786517
89	0.2888889±0.0777778
	1
90	0.3076923±0.0769231
91	0.3260870±0.0760870
92	0.3440860±0.0752688
193	:
! =	0.2872340±0.0744681
94	0.3052632±0.0736842
95	0.3229167±0.0729167
•	:
96	0.3402062±0.0721649
97	0.2857143±0.0714286
98	0.3030303±0.0707071
99	0.3200000±0.0700000
:	:
100	0.3366337±0.0693069
101	0.2843137±0.0686275
102	0.3009709±0.0679612
:	:
103	0.3173077±0.0673077
104	0.3333333±0.0666667
105	0.2830189±0.0660377
	! !
106	0.2990654±0.0654206
107	0.3148148±0.0648148
108	0.3302752±0.0642202
	:
109	0.3454545±0.0636364
110	0.2972973±0.0630631
1111	0.3125000±0.0625000
1112	0.3274336±0.0619469
	:
1113	0.3421053±0.0614035
114	0.2956522±0.0608696
1115	0.3103448±0.0603448
!	:
116	0.3247863±0.0598291
117	0.3389831±0.0593220
·	0.2941176±0.0588235
118	1 0.29411/070.00882301

119	0.3083333±0.0583333	
120	0.3223140±0.0578512	
121	0.3360656±0.0573770	
122	0.2926829±0.0569106	
123	0.3064516±0.0564516	
124	0.3200000±0.0560000	
125	0.3333333±0.0555556	
126	0.2913386±0.0551181	
127	0.3046875±0.0546875	
128	0.3178295±0.0542636	
129	0.3307692±0.0538462	
130	0.2900763±0.0534351	
131	0.3030303±0.0530303	
132	0.3157895±0.0526316	
133	0.3283582±0.0522388	
134	0.3407407±0.0518519	
135	0.3014706±0.0514706	
136	0.3138686±0.0510949	
137	0.3260870±0.0507246	
138	0.3381295±0.0503597	
139	0.3000000±0.0500000	

Таблица 2 – результат работы метода Фибоначчи

Количество точек (N) Точка минимума
1	-0.1631190±2.1631190
2	0.6631190±1.3368810
3	0.1524758±0.8262379
4	0.4680706±0.5106431
5	0.2730223±0.3155948
6	0.3935688±0.1950483
7	0.3190670±0.1205465
8	0.2730223±0.0745018
9	0.3014795±0.0460447

Построим график зависимостей погрешности от числа точек N (для оптимального пассивного поиска).

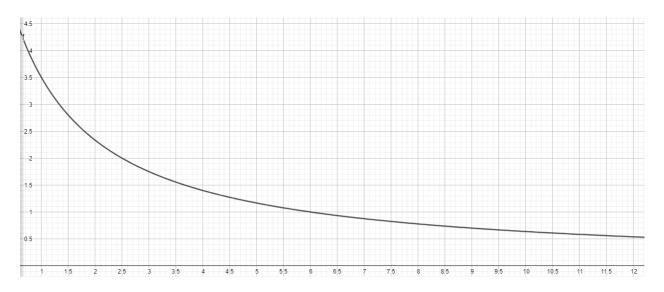


Рисунок 2 — График зависимости погрешности от числа точек для оптимального пассивного поиска

Ссылка на репозиторий с выполненной работой: https://github.com/shreddered/lab-01

4. Выводы

Данный эксперимент показал, что метод Фибоначчи эффективнее метода оптимально пассивного поиска при нахождении экстремума унимодальной функции одного переменного.

Приложение 1. Исходный код программы

Файл source/app.d

```
import algorithms;
import std.math : exp;
import std.stdio : writeln, writefln;
alias fun = (x) => (1 - x) * (1 - x) + exp(x);
int main() {
    writeln("Таблица 1 -- поиск минимума методом оптимального пассивного поиска");
    ISearcher searcher = new OptimalPassiveSearcher!fun;
    searcher.setInterval(-5, 2);
    searcher.search(0.1);
    writeln;
    writeln("Таблица 2 -- поиск минимума методом Фибоначчи");
    searcher = new FibonacciSearcher!fun;
    searcher.setInterval(-5, 2);
    searcher.search(0.1);
    return 0;
}
```

Файл source/algorithms/searcher.d

```
module algorithms.searcher;
interface ISearcher {
   public void setInterval(in double a, in double b);
   public void search(const double eps);
}
```

Файл source/algorithms/optimal_search.d

```
module algorithms.optimal search;
import algorithms.searcher;
import std.algorithm : map, min, minIndex;
import std.array : array;
import std.math : exp;
import std.range : iota;
import std.stdio : writeln, writefln;
class OptimalPassiveSearcher(alias func) : ISearcher {
   private double a, b;
   private double delta(in ulong n) @safe pure nothrow {
       return (_b - _a) / (n + 1);
   public override void setInterval(in double a, in double b) {
       a = a;
       _b = b;
   public override void search(const double eps = 0.1) {
       ulong n;
       writefln("|%-20s|%-20s|", "Количество точек (N)", "Точка минимума");
       writeln("|-----|");
       double ans;
       for (n = 2; delta(n) * 2 >= eps; ++n) {
           double[] x = iota(1, n + 1)
               .map! (k \Rightarrow delta(n) * k + _a)
               .array;
           ulong index = x.map!func
               .minIndex;
           ans = x[index];
           writefln("|%-20d|%- 3.7f±%-3.7f|", n, ans, delta(n));
       writeln("|-----|");
       writefln!"x = %3.7f\pm %3.7f"(ans, delta(n - 1));
```

Файл source/algorithms/fibonacci.d

```
module algorithms.fibonacci;
import algorithms.searcher;
import std.array : array;
import std.range : recurrence, take;
import std.stdio : writeln, writefln;
class FibonacciSearcher(alias func) : ISearcher {
   private double a, b;
   private pragma(inline) double interval() const @safe pure nothrow {
       return b - a;
   public override void setInterval(in double a, in double b) {
        a = a;
       b = b;
   public override void search(const double eps = 0.1) {
       // first N fibonacci numbers will be evaluated at compile time
       enum ulong N = 30;
       enum fib = recurrence!"a[n - 1] + a[n - 2]"(1, 1)
           .take(N + 3)
           .array;
       // table header
       writefln!"N = %d"(N);
       writefln("|%-20s|%-20s|", "Количество точек (N)", "Точка минимума");
       writeln("|-----|");
       // preparations
       double x1 = a + interval() * fib[N] / fib[N + 2],
             x2 = _a + _b - x1;
       double y1 = \overline{func(x1)}, y2 = func(x2);
       for (ulong k = 0; k != N && interval() >= eps; ++k) {
           if (y1 > y2) {
               _a = x1;
               x1 = x2;
               x2 = _a + _b - x1;
               y1 = y2;
              y2 = func(x2);
           else {
               b = x2;
               x2 = x1;
               x1 = _a + _b - x2;
              y2 = y1;
               y1 = func(x1);
           writefln!"|\$-20d|\$-3.7f\pm\$-3.7f|"(k + 1, (a + b) / 2,
interval() / 2);
       writeln("|-----|");
       writefln!"x = 3.7f\pm3.7f"((a + b) / 2, interval() / 2);
```

Файл source/algorithms/package.d

```
module algorithms;

public import algorithms.fibonacci;
public import algorithms.optimal_search;
public import algorithms.searcher;
```