|  |  |
| --- | --- |
|  | Московский Государственный Технический Университет им. Н.Э.Баумана  Факультет ИУ «Информатика и системы управления»  Кафедра ИУ8 «Информационная Безопасность» |

Отчёт по лабораторной работе №1

«Исследование методов прямого поиска экстремума унимодальной функции одного переменного»

по дисциплине

«Теория систем и системный анализ»

Выполнил: Т.А. Орбелян

Проверил: И.С.Строганов

Группа: ИУ8-11М

Вариант: 6

Оценка:

Москва 2021

**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.

**Постановка задачи:**

На интервале [-5, 2] задана унимодальная функция одного переменного f(x) = (1 - x)­­2 + ex. Используя метод Фибоначчи, найти интервал нахождения минимума f(x) при заданной наибольшей допустимой длине интервала неопределенности ε = 0,1. Провести сравнение с методом оптимального пассивного поиска. Результат, в зависимости от числа точек разбиения N, представить в виде таблицы.

**Практическая часть:**

Программы написаны на языке Python версии 3.8 в интегрированной среде разработки Pycharm версии 2021.2. Файлы доступны для скачивания: https://github.com/RedCactusInn/IU8/tree/master/TSiSA/lab\_1/src

В файле function.py реализована функция f(x) (рисунок 1).

import math  
  
  
def function(x: float):  
 function\_value = (1 - x) \* (1 - x) + math.exp(x)  
 return function\_value

Рисунок 1. Содержимое файла function.py

По алгоритму оптимального пассивного поиска файле passive\_search.py реализована функция поиска (рисунок 2).

import function  
function = function.function  
  
  
def x\_is\_in\_borders(x: float, a\_border: float, b\_border: float) -> bool:  
 if a\_border > b\_border:  
 swap\_tuple = (b\_border, a\_border)  
 a\_border, b\_border = swap\_tuple  
 if a\_border <= x <= b\_border:  
 return True  
 return False  
  
  
def find\_minimum\_passive\_search(a\_border: float, b\_border: float, epsilon: float) -> tuple:  
 x = a\_border  
 dots = []  
 while x\_is\_in\_borders(x, a\_border, b\_border):  
 dots.append((x, function(x)))  
 x += epsilon / 2  
 min\_dot = min(dots, key=lambda elem: elem[1])  
 print(f"INFO: Number of estimations = {len(dots)}")  
 x\_min, y\_min = min\_dot  
 print(f"INFO: Minimum found in x = {x\_min}, y = {y\_min}")  
 return min\_dot, dots

Рисунок 2. Содержимое файла passive\_search.py

В файле fibonacci\_search.py реализован алгоритм поиска методом Фибоначчи (рисунок 3).

import math  
  
import function  
function = function.function  
  
  
class FibonacciSequence:  
  
 \_sequence = []  
  
 def \_\_init\_\_(self, length: int = 10):  
 self.\_sequence = [1, 1]  
 self.extend\_to\_length(length)  
  
 def extend\_to\_length(self, length: int) -> list:  
 assert (len(self.\_sequence) > 1)  
 while len(self.\_sequence) < length:  
 self.\_sequence.append(self.\_sequence[-1] + self.\_sequence[-2])  
  
 def lengthen\_by(self, number\_of\_elements: int):  
 for i in range(number\_of\_elements):  
 self.\_sequence.append(self.\_sequence[-1] + self.\_sequence[-2])  
  
 def get\_sequence(self):  
 return self.\_sequence  
  
  
def get\_number\_of\_iterations(low\_border):  
 fibonacci\_sequence = FibonacciSequence()  
 while fibonacci\_sequence.get\_sequence()[-1] < low\_border:  
 fibonacci\_sequence.lengthen\_by(1)  
 return len(fibonacci\_sequence.get\_sequence())  
  
  
def find\_minimum\_fibonacci\_search(a\_border: float, b\_border: float, epsilon: float) -> tuple:  
 low\_border = math.ceil((b\_border - a\_border) / epsilon)  
 number\_of\_iterations = get\_number\_of\_iterations(low\_border)  
  
 fibonacci\_sequence = FibonacciSequence(number\_of\_iterations).get\_sequence()  
  
 x\_1 = a\_border + (b\_border - a\_border) \* fibonacci\_sequence[-3] / fibonacci\_sequence[-1]  
 x\_2 = a\_border + (b\_border - a\_border) \* fibonacci\_sequence[-2] / fibonacci\_sequence[-1]  
 y\_1 = function(x\_1)  
 y\_2 = function(x\_2)  
  
 dots = [(x\_1, y\_1), (x\_2, y\_2)]  
  
 for i in range(number\_of\_iterations):  
 if y\_1 > y\_2:  
 a\_border = x\_1  
 x\_1 = x\_2  
 y\_1 = y\_2  
 x\_2 = b\_border - (x\_1 - a\_border)  
 y\_2 = function(x\_2)  
 dots.append((x\_2, y\_2))  
 continue  
 b\_border = x\_2  
 x\_2 = x\_1  
 y\_2 = y\_1  
 x\_1 = a\_border + (b\_border - x\_2)  
 y\_1 = function(x\_1)  
 dots.append((x\_1, y\_1))  
 min\_dot = (x\_2, y\_2) if y\_2 < y\_1 else (x\_1, y\_1)  
  
 print(f"INFO: Number of estimations = {len(dots)}")  
 x\_min, y\_min = min\_dot  
 print(f"INFO: Minimum found in x = {x\_min}, y = {y\_min}")  
 return min\_dot, dots

Рисунок 3. Содержимое файла fibonacci\_search.py

В файле main.py выполняется сравнение метода Фибоначи с методом оптимального пассивного поиска при разных значениях допустимой погрешности (рисунок 4).

import re  
  
import matplotlib.pyplot as plt  
import time  
import passive\_search  
import fibonacci\_search  
  
  
def plot\_dots(dots):  
 plt.scatter(\*zip(\*dots))  
 plt.show()  
  
  
def main():  
  
 a\_border = -5.0  
 b\_border = 2.0  
  
 methods = (  
 fibonacci\_search.find\_minimum\_fibonacci\_search,  
 passive\_search.find\_minimum\_passive\_search  
 )  
  
 for epsilon in (0.1, 0.01, 0.001, 0.0001, 0.00001):  
 print(f"\n\n{'+' \* 80}\nepsilon = {epsilon};\n{'+' \* 80}")  
 for method in methods:  
 tic = time.perf\_counter()  
 print(f"\n=== Computations started\n{'-' \* 30}")  
 print(f"INFO: Using method: {re.sub('(find\_minimum\_|\_search)','',method.\_\_name\_\_).upper()}")  
 min\_dot, dots = method(a\_border, b\_border, epsilon)  
 toc = time.perf\_counter()  
 print(f"{'-' \* 30}\n=== Computations finished in {toc - tic} seconds")  
  
 # plot\_dots(dots)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Рисунок 4. Содержимое файла main.py

Вывод программы представлен на рисунке 5.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
epsilon = 0.1;  
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
  
=== Computations started  
------------------------------  
INFO: Using method: FIBONACCI  
INFO: Number of estimations = 13  
INFO: Minimum found in x = 0.3483146067415728, y = 1.8413717278346602  
------------------------------  
=== Computations finished in 0.0001919000000000226 seconds  
  
=== Computations started  
------------------------------  
INFO: Using method: PASSIVE  
INFO: Number of estimations = 141  
INFO: Minimum found in x = 0.2999999999999906, y = 1.8398588075760036  
------------------------------  
=== Computations finished in 0.00012420000000001874 seconds  
  
  
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
epsilon = 0.01;  
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
  
=== Computations started  
------------------------------  
INFO: Using method: FIBONACCI  
INFO: Number of estimations = 18  
INFO: Minimum found in x = 0.31205673758867025, y = 1.8394981398486294  
------------------------------  
=== Computations finished in 0.00010599999999993948 seconds  
  
=== Computations started  
------------------------------  
INFO: Using method: PASSIVE  
INFO: Number of estimations = 1401  
INFO: Minimum found in x = 0.31499999999991574, y = 1.8394843109569967  
------------------------------  
=== Computations finished in 0.0010582999999999565 seconds  
  
  
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
epsilon = 0.001;  
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
  
=== Computations started  
------------------------------  
INFO: Using method: FIBONACCI  
INFO: Number of estimations = 23  
INFO: Minimum found in x = 0.31490955600155335, y = 1.8394843012882653  
------------------------------  
=== Computations finished in 9.959999999997748e-05 seconds  
  
=== Computations started  
------------------------------  
INFO: Using method: PASSIVE  
INFO: Number of estimations = 14001  
INFO: Minimum found in x = 0.31499999999994893, y = 1.8394843109569967  
------------------------------  
=== Computations finished in 0.011626200000000031 seconds  
  
  
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
epsilon = 0.0001;  
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
  
=== Computations started  
------------------------------  
INFO: Using method: FIBONACCI  
INFO: Number of estimations = 27  
INFO: Minimum found in x = 0.3149616794293104, y = 1.8394843034945958  
------------------------------  
=== Computations finished in 0.0006846000000000352 seconds  
  
=== Computations started  
------------------------------  
INFO: Using method: PASSIVE  
INFO: Number of estimations = 140001  
INFO: Minimum found in x = 0.3148999999949901, y = 1.8394843018769693  
------------------------------  
=== Computations finished in 0.12728060000000008 seconds  
  
  
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
epsilon = 1e-05;  
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
  
=== Computations started  
------------------------------  
INFO: Using method: FIBONACCI  
INFO: Number of estimations = 32  
INFO: Minimum found in x = 0.31492356138282585, y = 1.8394843009815038  
------------------------------  
=== Computations finished in 0.006198400000000048 seconds  
  
=== Computations started  
------------------------------  
INFO: Using method: PASSIVE  
INFO: Number of estimations = 1400001  
INFO: Minimum found in x = 0.3149249998951797, y = 1.839484300987432  
------------------------------  
=== Computations finished in 1.2307411 seconds  
  
Process finished with exit code 0

Рисунок 5. Вывод программы

График функции приведён на рисунке 6.

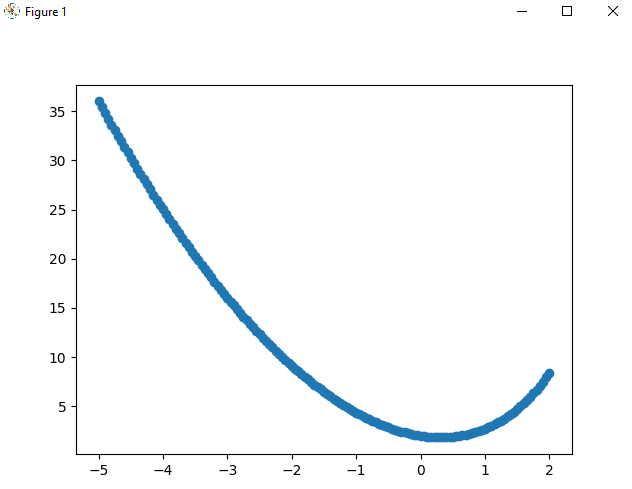


Рисунок 6. График функции

Результаты сравнения представлены на таблице 1.

Таблица 1 Результаты сравнения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Точность | 0.1 | 0.01 | 0.001 | 0.0001 | 1e-5 |
| Число точек измерения, метод пассивного поиска | 141 | 1401 | 14001 | 140001 | 1400001 |
| Затраченное время, метод пассивного поиска, сек | 0,0001242 | 0,0010582 | 0.0116262 | 0.1272806 | 1.2307411 |
| Число точек измерения, метод поиска Фибоначчи | 13 | 18 | 23 | 27 | 32 |
| Затраченное время, метод поиска Фибоначчи, сек | 0,0001919 | 0,0001060 | 9,96e-5 | 0.0006846 | 0.0061984 |

**Вывод:**

Исследовано функционирование и проведён сравнительный анализ алгоритмов прямого поиска экстремума методом оптимального пассивного поиска и методом Фибоначчи на примере унимодальной функции одного переменного.

Метод оптимального пассивного поиска и метод поиска Фибоначчи имеют разную алгоритмическую сложность. На малом значении точности это не заметно, однако с его увеличением очевидно, что метод поиска Фибоначчи работает значительно быстрее.