**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Кафедра безопасности информационных систем**

**ОТЧЁТ**

по практической работе работе №2 на тему:   
**«Анализ алгоритмов и программ»**

по дисциплине «Алгоритмы и структуры данных»

Выполнил: студент группы ИСТ-114,Константинов А.Д.

«21» сентября 2022 г. \_\_\_\_\_\_\_\_\_\_\_/А.Д. Константинов/

Принял: к.ф.-м.н., доцент, И.А. Моисеев

«05» октября 2022 г. \_\_\_\_\_\_\_\_\_\_\_/ И.А. Моисеев /

**Содержание:**

[Цель работы: 3](#_Toc114692552)

[Листинг кода: 3](#_Toc114692553)

[Результат работы программы: 7](#_Toc114692554)

[Вывод: 7](#_Toc114692555)

[Приложение: 16](#_Toc114692556)

Цель работы:

Найти локальный минимум (максимум) функции вида f(x) = на заданном интервале [a,b] с заданной точностью ε > 0 одним из способов:

1) Методом «деления отрезка» пополам;

2) Методом «золотого сечения»;

3) Методом «Фибоначчи».

Провести анализ разработанного алгоритма и программы и сравнить с аналогичным решением с помощью алгоритма «пассивного поиска».

# Листинг кода:

#include <iostream>  
#include <cmath>  
#include <ctime>  
float func(float x) {  
 return pow(x,3)-x+exp(-x);  
}  
void PassiveSearch(float left,float right, float e)  
{  
 srand(time(0));  
 int steps = 0;  
 float minPoint = 0;  
 float minVal = 99999999;  
 for (float i = left; i <= right; i+=e) {  
 steps++;  
 float tmp = func(i);  
 if(minVal > tmp) {  
 minVal = tmp;  
 minPoint = i;  
 }  
 }  
  
 std::cout << "Result X:" << minPoint << " Y:" << func(minPoint) <<" Steps:"<< steps<<" Time:" << clock()/1000.0<< std::endl;  
}  
void Task1\_SearchHalfPart(float a,float b,float e){  
 float x1;  
 float a1=0,b1 = 999999;  
 float a0 = a;  
 float b0 = b;  
 srand(time(0));  
 int steps = 0;  
 while(b1-a1 > e) {  
 float om = (b0 - a0) / 10;  
 float alp0 = (a0 + b0) / 2 - om;  
 float bet0 = alp0 + 2 \* om;  
 float fAlp = func(alp0);  
 float fBet = func(bet0);  
  
  
 if (fAlp <= fBet) {  
 x1 = alp0;  
 a1 = a0;  
 b1 = bet0;  
 } else {  
 a1 = alp0;  
 x1 = bet0;  
 b1 = b0;  
 }  
  
 a0 = a1;  
 b0 = b1;  
 steps++;  
 std::cout << "x:" << (a1+b1)/2 << " y:" << func((a1+b1)/2) << std::endl;  
 }  
 float res = (a1 + b1) / 2;  
 std::cout << "Result X:" << res << " Y:" << func(res) <<" Steps:"<< steps<<" Time:" << clock()/1000.0 << std::endl;  
}  
void Task1\_SearchGoldCut(float a,float b,float e){  
 float x1;  
 float a1=0,b1 = 999999;  
 float a0 = a;  
 float b0 = b;  
 srand(time(0));  
 int steps = 0;  
  
 while(b1-a1 > e) {  
 float om = b0 - a0;  
 float alp0 =a0+(2\*om)/(3+sqrt(5));  
 float bet0 = a0+(2\*om)/(1+sqrt(5));  
 float fAlp = func(alp0);  
 float fBet = func(bet0);  
  
  
 if (fAlp <= fBet) {  
 x1 = alp0;  
 a1 = a0;  
 b1 = bet0;  
 } else {  
 a1 = alp0;  
 x1 = bet0;  
 b1 = b0;  
 }  
  
 a0 = a1;  
 b0 = b1;  
 steps++;  
 std::cout << "x:" << (a1+b1)/2 << " y:" << func((a1+b1)/2) << std::endl;  
 }  
 float res = (a1 + b1) / 2;  
 std::cout << "Result X:" << res << " Y:" << func(res) <<" Steps:"<< steps<<" Time:" << clock()/1000.0 << std::endl;  
}  
float Fib(int n)  
{  
 if(n == 0) return 1;  
 if(n== 1) return 1;  
 return Fib(n-1) + Fib(n-2);  
}  
void Task1\_SearchFib(float a, float b, float e)  
{  
 int f0 = 1;  
 int f1=1;  
 float a0 = a;  
 float b0 = b;  
 float d0 = b0-a0;  
 int N = 1;  
 srand(time(0));  
 int steps = 0;  
 while(d0/Fib(N) >= e)  
 {  
 N++;  
 }  
  
  
 float x1=0;  
 float a1=0,b1 = 999999;  
 int k =0;  
 while(b1-a1 > e) {  
 float om = b0 - a0;  
 float alp0 = a0 + Fib(N-k-2)/Fib(N-k)\*om;  
 float bet0 = a0 + Fib(N-k-1)/Fib(N-k)\*om;  
 float fAlp = func(alp0);  
 float fBet = func(bet0);  
  
 if (fAlp <= fBet) {  
 x1 = alp0;  
 a1 = a0;  
 b1 = bet0;  
 } else {  
 a1 = alp0;  
 x1 = bet0;  
 b1 = b0;  
 }  
  
 a0 = a1;  
 b0 = b1;  
 k++;  
 steps++;  
 std::cout << "x:" << (a1+b1)/2 << " y:" << func((a1+b1)/2) << std::endl;  
 }  
 float res = (a1 + b1) / 2;  
 std::cout << "Result X:" << res << " Y:" << func(res) <<" Steps:"<< steps<<" Time:" << clock()/1000.0 << std::endl;  
}  
int main() {  
 float left,right,e;  
// std::cout << "Current function y=x^3-x+e^(-x)" << std::endl;  
// std::cout << "Type left limit:";  
// std::cin >> left;  
// std::cout << "Type right limit:";  
// std::cin >> right;  
// std::cout << "Type accuracy:";  
// std::cin >> e;  
// std::cout<<std::endl;  
//  
// PassiveSearch(left,right,e);  
 e= 0.001;  
 std::cout<< "Task 2 Functions(accuracy = 0.001)" << std::endl;  
 std::cout << std::endl<<"Part 1: Searching extremum in range[-5,-3]" << std::endl;  
 left = -5;  
 right = -3;  
 std::cout <<std::endl<< "Method 1: Classic search: "<< std::endl; PassiveSearch(left,right,e);  
 std::cout <<std::endl<< "Method 2: Like-binary search: "<< std::endl;Task1\_SearchHalfPart(left,right,e);  
 std::cout <<std::endl<< "Method 3: Gold-cut search: "<< std::endl;Task1\_SearchGoldCut(left,right,e);  
 std::cout <<std::endl<< "Method 4: Fib search: "<< std::endl;Task1\_SearchFib(left,right,e);  
  
 std::cout << std::endl<<"Part 2: Search Max-points in range [-3,0]"; std::cout << std::endl;  
 left = -3;  
 right = 0;  
 std::cout <<std::endl<< "Method 1: Classic search: "<< std::endl; PassiveSearch(left,right,e);  
 std::cout <<std::endl<< "Method 2: Like-binary search: "<< std::endl;Task1\_SearchHalfPart(left,right,e);  
 std::cout <<std::endl<< "Method 3: Gold-cut search: "<< std::endl;Task1\_SearchGoldCut(left,right,e);  
 std::cout <<std::endl<< "Method 4: Fib search: "<< std::endl;Task1\_SearchFib(left,right,e);  
  
 std::cout <<std::endl << "Part 3: Search Min-points in range [0,3]"; std::cout << std::endl;  
 left = 0;  
 right = 3;  
 std::cout <<std::endl<< "Method 1: Classic search: "<< std::endl; PassiveSearch(left,right,e);  
 std::cout <<std::endl<< "Method 2: Like-binary search: "<< std::endl;Task1\_SearchHalfPart(left,right,e);  
 std::cout <<std::endl<< "Method 3: Gold-cut search: "<< std::endl;Task1\_SearchGoldCut(left,right,e);  
 std::cout <<std::endl<< "Method 4: Fib search: "<< std::endl;Task1\_SearchFib(left,right,e);  
  
 std::cout <<std::endl<< "End. Thanks for using our software" << std::endl;  
 return 0;  
}

# Результат работы программы:

# /Users/andrew/Projects/testproj/cmake-build-debug/testproj

# Task 2 Functions(accuracy = 0.001)

# Part 1: Searching extremum in range[-5,-3]

# Method 1: Classic search:

# Result X:-3.6791 Y:-6.50965 Steps:2001 Time:21.93

# Method 2: Like-binary search:

# x:-3.6 y:-6.45776

# x:-3.84 y:-6.25763

# x:-3.696 y:-6.50706

# x:-3.6096 y:-6.46937

# x:-3.66144 y:-6.507

# x:-3.69254 y:-6.508

# x:-3.67388 y:-6.50943

# x:-3.68508 y:-6.50931

# x:-3.67836 y:-6.50964

# x:-3.68239 y:-6.50954

# x:-3.67997 y:-6.50964

# x:-3.67852 y:-6.50965

# x:-3.67939 y:-6.50965

# x:-3.67887 y:-6.50965

# x:-3.67856 y:-6.50965

# Result X:-3.67856 Y:-6.50965 Steps:15 Time:22.019

# Method 3: Gold-cut search:

# x:-3.61803 y:-6.47841

# x:-3.8541 y:-6.2089

# x:-3.7082 y:-6.50198

# x:-3.61803 y:-6.47841

# x:-3.67376 y:-6.50942

# x:-3.7082 y:-6.50198

# x:-3.68692 y:-6.50908

# x:-3.67376 y:-6.50942

# x:-3.68189 y:-6.50957

# x:-3.67687 y:-6.50961

# x:-3.67997 y:-6.50964

# x:-3.67805 y:-6.50964

# x:-3.67924 y:-6.50965

# x:-3.67851 y:-6.50964

# x:-3.67896 y:-6.50965

# x:-3.67868 y:-6.50965

# Result X:-3.67868 Y:-6.50965 Steps:16 Time:22.101

# Method 4: Fib search:

# x:-3.61803 y:-6.47841

# x:-3.8541 y:-6.2089

# x:-3.7082 y:-6.50198

# x:-3.61803 y:-6.47841

# x:-3.67376 y:-6.50942

# x:-3.7082 y:-6.50198

# x:-3.68692 y:-6.50908

# x:-3.67376 y:-6.50942

# x:-3.68189 y:-6.50957

# x:-3.67686 y:-6.50961

# x:-3.67995 y:-6.50964

# x:-3.67802 y:-6.50964

# x:-3.67918 y:-6.50965

# x:-3.67841 y:-6.50964

# x:-3.67879 y:-6.50965

# x:-3.67918 y:-6.50965

# Result X:-3.67918 Y:-6.50965 Steps:16 Time:22.483

# Part 2: Search Max-points in range [-3,0]

# Method 1: Classic search:

# Result X:-3 Y:-3.91446 Steps:3001 Time:22.738

# Method 2: Like-binary search:

# x:-2.1 y:1.00517

# x:-2.46 y:-0.722125

# x:-2.676 y:-1.9599

# x:-2.8056 y:-2.74138

# x:-2.88336 y:-3.21399

# x:-2.93002 y:-3.49622

# x:-2.95801 y:-3.66445

# x:-2.97481 y:-3.76483

# x:-2.98488 y:-3.82483

# x:-2.99093 y:-3.86074

# x:-2.99456 y:-3.88224

# x:-2.99673 y:-3.89514

# x:-2.99804 y:-3.90287

# x:-2.99882 y:-3.90751

# x:-2.99929 y:-3.91029

# x:-2.99958 y:-3.91196

# Result X:-2.99958 Y:-3.91196 Steps:16 Time:22.833

# Method 3: Gold-cut search:

# x:-2.07295 y:1.11347

# x:-2.42705 y:-0.544245

# x:-2.6459 y:-1.78134

# x:-2.78115 y:-2.59292

# x:-2.86475 y:-3.10096

# x:-2.91641 y:-3.41411

# x:-2.94834 y:-3.60644

# x:-2.96807 y:-3.72463

# x:-2.98027 y:-3.79736

# x:-2.9878 y:-3.84218

# x:-2.99246 y:-3.86983

# x:-2.99534 y:-3.88689

# x:-2.99712 y:-3.89743

# x:-2.99822 y:-3.90393

# x:-2.9989 y:-3.90796

# x:-2.99932 y:-3.91044

# x:-2.99958 y:-3.91198

# Result X:-2.99958 Y:-3.91198 Steps:17 Time:22.938

# Method 4: Fib search:

# x:-2.07295 y:1.11347

# x:-2.42705 y:-0.544245

# x:-2.6459 y:-1.78135

# x:-2.78115 y:-2.59292

# x:-2.86475 y:-3.10096

# x:-2.91641 y:-3.4141

# x:-2.94834 y:-3.60644

# x:-2.96807 y:-3.72463

# x:-2.98027 y:-3.79737

# x:-2.9878 y:-3.84217

# x:-2.99247 y:-3.86985

# x:-2.99534 y:-3.88686

# x:-2.99713 y:-3.89748

# x:-2.99821 y:-3.90385

# x:-2.99892 y:-3.9081

# x:-2.99928 y:-3.91022

# x:-2.99964 y:-3.91234

# Result X:-2.99964 Y:-3.91234 Steps:17 Time:23.534

# Part 3: Search Min-points in range [0,3]

# Method 1: Classic search:

# Result X:0.705994 Y:0.139511 Steps:3001 Time:23.748

# Method 2: Like-binary search:

# x:0.9 y:0.23557

# x:0.54 y:0.200212

# x:0.756 y:0.145622

# x:0.6264 y:0.153898

# x:0.70416 y:0.139515

# x:0.750816 y:0.144419

# x:0.722822 y:0.140213

# x:0.706026 y:0.139511

# x:0.695949 y:0.139732

# x:0.701995 y:0.139542

# x:0.705623 y:0.13951

# x:0.7078 y:0.139521

# x:0.706494 y:0.139512

# x:0.70571 y:0.13951

# x:0.70524 y:0.139511

# x:0.705522 y:0.13951

# Result X:0.705522 Y:0.13951 Steps:16 Time:23.839

# Method 3: Gold-cut search:

# x:0.927051 y:0.265397

# x:0.572949 y:0.178993

# x:0.791796 y:0.157644

# x:0.656541 y:0.145101

# x:0.740133 y:0.14236

# x:0.688471 y:0.140203

# x:0.7204 y:0.140028

# x:0.700666 y:0.139569

# x:0.712862 y:0.139634

# x:0.705325 y:0.13951

# x:0.709983 y:0.139555

# x:0.707104 y:0.139515

# x:0.705325 y:0.13951

# x:0.706425 y:0.139512

# x:0.705745 y:0.13951

# x:0.705325 y:0.13951

# x:0.705585 y:0.13951

# Result X:0.705585 Y:0.13951 Steps:17 Time:23.927

# Method 4: Fib search:

# x:0.927051 y:0.265397

# x:0.572949 y:0.178993

# x:0.791796 y:0.157644

# x:0.656541 y:0.145101

# x:0.740134 y:0.14236

# x:0.688472 y:0.140202

# x:0.720402 y:0.140028

# x:0.70067 y:0.139569

# x:0.712868 y:0.139634

# x:0.705334 y:0.13951

# x:0.709998 y:0.139555

# x:0.707127 y:0.139515

# x:0.705334 y:0.13951

# x:0.70641 y:0.139512

# x:0.705692 y:0.13951

# x:0.705334 y:0.13951

# x:0.704975 y:0.139511

# Result X:0.704975 Y:0.139511 Steps:17 Time:24.459

# End. Thanks for using our software

# Process finished with exit code 0

# Вывод:

В ходе выполнения практической работы была написана программа, которая находит локальные минимумы(максимумы) функции на трех отрезках – [-5,-3];[-3,0];[0,3]. Каждый промежуток решается при помощи четырех различных методов – пассивный поиск(последовательный перебор чисел), поиск делением отрезка пополам, поиск золотым сечение, поиск методом Фибоначчи. Самым тяжелым для исполнения является пассивный поиск – он выполняет 3000 операций для поиска подходящего числа, самый быстрый алгоритм - поиск методом деления пополам, он требует в среднем на одну итерацию алгоритма меньше чем остальные, что является чрезвычайно хорошим результатом. По точности все алгоритмы выдают заданные значения – 0.001 – максимальное различие ответов. Вся программа поделена на 6 функций и работает на вывод в текстовом режиме. В результате выводит все расчеты и заканчивает свою работу.

# Приложение:

Тестовые файлы(a.txt;b.txt). и файл исходного кода main.cpp можно найти по адресу <https://github.com/StSiRe/Algoritms/tree/master/Algorithms%20and%20Data%20Structures/Second%20task/second>