ФГБОУ ВО

«Уфимский государственный авиационный технический университет»

Кафедра ТК

**ОТЧЕТ**

**по лабораторной работе № 2**

по дисциплине «Методы оптимизации»

**Тема: «Методы одномерной оптимизации, использующие производные»**

Вариант № 24

Выполнила: студентка гр. ИВТ-221

Самсонова Е.О.

Проверил: доцент каф. ТК

Хасанов А.Ю.

Уфа 2021

Содержание

[Задание 3](#_Toc70284578)

[Выполнение задания 3](#_Toc70284579)

[Расчетные таблицы 5](#_Toc70284580)

[График эффективности методов 15](#_Toc70284586)

[Код программы 15](#_Toc70284587)

[Блок-схемы функций методов 15](#_Toc70284588)

[Вывод 24](#_Toc70284589)

Задание

Вычислить минимальное значение функции на интервале [-2;0] с заданной точностью ɛ.

Выполнение задания

Целевая функция:

Область неопределенности:[-2;0]

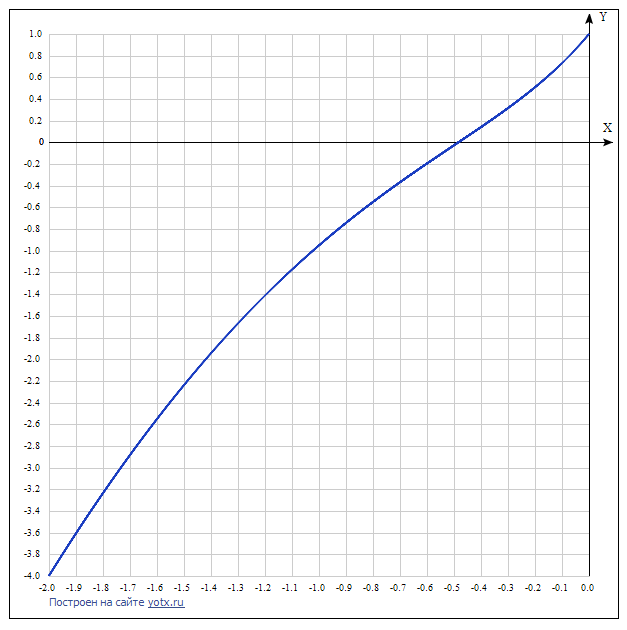


Рисунок 1 − График функции на интервале [-2;0]

Проверка применимости методов, использующих производные, к

заданной целевой функции на заданном отрезке локализации. Критерием

применимости является выпуклость целевой функции на заданном отрезке

локализации.

Первый критерий выпуклости:

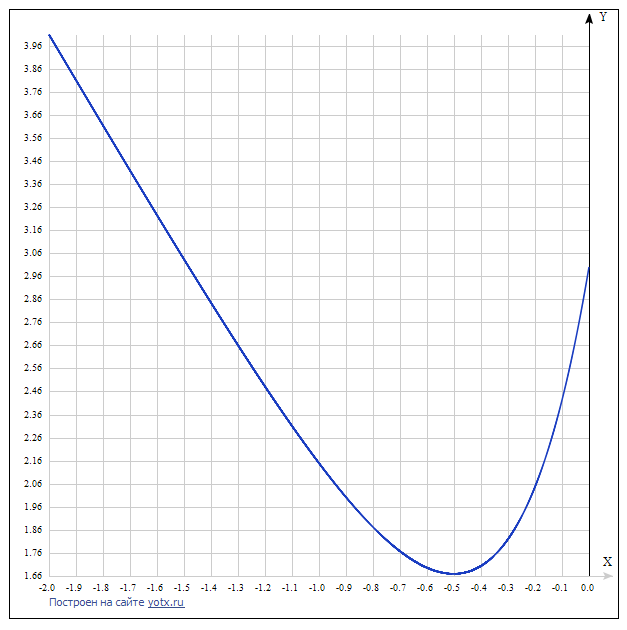


Рисунок 2 − График функции первой производной на интервале [-2;0]

Второй критерий выпуклости:

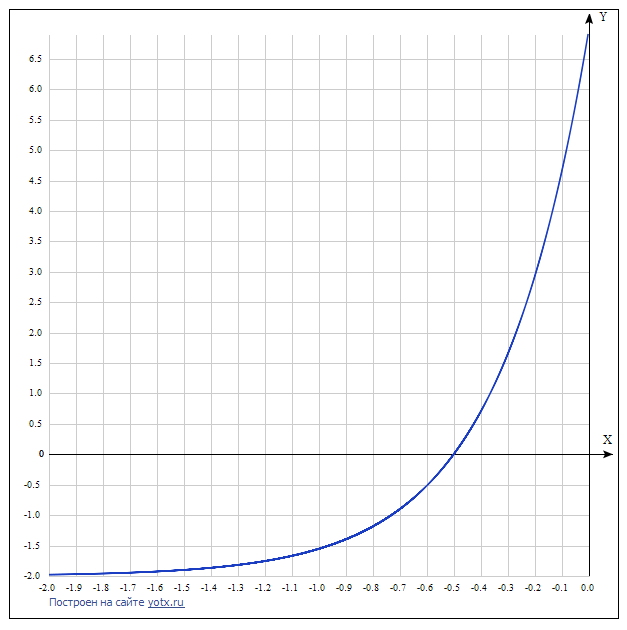


Рисунок 3 − График функции второй производной на интервале [-2;0]

И из 1-го, и 2-ого критериев следует, что заданная целевая

функция на заданном отрезке локализации является выпуклой.

Следовательно, к ней применимы рассматриваемые методы, использующие

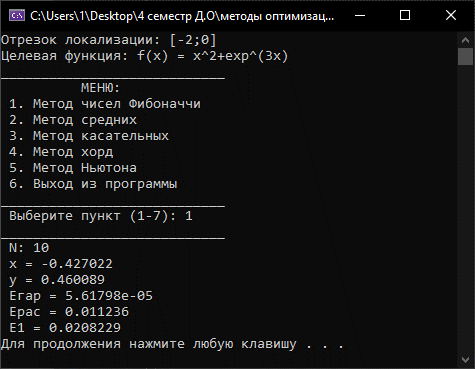
производные.

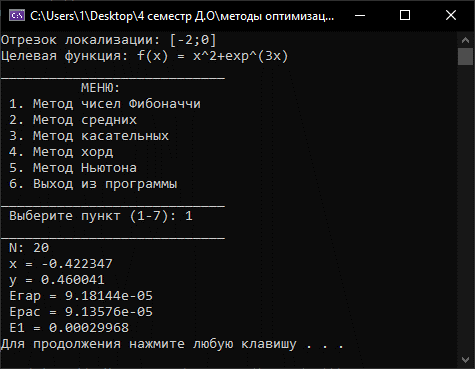
В ходе решения были использованы методы

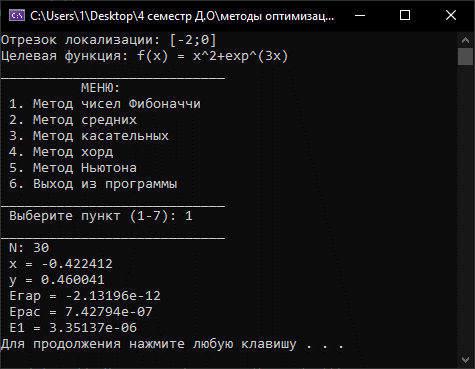
1. Метод чисел Фибоначчи
2. Метод средних
3. Метод касательных
4. Метод хорд
5. Метод Ньютона

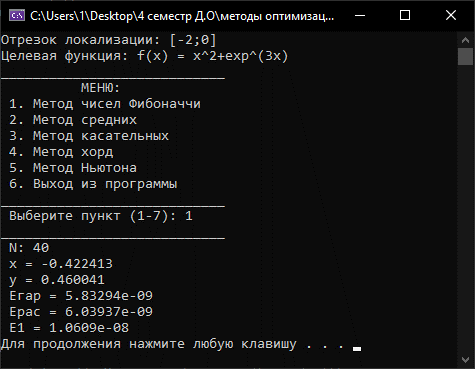
Расчетные таблицы

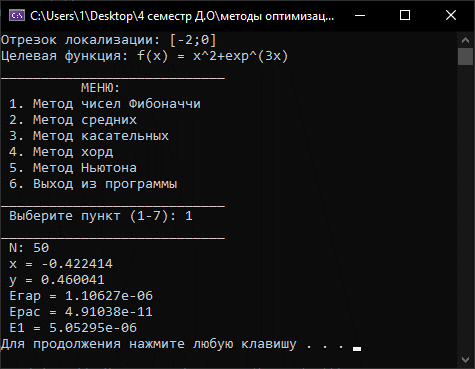
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Метод чисел Фибоначчи | | | | | |
| N |  |  |  |  |  |
| 10 | -0,427022 | 0,460089 | 5,61798e-05 | 0,0208229 | 0,011236 |
| 20 | -0,422347 | 0,460041 | 9,18144e-05 | 0,00029968 | 9,13576e-05 |
| 30 | -0,422412 | 0,460041 | -2,13196e-12 | 3,35137e-06 | 7,42794e-07 |
| 40 | -0,422413 | 0,460041 | 5,83294e-09 | 1,0609e-08 | 6,03937e-09 |
| 50 | -0,422414 | 0,460041 | 1,10627e-06 | 5,05295e-06 | 4,91038e-11 |



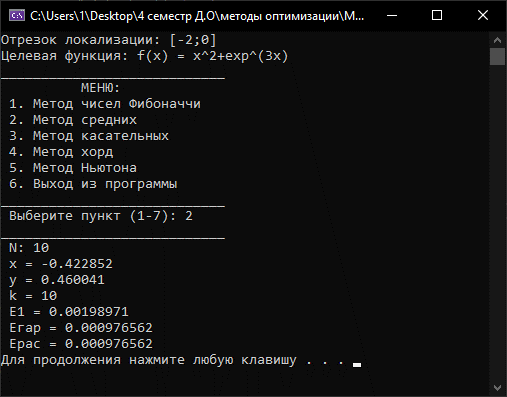


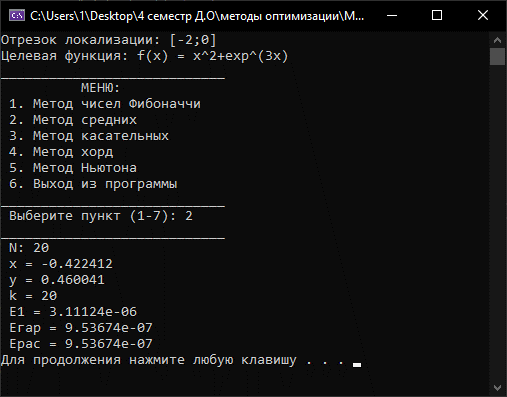


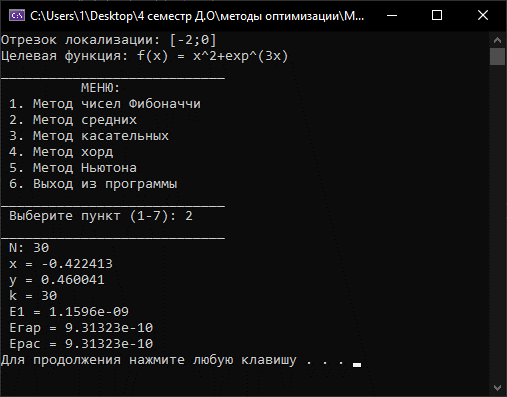


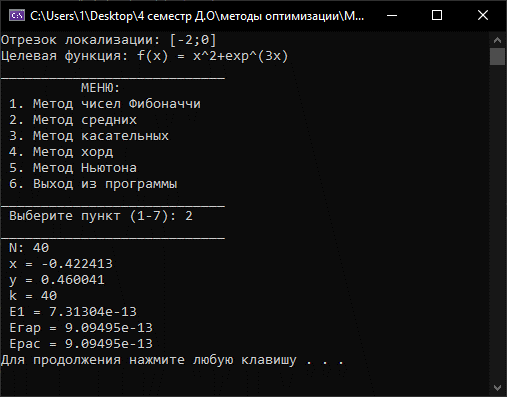


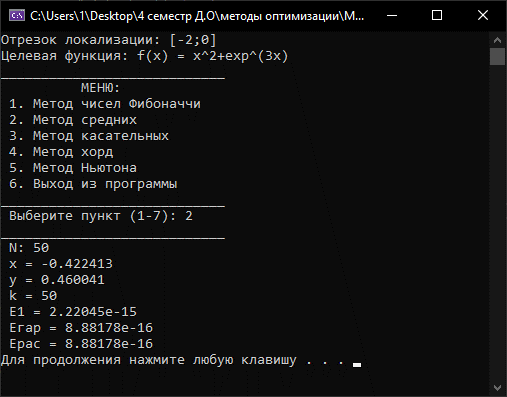
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Метод средних | | | | | | |
| N |  |  | k |  |  |  |
| 10 | -0,422852 | 0,460041 | 10 | 0,00198971 | 0,000976562 | 0,000976562 |
| 20 | -0,422412 | 0,460041 | 20 | 3,11124e-06 | 9,53674e-07 | 9,53674e-07 |
| 30 | -0,422413 | 0,460041 | 30 | 1,1596e-09 | 9,31323e-10 | 9,31323e-10 |
| 40 | -0,422413 | 0,460041 | 40 | 7,31304e-13 | 9,094953e-13 | 9,094953e-13 |
| 50 | -0,422413 | 0,460041 | 50 | 2,22045e-15 | 8,88178e-16 | 8,88178e-16 |



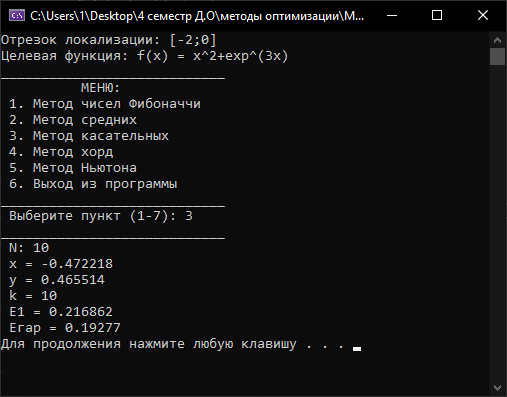


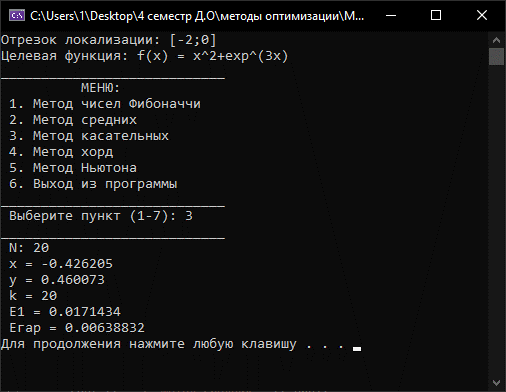


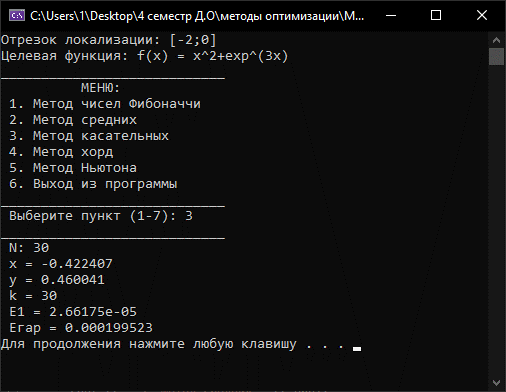


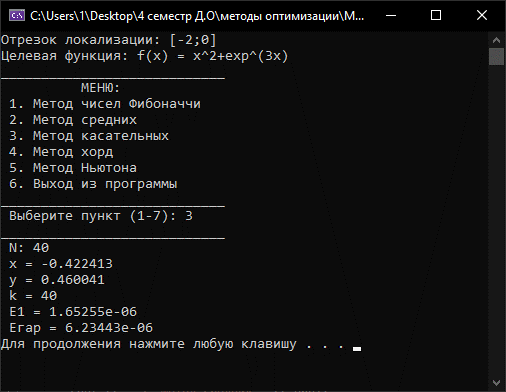


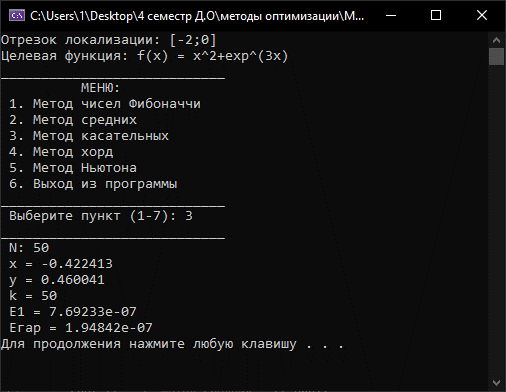
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Метод касательных | | | | | |
| N |  |  | k |  |  |
| 10 | -0,472218 | 0,465514 | 10 | 0,216862 | 0,19277 |
| 20 | -0,426205 | 0,460073 | 20 | 0,0171434 | 0,00638832 |
| 30 | -0,422407 | 0,460041 | 30 | 2,66175e-05 | 0,000199523 |
| 40 | -0,422413 | 0,460041 | 40 | 1,65255e-06 | 6,23443e-06 |
| 50 | -0,422413 | 0,460041 | 50 | 7,69233e-07 | 1,94842e-07 |



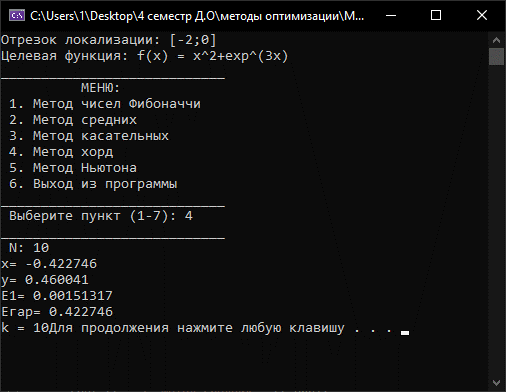


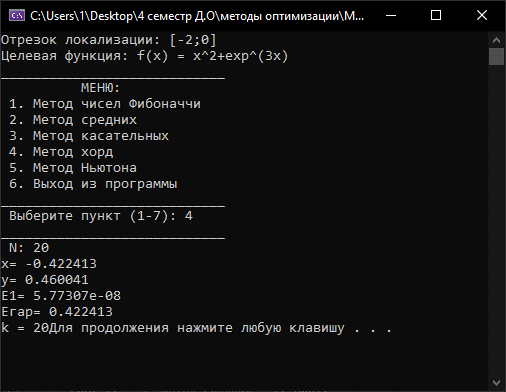


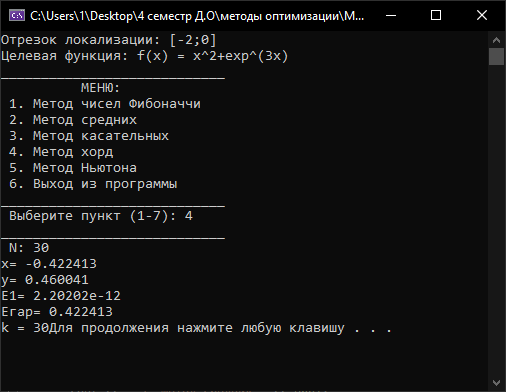


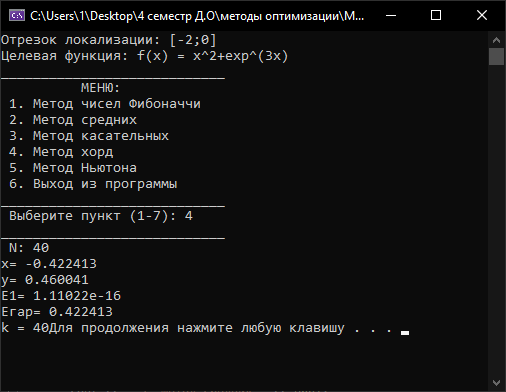


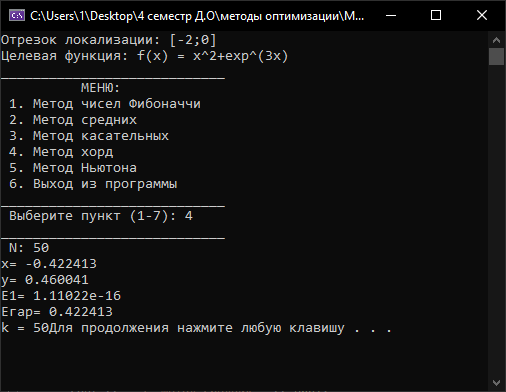
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Метод хорд | | | | | |
| N |  |  | k |  |  |
| 10 | -0,422746 | 0,460041 | 10 | 0,0051317 | 0,422746 |
| 20 | -0,422413 | 0,460041 | 20 | 5,77307e-08 | 0,422413 |
| 30 | -0,422413 | 0,460041 | 30 | 2,20202e-12 | 0,422413 |
| 40 | -0,422413 | 0,460041 | 40 | 1,11022e-16 | 0,422413 |
| 50 | -0,422413 | 0,460041 | 50 | 1,11022e-16 | 0,422413 |



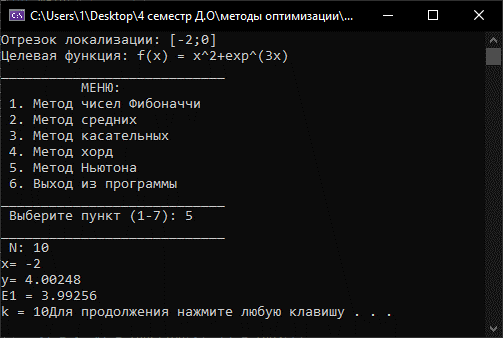








|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод Ньютона | | | | |
| N |  |  | k |  |
| 10 | -2 | 4,00248 | 10 | 3,99256 |
| 20 | -2 | 4,00248 | 20 | 3,99256 |
| 30 | -2 | 4,00248 | 30 | 3,99256 |
| 40 | -2 | 4,00248 | 40 | 3,99256 |
| 50 | -2 | 4,00248 | 50 | 3,99256 |



**Вывод:** для заданной целевой функции на заданном отрезке

локализации при заданных числах экспериментов для рассмотренных

методов наилучший результат, по Е1 (значению производной), дают метод Ньютона и метод хорд.

График эффективности методов

Рисунок 4 − Сравнение методов по точности от количества экспериментов

Код программы

#include <iostream>

#include <math.h>

#include <conio.h>

#include <cstdio>

#include <clocale>

using namespace std;

const int M = 100;

void casat(double a, double b);

void srednix(double a, double b);

void fiban(double a, double b);

void newton(double a, double b);

void hord(double a, double b);

double function(double x);

double proizv(double x);

double proizv2(double x);

int main()

{

double a = -2, b = 0;

setlocale(LC\_ALL, "Russian");

int H;

while (1)

{

system("cls");

cout << "Отрезок локализации: " << "[" << a << ";" << b << "]" << endl;

cout << "Целевая функция: f(x) = x^2+exp^(3x)" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << " МЕНЮ:" << endl;

cout << " 1. Метод чисел Фибоначчи " << endl;

cout << " 2. Метод средних " << endl;

cout << " 3. Метод касательных " << endl;

cout << " 4. Метод хорд " << endl;

cout << " 5. Метод Ньютона " << endl;

cout << " 6. Выход из программы" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << " Выберите пункт (1-7): ";

cin >> H;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

switch (H)

{

case 1:fiban(a, b); break;

case 2:srednix(a, b); break;

case 3:casat(a, b); break;

case 4:hord(a, b); break;

case 5:newton(a, b); break;

case 6: system("pause");

return 0;

default: cout << " Такого пункта нет в меню.\n"; break;

}

}

}

double function(double x)

{

return pow(x, 2) + exp(3\*x);

}

double proizv(double x)

{

return 3\*exp(3\*x)+2\*x;

}

double proizv2(double x)

{

return 9\*exp(3\*x)+2;

}

void casat(double a, double b)

{

int N, k = 0;

double z1, x, y2, z2, E1=0, z, y, x1=0, y1, Egar=0;

cout << " N: "; cin >> N;

y1 = function(a);

z1 = proizv(a);

y2 = function(b);

z2 = proizv(b); k = 4;

do

{

x = ((z2 \* b - z1 \* a) - (y2 - y1)) / (z2 - z1);

y = function(x);

z = proizv(x); k = k + 2;

if (z == 0)

{

x1 = x; y1 = y; Egar = 0; E1 = 0;

}

else

{

if (z > 0)

{

b = x; y2 = y; z2 = z;

}

else

{

a = x; y1 = y; z1 = z;

}

}

} while (k < N);

x1 = x; y1 = y; Egar = b - a; E1 = abs(proizv(x1));

cout << " x = " << x1 << "\n y = " << y1 << "\n k = " << k << "\n E1 = "

<< E1 << "\n Eгар = " << Egar << "\n";

system("pause");

}

void srednix(double a, double b)

{

int N, k;

double x, E1, z, x1, y1, Egar, Epac;

cout << " N: "; cin >> N;

Epac = (b - a) / (pow(2, N+1));

k = 0;

do

{

x = (a + b) / 2; z = proizv(x); k = k + 1;

if (abs(z)>0)

{

if (z > 0) b = x;

else a = x;

}

else

{

Egar = 0; x1 = x; y1 = function(x1); E1 = z;

}

} while (k < N);

Egar = (b - a) / 2;

x1 = (a + b) / 2;

y1 = function(x1);

E1 = abs(proizv(x1));

cout << " x = " << x1 << "\n y = " << y1 << "\n k = " << k << "\n E1 = "

<< E1 << "\n Eгар = " << Egar << "\n Epac = " << Epac << "\n";

system("pause");

}

void fiban(double a, double b)

{

double Egar, E1, d, x1, x2, y1, y2, x, y, F[M], Epac;

int i, n;

setlocale(LC\_ALL, "Russian");

cout << " N: "; cin >> n;

F[0] = 1; F[1] = 1;

for (i = 2; i <= n; i++)

{

F[i] = F[i - 1] + F[i - 2];

}

x1 = a + ((F[n - 2] / F[n]) \* (b - a)); x2 = a + ((F[n - 1] / F[n]) \* (b - a)); y1 =

function(x1);

y2 = function(x2); d = ((b - a) / (2 \* F[n])) / 100; Epac = (b - a) / (2 \* F[n]);

for (i = 0; i <= (n - 3); i++)

{

if (y1 <= y2) {

b = x2; x2 = x1; y2 = y1; x1 = a + b - x2; y1 = function(x1);

}

else {

a = x1; x1 = x2; y1 = y2; x2 = a + b - x1; y2 = function(x2);

}

}

if (y1 <= y2) { b = x2; x2 = x1; y2 = y1; }

else { a = x1; }

x1 = x2 - d;

y1 = function(x1);

if (y1 <= y2) { b = x2; }

else { a = x1; }

x = (a + b) / 2;

y = function(x);

Egar = (b - a) / 2;

E1 = fabs(proizv(x));

cout << " x = " << x << "\n y = " << y << "\n Eгар = " << Egar

<< "\n Ерас = " << Epac << "\n E1 = " << E1 << "\n";

system("pause");

}

void newton(double a, double b)

{

double x, xk=0, z, u, E1, x1=0, y1, N;

int k;

cout << " N: "; cin >> N;

x = a;

z = proizv(x);

u = proizv2(x);

k = 2;

while (k < N && z!=0) {

xk = x - z / u;

z = proizv(x); u = proizv2(x);

k += 2;

}

x1 = x; y1 = function(x); E1 = fabs(z);

cout << "x= " << x1 << "\ny= " << y1 << "\nE1 = " << E1 << "\nk = " << k;

system("pause");

}

void hord(double a, double b)

{

double N, e = 0, x1 = 0, y1 = 0, Z = 0, Z1 = 0, Z2 = 0, x = 0, ecr = 0;

int k;

cout << " N: "; cin >> N;

Z1 = proizv(a); Z2 = proizv(b); k = 2;

do

{

x = a - (Z1 / (Z2 - Z1)) \* (b - a);

Z = proizv(x);

k = k + 1;

if (abs(Z) > 0)

{

if (Z > 0) { b = x; Z2 = Z; }

else { a = x; Z1 = Z; }

}

else

{

x1 = x; y1 = function(x1); ecr = 0; e = 0; break;

}

} while (k < N);

e = fabs(Z); x1 = x; y1 = function(x1); ecr = b - a;

cout << "x= " << x1 << "\ny= " << y1 << "\nE1= " << e << "\nEгар= " << ecr << "\nk = " << k;

system("pause");

}

Блок-схемы функций методов



Рисунок 5 − Блок-схема функции метода чисел Фибоначчи



Рисунок 6 − Блок-схема функции метода средней точки



Рисунок 7 − Блок-схема функции метода хорд



Рисунок 8 − Блок-схема функции метода касательных



Рисунок 9 − Блок-схема функции метода Ньютона

Вывод

В ходе лабораторной работы были изучены различные методы одномерной оптимизации и сравнены эффективности их применения для конкретных целевых функций. По графику видно, что одни из самых точных — это метод Ньютона и хорд.