ФГБОУ ВО

«Уфимский государственный авиационный технический университет»

Кафедра ТК

**ОТЧЕТ**

**по лабораторной работе № 1**

по дисциплине «Методы оптимизации»

Тема: «Методы одномерной оптимизации»

Вариант № 4

Выполнил: студент гр. ИВТ-221

Хуснутдинов Х.В.

Проверил: доцент каф. ТК

Хасанов А.Ю.

Уфа 2021

Задание

Вычислить минимум функции на интервале [0;1] с заданной точностью ɛ.

Выполнение задания

Целевая функция:

Область неопределенности:[0;1]

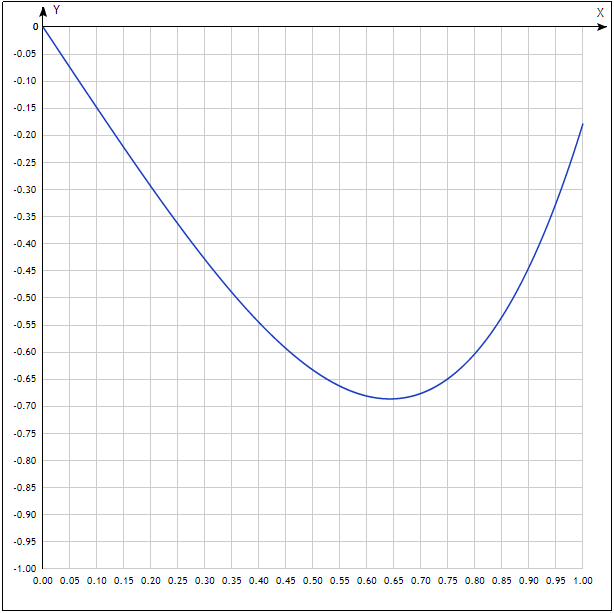


Рисунок 1 − График функции: на интервале [0;2]

В ходе решения были использованы методы

1. Метод перебора
2. Пассивно-оптимальный метод
3. Метод деления отрезка пополам
4. Метод золотого сечения
5. Метод дихотомии
6. Метод поразрядного поиска

Расчетные таблицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Метод золотого сечения | | | | | |
| ɛ | x | y |  | N |  |
| 0,1 | 0,600813 | -0,681222 | 0,072949 | 5 | 5 |
| 0,01 | 0,645898 | -0,686187 | 0,00657781 | 10 | 10 |
| 0,001 | 0,642199 | -0,686218 | 0,000959689 | 14 | 14 |
| 0,0001 | 0,642619 | -0,686218 | 8,65351e-05 | 19 | 19 |
| 0,00001 | 0,642632 | -0,686218 | 7,80287e-06 | 24 | 24 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод дихотомии | | | | |
| ɛ | x | y |  | N |
| 0,1 | 0,68375 | -0,681035 | 0,07125 | 6 |
| 0,01 | 0,648141 | -0,686128 | 0,00879688 | 12 |
| 0,001 | 0,643038 | -0,686218 | 0,000588184 | 20 |
| 0,0001 | 0,642636 | -0,686218 | 7,10339e-05 | 26 |
| 0,00001 | 0,642631 | -0,686218 | 8,62938e-06 | 32 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод поразрядного поиска | | | | |
| ɛ | x | y |  | N |
| 0,1 | 0,625 | -0,685311 | -0,0625 | 12 |
| 0,01 | 0,644531 | -0,686207 | -0,00390625 | 22 |
| 0,001 | 0,642578 | -0,686218 | 0,000976563 | 25 |
| 0,0001 | 0,642639 | -0,686218 | 6,10352e-05 | 36 |
| 0,00001 | 0,642635 | -0,686218 | 3,8147e-06 | 45 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Пассивно-оптимальный метод | | | | | |
| ɛ | x | y |  | N |  |
| 0,1 | 0.6 | -0,681029 | 0,1 | 9 | 0,01 |
| 0,01 | 0,64 | -0,686198 | 0,01 | 99 | 0,001 |
| 0,001 | 0,643 | -0,686218 | 0,001 | 999 | 0,0001 |
| 0,0001 | 0,6426 | -0,686218 | 0,0001 | 9999 | 0,00001 |
| 0,00001 | 0,64264 | -0,686218 | 1e-05 | 99999 | 0,000001 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод перебора | | | | |
| ɛ | x | y |  | N |
| 0,1 | 0,6 | -0,681029 | 0,1 | 9 |
| 0,01 | 0,64 | -0,686198 | 0,01 | 99 |
| 0,001 | 0,643 | -0,686218 | 0,001 | 999 |
| 0,0001 | 0,6426 | -0,686218 | 0,0001 | 9999 |
| 0,00001 | 0,64264 | -0,686218 | 1e-05 | 99999 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод деления отрезка пополам | | | | |
| ɛ | x | y |  | N |
| 0,1 | 0,625 | -0,685311 | 0,0625 | 7 |
| 0,01 | 0,640625 | -0,686206 | 0,0078125 | 13 |
| 0,001 | 0,642578 | -0,686218 | 0,000976563 | 19 |
| 0,0001 | 0,642639 | -0,686218 | 6,10352e-05 | 27 |
| 0,00001 | 0,642639 | -0,686218 | 7,62939e-06 | 33 |

;

и т.д.

Графики эффективности пассивных и последовательных методов

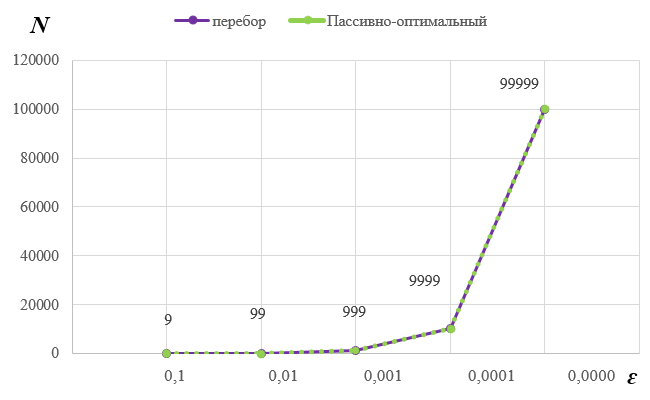


Рисунок 2 − График эффективности пассивных методов

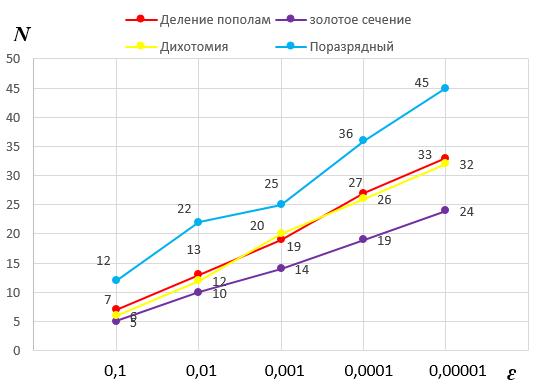


Рисунок 3 − График эффективности последовательных методов

Код программы

#include <iostream>

#include <math.h>

using namespace std;

double f(double x) //заданная функция

{

return pow(x, 4) - 1.5 \* atan(x);

}

void perebor(double a, double b)

{

int i, N;

double\* x, \* y, X = 0, Y = 0, Eгар = 0;

cout << " N = "; cin >> N;

x = new double[N];

y = new double[N];

for (i = 1; i <= N; i++)

{

x[i] = a + i \* ((b - a) / (N + 1));

}

for (i = 1; i <= N; i++)

{

y[i] = f(x[i]);

}

double yl = y[1];

int l = 1;

for (i = 1; i <= N; i++)

if (y[i] < yl)

{

yl = y[i];

l = i;

}

X = x[l];

Y = y[l];

Eгар = (b - a) / (N + 1);

cout << " х = " << X << "\n y = " << Y << "\n Егарант = " << Eгар << "\n N = " << N << "\n";

system("pause");

}

void passiv\_optimal(double a, double b)

{

int N, i, k;

double d, \* x, \* y, X = 0, Y = 0, Eгар = 0;

cout << " N = "; cin >> N; cout << " d = "; cin >> d;

x = new double[N];

y = new double[N];

if (N % 2 == 0)

{

k = N / 2;

for (i = 1; i <= k; i++)

{

x[2 \* i] = a + i \* (b - a) / (k + 1);

x[2 \* i - 1] = x[2 \* i] - d;

}

for (i = 1; i <= N; i++)

y[i] = f(x[i]);

double yl = y[1];

int l = 1;

for (i = 1; i <= N; i++)

if (y[i] < yl)

{

yl = y[i];

l = i;

}

X = x[l];

Y = y[l];

Eгар = (x[l + 1] - x[l - 1]) / 2;

}

else

{

for (i = 1; i <= N; i++)

x[i] = a + i \* (b - a) / (N + 1);

for (i = 1; i <= N; i++)

y[i] = f(x[i]);

double yl = y[1];

int l = 1;

for (i = 1; i <= N; i++)

if (y[i] < yl)

{

yl = y[i];

l = i;

}

X = x[l];

Y = y[l];

Eгар = (b - a) / (N + 1);

}

cout << " х = " << X << "\n y = " << Y << "\n Егарант = " << Eгар << "\n";

system("pause");

return;

}

void porazryad\_poisk(double a, double b)

{

int k;

double E, h, x0, y0, x1, y1, Eгар, X, Y;

cout << " Введите точность Е : "; cin >> E;

h = (b - a) / 4; //шаг перехода в следующую точку

x0 = a; //текущая точка

y0 = f(x0);

k = 1; //кол-во (вычисл-ых) экспериментов, понадобившихся для достижения заданной точности

cy:

x1 = x0 + h; //следующая точка

y1 = f(x1);

k++;

if (y0 > y1) //функция убывает

{

x0 = x1;

y0 = y1;

}

else goto sl; //функция начала расти

if ((x0 > a) && (x0 < b)) //находимся на отрезке

goto cy;

sl:

if (abs(h) <= E) //точность достигнута

{

X = x0;

Y = y0;

Eгар = h;

}

else //точность не достигнута, меняем направление

{

x0 = x1;

y0 = y1;

h = -h / 4;

goto cy;

}

cout << " х = " << X << "\n y = " << Y << "\n Егарант = " << Eгар << "\n N = " <<

k << "\n";

system("pause");

return;

}

void delenie\_popolam(double a, double b)

{

double E, Eгар;

cout << "Введите точность Е: "; cin >> E;

double x[5], y[4], yl, xl;

int i, N, l;

x[2] = (a + b) / 2; //середина отрезка неопределенности

y[2] = f(x[2]);

N = 1; //счетчик экспериментов

while (b - a > 2 \* E)

{

x[0] = a;

x[4] = b;

x[1] = (a + x[2]) / 2;

y[1] = f(x[1]);

x[3] = (x[2] + b) / 2;

y[3] = f(x[3]);

N += 2;

/\*поиск минимума\*/

yl = y[1]; xl = x[1]; l = 1;

for (i = 2; i < 4; i++)

if (yl > y[i]) { yl = y[i]; xl = x[i]; l = i; }

a = x[l - 1];

b = x[l + 1];

x[2] = xl;

y[2] = yl;

}

Eгар = (b - a) / 2; //достигнутая гарантированная точность решения

cout << " x = " << x[2] << "\n y = " << y[2] << "\n N = " << N << "\n Егарант = " << Eгар << "\n";

system("pause");

}

void dihotomiya(double a, double b)

{

double E, Eгар;

cout << "Введите точность Е: "; cin >> E;

double d = E / 10, x, y1, y2;

int N; N = 0; //счетчик экспериментов

while (b - a > 2 \* E)

{

x = (a + b) / 2;

y1 = f(x - d);

y2 = f(x + d);

N += 2;

(y1 < y2 ? b = x + d : a = x - d);

}

x = (a + b) / 2;

y1 = f(x);

Eгар = (b - a) / 2; //достигнутая гарантированная точность решения

cout << " x = " << x << "\n y = " << y1 << "\n N = " << N << "\n Егарант = " << Eгар << "\n";

system("pause");

}

void zolotoe\_sechenie(double a, double b) //метод золотого сечения

{

int k = 0; //кол-во (вычисл-ых) экспериментов, понадобившихся для достижения заданной точности

double E;

cout << "Введите точность Е: "; cin >> E;

double t = (1 + sqrt(5)) / 2; //пропорция t золотое сечение

double x1, x2, y1, y2, Eгар, x, y, Np;

Np = ceil((log((b - a) / (2 \* E)) / log(t)) + 1); //расчетное число экспериментов для достижения заданной точности

x1 = b - (b - a) / t; //первый выбор точки x, делящей отрезок [a;b] в пропорции золотого сечения

y1 = f(x1);

x2 = a + (b - a) / t;//второй выбор точки x, делящей отрезок [a;b] в пропорции золотого сечения

y2 = f(x2);

k = 2;

do

{

if (y1 < y2) {

b = x2; x2 = x1; y2 = y1; x1 = a + b - x2;//определение симметричной точки

y1 = f(x1);

}

else {

a = x1; x1 = x2; y1 = y2; x2 = a + b - x1;//определение симметричной точки

y2 = f(x2);

}

k++;

} while ((b - a) > 2 \* E \* t);

//отрезок локализации:

{ if (y1 < y2) b = x2;

else a = x1;

}

x = (a + b) / 2; //приближенное решение (середина последнего отрезка локализации)

y = f(x);

Eгар = (b - a) / 2; //достигнутая гарантированная точность решения

cout << " x = " << x << "\n y = " << y << "\n N = " << k << "\n N расчет = " << Np <<

"\n Е гарант = " << Eгар << "\n";

system("pause");

}

void main()

{

double a = 0, b = 1;

setlocale(LC\_ALL, "Russian");

int j;

while (1)

{

system("cls");

cout << "1. Метод перебора " << endl;

cout << "2. Пассивный оптимальный метод " << endl;

cout << "3. Поразрядный поиск " << endl;

cout << "4. Деление отрезка пополам " << endl;

cout << "5. Дихотомия " << endl;

cout << "6. Золотое сечение " << endl;

cout << "7. Выход из программы." << endl;

cout << " Ваш выбор (1-7): ";

cin >> j;

switch (j)

{

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

case 2:passiv\_optimal(a, b); break;

case 3:porazryad\_poisk(a, b); break;

case 4:delenie\_popolam(a, b); break;

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

case 6:zolotoe\_sechenie(a, b); break;

case 7: cout << " Конец работы.\n"; system("pause"); return;

default: cout << " Ошибка: нет такого пункта меню.\n"; break;

}

}

}

Блок-схемы функций методов



Рисунок 4 − Блок-схема метода перебора



Рисунок 5 − Блок-схема пассивно-оптимального метода



Рисунок 6 − Блок-схема поразрядного поиска



Рисунок 7 − Блок-схема метода деления отрезка пополам



Рисунок 8 − Блок-схема метода дихотомии



Рисунок 9 − Блок-схема метода золотого сечения

Вывод: в ходе лабораторной работы были сравнены пассивный оптимальный алгоритм, поразрядный метод и метод золотого сечения. Для заданное целевой функции на заданном отрезке локализации лучшие результаты дал метод золотого сечения.