|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 13

Выполнил: Мусина К. Р.,

студент группы ИУ8-32

Проверил: Коннова Н. С.,

доцент каф. ИУ8

г. Москва,

2020 г.

# 1. Цель работы

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

# 2. Постановка задачи

Унимодальная функция:

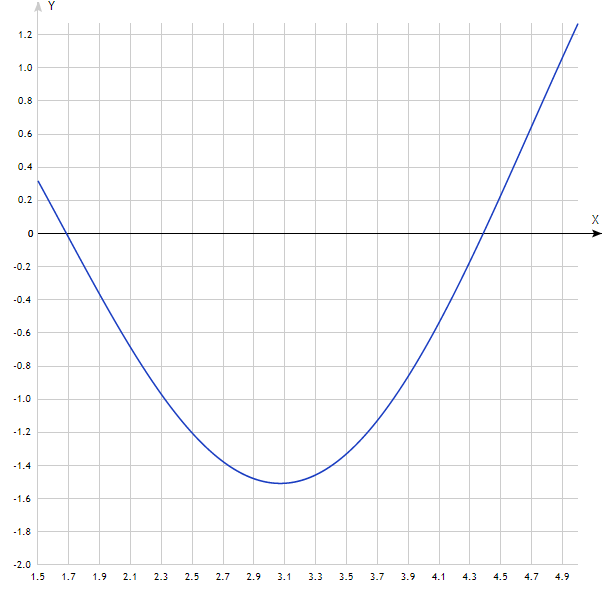
Отрезок поиска:

Методы поиска: оптимальный пассивный, дихотомия.

Наибольшая длина интервала неопределенности:

# 3. Ход работы

Рисунок 1 демонстрирует график унимодальной функции.



**Рисунок 1** – график функции.

**Реализация оптимального пассивного поиска.**

Точки расположены равномерно по отрезку, следовательно, координата точки с номером :

**Таблица 1** – применение оптимального пассивного поиска

| N | x | s |

|-----|-----------|------------|

| 1 | 3.25 | +-1.75 |

| 2 | 2.666667 | +-1.16667 |

| 3 | 3.25 | +-0.875 |

| 4 | 2.9 | +-0.7 |

| 5 | 3.25 | +-0.583333|

| 6 | 3 | +-0.5 |

| 7 | 3.25 | +-0.4375 |

| 8 | 3.05556 | +-0.388889|

| 9 | 2.9 | +-0.35 |

| 10 | 3.09091 | +-0.318182|

| 11 | 2.95833 | +-0.291667|

| 12 | 3.11538 | +-0.269231|

| 13 | 3 | +-0.25 |

| 14 | 3.13333 | +-0.233333|

| 15 | 3.03125 | +-0.21875 |

| 16 | 3.14706 | +-0.205882|

| 17 | 3.05556 | +-0.194444|

| 18 | 3.315789 | +-0.184211|

| 19 | 3.075 | +-0.175 |

| 20 | 3 | +-0.166667|

| 21 | 3.09091 | +-0.159091|

| 22 | 3.02174 | +-0.152174|

| 23 | 3.10417 | +-0.145833|

| 24 | 3.04 | +-0.14 |

| 25 | 3.11538 | +-0.134615|

| 26 | 3.05556 | +-0.12963 |

| 27 | 3.125 | +-0.125 |

| 28 | 3.06897 | +-0.12069 |

| 29 | 3.01667 | +-0.116667|

| 30 | 3.08065 | +-0.112903|

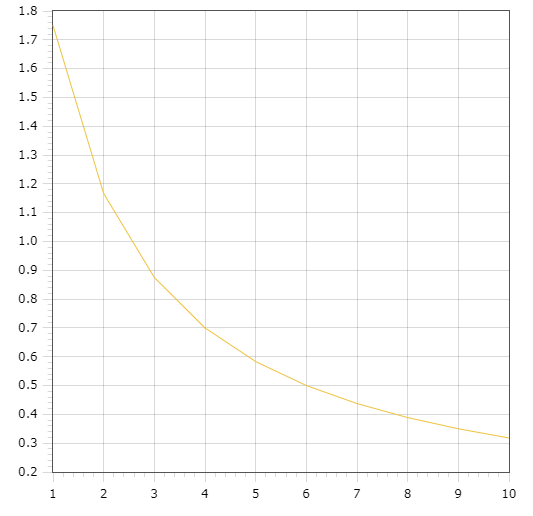
| 31 | 3.03125 | +-0.109375|

| 32 | 3.09091 | +-0.106061|

| 33 | 3.4412 | +-0.102941|

| 34 | 3.1 | +-0.1 |

При N = 34 достигается заданная неопределенность в методе оптимального пассивного поиска.



**Рисунок 2** – график зависимости погрешности от числа измерений (первые 10) метода опт. пассивного поиска.

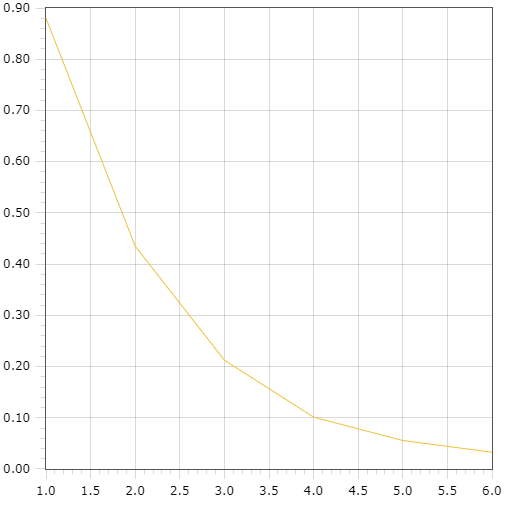
**Реализация метода дихотомии.**

Отрезок поиска делится пополам точкой x. Вычисляются значения функции на границах окрестности точки x: . Исключается левая половина, если значение функции в точке левой границы окрестности больше, чем в правой. Иначе, исключается правая половина. Эти действия повторяются до тех пор, пока отрезок поиска будет больше отрезка неопределенности.

**Таблица 2** – последовательный поиск (метод дихотомии)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Начало | Конец | Длина интервала | f(Xleft) | f(Xright) |
| 1.5 | 5 | 3.5 | 0.317566 | 1.26629 |
| 1.5 | 3.26 | 1.76 | -1.47978 | -1.47278 |
| 2.39 | 3.26 | 0.87 | -1.08281 | -1.05885 |
| 2.835 | 3.26 | 0.425 | -1.45418 | -1.4448 |
| 3.0575 | 3.26 | 0.2025 | -1.50757 | -1.50666 |
| 3.0575 | 3.16875 | 0.11125 | -1.50181 | -1.4987 |
| 3.0575 | 3.12312 | 0.065625 | -1.50672 | -1.50507 |

Минимум функции в точке: , а значение функции в этой точке

**

**Рисунок 3** – график зависимости погрешности от числа измерений метода дихотомии.

# 4. Выводы

В данной лабораторной работе был найден минимум унимодальной функции с помощью метода оптимального пассивного поиска и метода дихотомии. Из приведенного выше хода работы можно сделать вывод, что метод дихотомии эффективнее метода пассивного поиска, т.к. требует меньшего количества контрольных точек.

**5. Контрольный вопрос**

В чем суть метода золотого сечения?

Исходный отрезок делится на две части по правилу золотого сечения (отношение длины всего отрезка к большей части = отношению длины большей части к меньшей).

Проверяется значение в этих точках. Запоминается минимальное значение функции и точка, ему соответствующая.

Допустим на отрезке ab золотое сечение образуют точки c и d (ab/cb = cb/ac), тогда, если f(c)<f(x), то отрезок db отбрасывается, как заведомо больший минимума, и делится уже новый отрезок ad по принципу, описанному выше, пока величина отрезка минимума, больше заданной погрешности.

От метода дихотомии отличается тем, что на каждой новой итерации нужно вычислять только одно значение оптимального критерия.

# Приложение 1. Код программы lab-01.cpp.

|  |
| --- |
|  |
|  | #include <vector> |
|  | #include <iostream> |
|  | #include <algorithm> |
|  | #include <map> |
|  | #include<cmath> |
|  |  |
|  | double used\_f(double x) { |
|  | return (2\*cos(x)+log10(x)); |
|  | } |
|  |  |
|  | int main() { |
|  |  |
|  | const float eps = 0.1; |
|  | double b = 5.0; |
|  | double a = 1.5; |
|  | unsigned int N = 1; |
|  | double Xi; |
|  | double s; |
|  | double min; |
|  | std::map<double, double> values; |
|  | std::cout << "passive search\n"; |
|  | do { |
|  | for (auto k = 1; k < N + 1; ++k) { |
|  | Xi = ((b - a) \* k / (N + 1)) + a; |
|  | values.insert(std::make\_pair(used\_f(Xi), Xi)); |
|  | } |
|  | s = (b - a) / (N + 1); |
|  | std::map<double, double> ::iterator it = values.begin(); |
|  | std::cout << "N = " << N << " = " << it->second << " +- " << s  << "\n"; |
|  | values.clear(); |
|  | ++N; |
|  | } while (s > eps); |
|  |  |
|  | const float delta = 0.01; |
|  | double Xleft, Xright; |
|  | std::cout << "\ndihotomiya\n"; |
|  | std::cout << "a=" << a << " b=" << b << " b-a=" << (b - a) << " "  << used\_f(a) << " " << used\_f(b) << "\n"; |
|  | double a1, b1; |
|  | do { |
|  | Xleft = 0.5 \* (b + a) - delta; |
|  | Xright = 0.5 \* (b + a) + delta; |
|  | values.insert(std::make\_pair(used\_f(Xleft), Xleft)); |
|  | values.insert(std::make\_pair(used\_f(Xright), Xright)); |
|  | std::map<double, double> ::iterator it = values.begin(); |
|  | a1 = it->second; |
|  | ++it; |
|  | b1 = it->second; |
|  | if (a1 > b1) |
|  | a = a1; |
|  | else |
|  | b = b1; |
|  | --it; |
|  | std::cout << "a=" << a << " b=" << b << " b-a=" << (b - a) |
|  | << "\nf(Xleft)=" << it->first; |
|  | ++it; |
|  | s = 0.5 \* (b - a); |
|  | std::cout << " f(Xright)=" << it->first << "\nmin:" |
|  | << (a + b) / 2 << " +- " << s << "\n"; |
|  | values.clear(); |
|  | } while ((b - a) > eps); |
|  | system("pause"); |
|  | return 0; |
|  |  |
|  | } |
|  |  |

**Ссылка на git-репозиторий:** <https://github.com/CamilaMusina/tsisa-lab-01>