**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»

КАФЕДРА «ТЕОРЕТИЧЕСКАЯ ИНФОРМАТИКА И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ»

# Отчёт по лабораторной работе № 7

# «Одномерная численная оптимизация»

***по курсу***

***«Численные методы»***

Студент группы ИУ9-61Б Бакланова А.Д.

Преподаватель Домрачева А.Б.

*Москва, 2020 г.*

**Цель работы**

Целью данной работы является реализация 3 методов одномерной численной оптимизации: метод дихотомии, метод золотого сечения, метод парабол.

**Задание**

Найти точку минимума $x^*$ и минимум $f^* = f(x^*)$ функции $f(x)$ на отрезке $[a, b]$ с точностью $\varepsilon$.

**Теоретические сведения:**

*Метод дихотомии (деления отрезка пополам):*

Поиск минимума начинается с выбора двух точек $x_1$ и $x_2$, лежащих на расстоянии $\delta$ от середины отрезка $x = (a + b)/2$ слева и справа; $x_1 = x - \delta, \quad x_2 = x + \delta$. Здесь $\delta$ — постоянная, являющаяся параметром метода, $0 < \delta < (b - a)/2$. После выбора $x_1$ и $x_2$ вычисляют значения $f(x_1)$ и $f(x_2)$ и сравнивают их между собой. Если $f(x_1) \leq f(x_2)$, то полагают $a_1 = a, \quad b_1 = x_1$. Иначе $a_1 = x_1, \quad b_1 = b$.

Отправляясь от $[a_1, b_1]$, опять находят точки $x_1$ и $x_2$:

$x_1 = (a_1 + b_1)/2 - \delta, \quad x_2 = (a_1 + b_1)/2 + \delta$ по описанному алгоритму определяет отрезок $[a_2, b_2]$ и т.д. Процесс деления отрезка пополам продолжают до тех пор, пока не получится отрезок $[a_k, b_k]$ длины $b_k - a_k \leq 2\varepsilon$. За точку минимума принимают $x = (a_k + b_k)/2$

*Метод золотого сечения:*

Алгоритм метода золотого сечения отличается от алгоритма метода дихотомии лишь выбором точек $x_1$ и $x_2$.

Золотым сечением отрезка называется его деление на две неровные части так, что отношение длины всего отрезка к длине большей части равняется отношению большей части к длине меньшей части отрезка. Золотое сечение отрезка $[a, b]$ производится двумя точками $x_1 = a + (3 - \sqrt{5})(b - a)/2$ и $x_1 = a + (\sqrt{5} - 1)(b - a)/2$, расположенными симметрично относительно середины отрезка.

$a < x_1 < x_2 < b$

$x_2 - x_1 < x_1 - a = b - x_2$

$\dfrac{x_2 - a}{x_1 - a} = \dfrac{x_1 - a}{x_2 - x_1}$ Аналогично точка $x_2$ — золотое сечение отрезка $[x_1, b]$.

*Метод парабол:*

В отличие от двух предыдущих методов требует, чтобы функция была дважды непрерывно дифференцируема. Пусть на k-м шаге метода мы получили приближение $x_k$ к точке минимума $x^*$. Разложим функцию в ряд Тейлора:

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{f''(x_k)}{2}(x - x_k)^2$$

Первая часть равенства — многочлен второй степени (парабола). Минимум этой параболы находится в точке

$x_{k+1} = x_k - f'(x_k)/f''(x_k)$

Таким образом, (k+1)-е приближение к точке минимума строится по рекуррентной зависимости, Процесс поиска минимума продолжается пока не будет выполнено равенство $|x_k - x_{k-1}| \leq \varepsilon$

**Дано:**

Функция $f(x)$ на отрезке $[a, b]$ с точностью $\varepsilon$

**Входные данные:**

$f(x) = tg(x) - 2sin(x)$

$[0, \frac{\pi}{4}]$

$\varepsilon = 0.03$

**Программная реализация:**

OptimizationMethods.java

```
import static java.lang.Math.*;

public class OptimizationMethods {
    public static void main(String[] args) {
        double a = 0;
        double b = PI/4;
        double eps = 0.03;
        dihotomy_method(a,b,eps);
        gold_method(a,b,eps);
        parabola_method(a,b,eps);
    }

    public static void dihotomy_method(double a, double b, double e) {
        double x1, x2 = 0;
        double x = 0;

        while (b-a > 2*e) {
            x = (a+b)/2;
            x1 = x - e;
            x2 = x + e;
            if (f(x1) <= f(x2))
                b = x1;
            else if (f(x1) > f(x2)) {
                a = x2;
            }

        }
        double res = x;
        System.out.println("dihonomy_method: "+res+" "+f(res)*180/PI);
    }
```

OptimizationMethods › parabola_method()

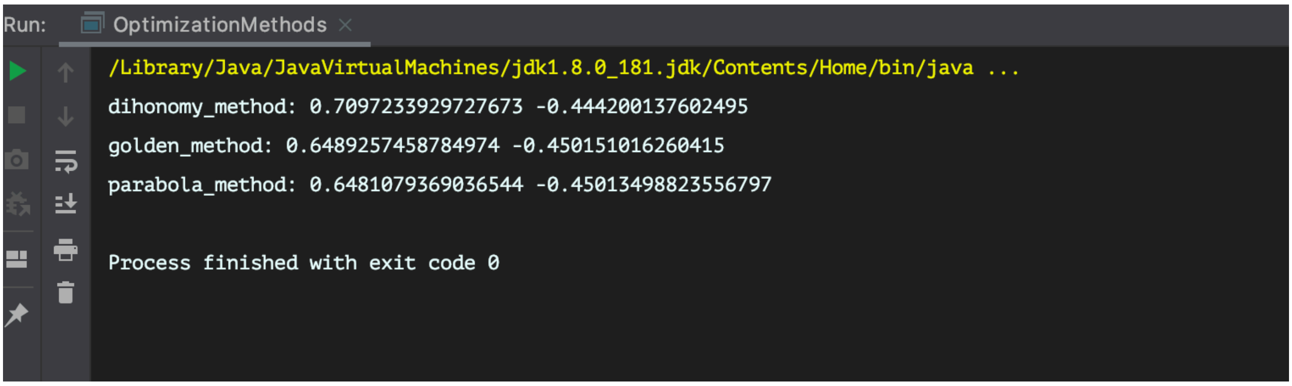
OptimizationMethods.java

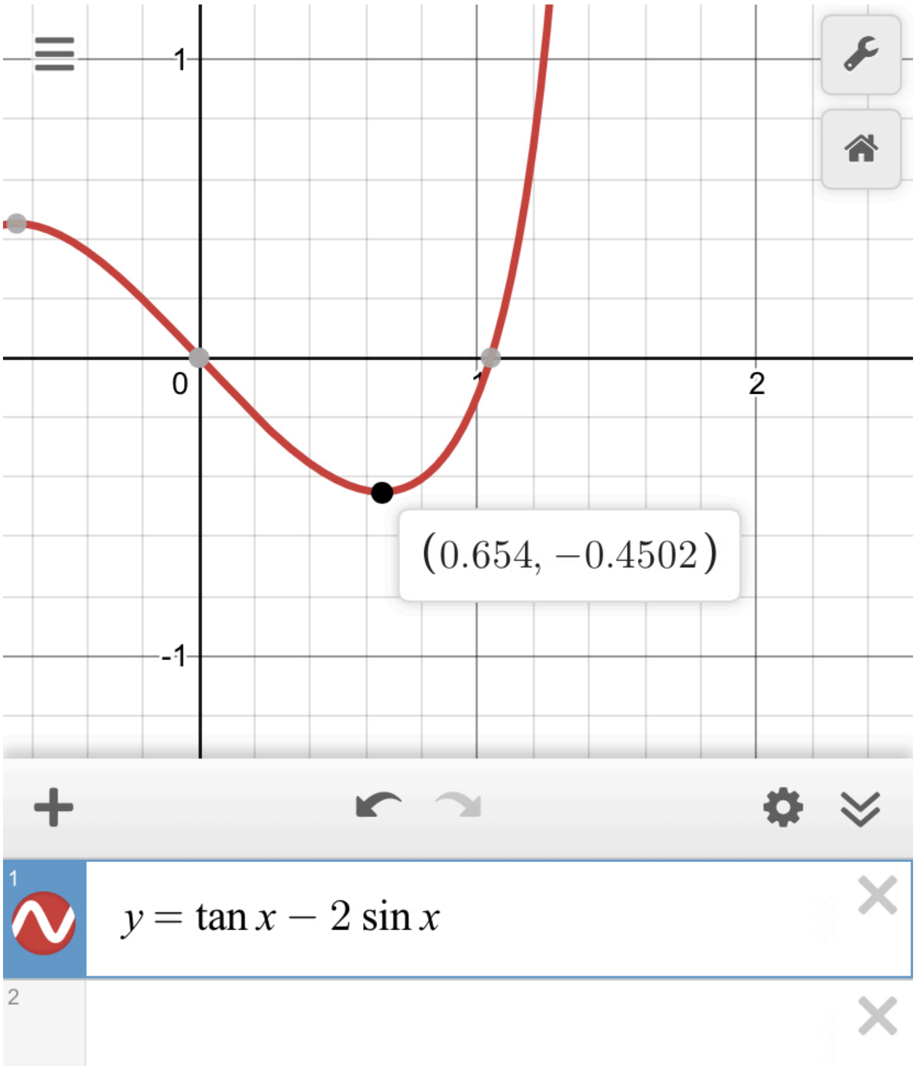
```java
    public static double f(double x) {
        return (Math.tan(x) - 2*Math.sin(x))*PI/180;
    }

    public static void gold_method(double a, double b, double e) {
        double x1, x2 = 0;
        double x = 0;
        double f1,f2 = 0;

        x = (a+b)/2;
        x1 = a+(3-sqrt(5))*(b-a)/2;
        x2 = a+(sqrt(5)-1)*(b-a)/2;
        f1 = f(x1);
        f2 = f(x2);
        while (b-a > 2*e) {

            if (f1 < f2) {
                b = x2;
                x2 = x1;
                f2 = f1;
                x1 = a + (3 - sqrt(5)) * (b - a) / 2;
                f1 = f(x1);
            }
            else {
                a = x1;
                x1 = x2;
                f1 = f2;
                x2 = a+(sqrt(5)-1)*(b-a)/2;
                f2 = f(x2);
```

OptimizationMethods.java

```java
            }
        }
        double res = (x1+x2)/2;
        System.out.println("golden_method: "+res+" "+f(res)*180/PI);
    }

    public static void parabola_method(double a, double b, double e) {
        double x1, x2 = 0;
        double x = 0;

        x = (a+b)/2;
        x1 = x - df(x)/ddf(x);

        while (x1 - x > e) {
            x = x1;
            x1 = x - df(x)/ddf(x);

        }
        double res = x1;
        System.out.println("parabola_method: "+res+" "+f(res)*180/PI);
    }

    public static double df(double x) {
        return -2*cos(x)+ 1/(cos(x)*cos(x));
    }

    public static double ddf(double x) {
        return 2*sin(x)+ 2*sin(x)/(cos(x)*cos(x));
    }
```

**Результат:**


Run:
OptimizationMethods
/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java ...
dihonomy_method: 0.7097233929727673 -0.444200137602495
golden_method: 0.6489257458784974 -0.450151016260415
parabola_method: 0.6481079369036544 -0.45013498823556797
Process finished with exit code 0


1
0
2
-1
(0.654, −0.4502)
y = tan x − 2 sin x

**Вывод:**

Итого, в результате лабораторной работы были реализованы методы одномерной численной оптимизации: метод дихотомии, метод золотого сечения, метод парабол. Для моего варианта самый лучший результат дали метод золотого сечения и метод параболы. Оба метода дали примерно одинаковый результат.