Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Лабораторная работа №3.1

«Методы одномерного поиска. (I часть).

Методы стягивающихся отрезков. Вариант 5.»

по курсу «Методы оптимизации»

Выполнила:

студентка 4 курса,

группы ИУ9-82

Козлова А. А.

Проверил:

Каганов Ю. Т.

2018 г

Цель работы.

1. Изучение алгоритмов одномерного поиска.
2. Разработка программ реализации алгоритмов одномерного поиска.
3. Вычисление экстремумов функции.

Постановка задачи:

Требуется найти безусловный минимум функции $f(x)$ одной переменной, т.е. такую точку $x^* \in R$, что $f(x^*) = min_{x \in R} f(x)$.

Поставленная задача одномерной минимизации может быть решена с помощью необходимых и достаточных условий безусловного экстремума. Однако проблема получения решения уравнения $\frac{df(x)}{dx} = 0$ может оказаться достаточно сложной. Более того, в практических задачах функция $f(x)$ может быть не задана в аналитическом виде или часто неизвестно, является ли она дифференцируемой. Поэтому получение численного решения поставленной задачи является актуальным.

Задание 1.

Найти экстремум функции

$f(x)=5x^6-36x^5-\frac{165}{2}x^4-60x^3+36$ **;** $x_0=-13.$

методами:

- **Половинного деления.**
- **Золотого сечения.**
- **Фибоначчи.**

1) Метод половинного деления

$x=7.55224609375$

$f(x)=-249296.76568738808$

2) Метод золотого сечения

$x=7.551436228696057$

$f(x)=-249296.67587221522$

3) Метод Фибоначчи

$x=7.560790273556231$

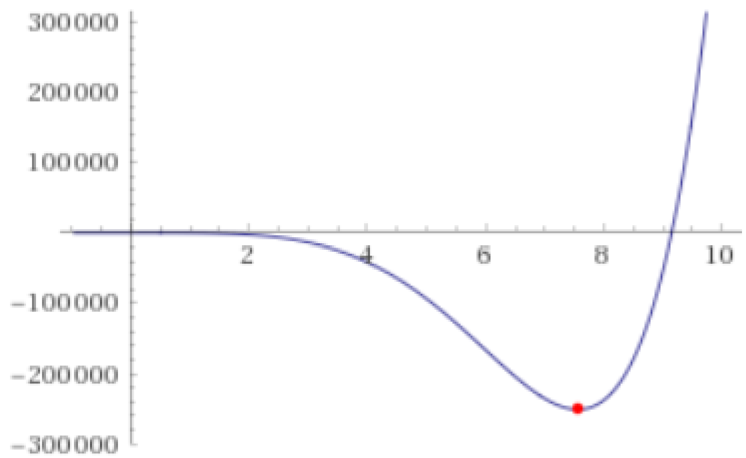
$f(x)=-249292.945944976$

Задание 2.

Найти все стационарные точки и значения функции $f(x)=5x^6-36x^5-\frac{165}{2}x^4-60x^3+36$; $x_0=-13.$ **соответствующие этим точкам.**

Для решения использовался сервис Wolphram Mathematica.

График функции:



Стационарная точка:

$x = -7.56001$, $f(x) = -250927$.

Задание 3.

Оценить скорость сходимости указанных алгоритмов и сравнить по времени получение результата оптимизации для разных методов.

1) Алгоритм Свенна:

Итервал: (1.0, 10.0)

2) Метод половинного деления

$x = 7.55224609375$

$f(x) = -249296.76568738808$

Количество итераций: 10

Для метода деления интервала пополам характеристика относительного уменьшения начального интервала неопределенности находится по формуле

$R(N) = \frac{1}{2^{\frac{N}{2}}}$, где N - количество вычислений функции.

Следовательно, $R(N) = 0.03125$

3) Метод золотого сечения

$x = 7.551436228696057$

$f(x) = -249296.67587221522$

Количество итераций: 15

Для метода золотого сечения характеристика относительного уменьшения начального интервала неопределенности находится по формуле

$R(N) = (0.618)^{N-1}$, где N - количество вычислений функции.

Следовательно, $R(N) = 0{,}00118$

4) Метод Фибоначчи

$x = 7.560790273556231$

$f(x) = -249292.945944976$

Количество итераций: 11

Для метода Фибоначчи характеристика относительного уменьшения начального интервала неопределенности находится по формуле

$R(N) = \frac{1}{F_N}$, где N - количество вычислений функции.

Следовательно, $R(N) = 0{,}01123$

На основе полученных данных можно сделать вывод, что скорость сходимости наиболее высокая у метода половинного деления, а у метода золотого сечения самая низкая.

Задание 4.

Реализовать алгоритмы программированием на одном из языков высокого уровня.

Язык реализации: Java

<u>Алгоритм Свенна</u>

```
while (true){
      if (f(x_0 - t) <= f(x_0) && f(x_0) >= f(x_0 + t)){
         t++;
      }
      if (f(x_0 - t) >= f(x_0) && f(x_0) <= f(x_0 + t)) {
         a_0 = x_0 - t;
         b_0 = x_0 + t;//начальный интервал неопределенности
         break;
      } else {
         if (f(x_0 - t) >= f(x_0) && f(x_0) >= f(x_0 + t)) {
            delta = t;
            a_0 = x_0;
            x_0 = x_0 + t;
         }
         if (f(x_0 - t) <= f(x_0) && f(x_0) <= f(x_0 + t)) {
            delta = -t;
            b_0 = x_0;
            x_0 = x_0 - t;
         }
         while (true) {
            double x_k = x_0 + 2 * delta;
            if (f(x_k) < f(x_0) && delta == t) {
               a_0 = x_0;
            }
            if (f(x_k) < f(x_0) && delta == -t) {
               b_0 = x_0;
            }
            if (f(x_k) >= f(x_0)) {
               break;
            }
            x_0 = x_k;
         }
         break;
      }
}
```

Метод половинного деления

```
public static double halfSectionMethod(double a, double b, double eps){
    double x_middle = (a+b)/2;
     double L = Math.abs(b-a);
     while (L > eps) {
        double y = a + L / 4;
        double z = b - L / 4;
        double f_x = f(x_middle);
        double f_y = f(y);
        double f_z = f(z);
         if (f_y < f_x) {
            b = x_middle;
            x_middle = y;
         } else {
            if (f_z < f_x) {
               a = x_middle;
               x_middle = z;
            } else {
               a = y;
               b = z;
            }
         }
         L = Math.abs(b-a);
         it++;
      }
      return x_middle;
   }
```

Метод золотого сечения

```
public static double  goldenSectionMethod(double a, double b, double eps){
   double z,y;
     double PHI = (1.+Math.sqrt(5.))/2.;
        while (Math.abs(b - a) > eps){
           z = b - (b - a) / PHI;
           y = a + (b - a) / PHI;
           if (f(y) <= f(z))
              a = z;
           else
               b = y;
            it++;
         }
```

```
        return (a + b) / 2;
    }
```

<u>Метод Фибоначчи</u>

```
public static double methodFibonacci(double a, double b, double eps){
    double L = Math.abs(b-a);
     double y,z;
     int N = 3;
     ArrayList<Double> fibArr = new ArrayList<>(Arrays.asList(new Double[]{1.,1.,2.,3.}));
     for (double f1 = 2., f2 = 3.; fibArr.get(fibArr.size()-1) < L/eps; ++N) {
         fibArr.add(f1+f2);
         f1 = f2;
         f2 = fibArr.get(fibArr.size()-1);
      }
      for (int i = 1; i != N-3; i++) {
          y = a + fibArr.get(N - i - 1) / fibArr.get(N - i + 1) * (b - a);
          z = a + fibArr.get(N - i) / fibArr.get(N - i + 1) * (b - a);
          if (f(y) <= f(z))
              b = z;
          else
              a = y;
          it++;
      }
      return (a + b) / 2;
    }
```