Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

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

Лабораторная работа №3.2

«Методы одномерного поиска. (II часть).

Методы интерполяции. 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 $f(x^*)=min_{x\in R} f(x)$.

Задание 1.

Найти экстремум функции

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

методами:

- **Квадратичной интерполяции.**
- **Кубической интерполяции.**

1) Метод квадратичной интерполяции.

Минимум:

$$x=7.556209$$

$$f(x)=-249296.08$$

2) Метод кубической интерполяции

Минимум:

$$x=7.560005$$

$$f(x)=-249293.66$$

Задание 2.

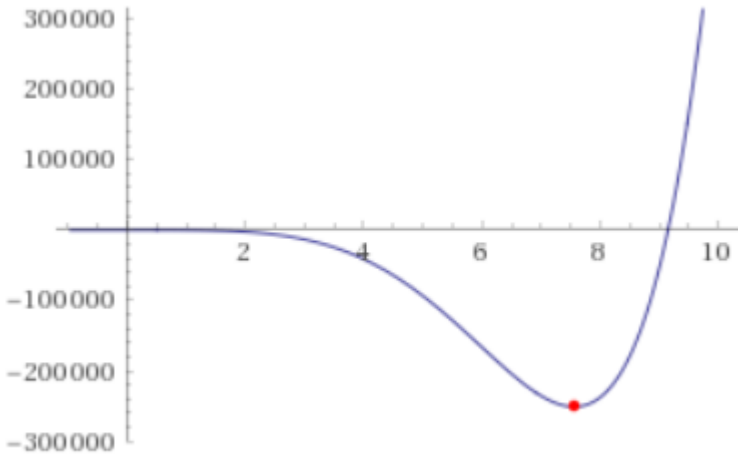
Найти все стационарные точки и значения функции

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

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) Метод квадратичной интерполяции

$x = 7.556209$

$f(x) = -249296.08$

Количество итераций 387

2) Метод кубической интерполяции

$x = 7.560005$

$f(x) = -249293.66$

Количество итераций 20

На основе полученных данных можно сделать вывод, что скорость сходимости наиболее высокая у метода кубической интерполяции.

Задание 4.

Реализовать алгоритмы программированием на одном из языков высокого уровня.

Язык реализации: Java

Метод квадратичной интерполяции

```
1.public static float quadraticInterpolation(float a1, float eps){
2.     float step = 0.01f;
3.     float a2, a3, f_min, alfa, a_min;
4.     float cond1, cond2;
5.
6.     while (true) {
7.       a2 = a1 + step;
8.       if (f(a1) > f(a2)) {
9.         a3 = a1 + 2 * step;
10.       } else {
11.         a3 = a1 - 2 * step;
12.       }
13.       while (true){
14.         it++;
15.         a_min = min(a1, a2, a3);
16.         f_min = f(a_min);
17.         float den = ((a2 - a3) * f(a1) + (a3 - a1) * f(a2) + (a1 - a2) * f(a3));
18.         if (Math.abs(den) > 0.1) {
19.           alfa = (float) (0.5 * ((Math.pow(a2, 2) - Math.pow(a3, 2)) * f(a1) +
```

```
                (Math.pow(a3, 2) - Math.pow(a1, 2)) * f(a2) +
                (Math.pow(a1, 2) - Math.pow(a2, 2)) * f(a3)) /
                den);
          } else {
            a1 = a_min + step; //перейти на ш 2
            break;
          }
          cond1 = (f_min - f(alfa)) / f(alfa);
          cond2 = (a_min - alfa) / alfa;
          if (Math.abs(cond1) > eps || Math.abs(cond2) > eps) {
            if (a1 <= alfa && alfa <= a3) {
              if (alfa < a2) {
                a3 = a2;
                a2 = alfa;
              } else {
                a1 = a2;
                a2 = alfa;
              }
            } else {
              a1 = alfa;
              break;
            }
          } else {
            return alfa;
          }
        }
      }
}
```

Метод кубической интерполяции

```
public static float qubicInterpolation(float a1, float eps){
    ArrayList<Float> points = new ArrayList<>();
    float a2 = a1;
    float alfa = 0.0f;
    float step = 0.02f;
    if (f_der(a1)< 0) {
      int i = 0;
      do {
        a1 = a2;
         a2 += pow(2, i) * step;
         ++i;
         it++;
       } while (f_der(a1) * f_der(a2) > 0.0f);
```

```
14.     } else {
15.       int i = 0;
16.       do {
17.         a1 = a2;
18.         a2 -= pow(2, i) * step;
19.         ++i;
20.         it++;
21.       } while (f_der(a1) * f_der(a2) > 0.0f);
22.     }
23.     float z,w,mu, cond1,cond2;
24.     while (true){
25.       it++;
26.       z = 3 * (f(a1) - f(a2)) / (a2 - a1) + f_der(a1) + f_der(a2);
27.       if (a1 < a2) {
28.         w = (float)Math.sqrt(pow(z, 2) - f_der(a1) * f_der(a2));
29.       } else {
30.         w = -(float)Math.sqrt(pow(z, 2) - f_der(a1) * f_der(a2));
31.
32.       }
33.       mu = (f_der(a2) + w - z) / (f_der(a2) - f_der(a1) + 2 * w);
34.
35.       if (mu < 0.0f) {
36.         alfa = a2;
37.       }
38.       if (0.0f <= mu && mu <= 1.0f) {
39.         alfa = a2 - mu * (a2 - a1);
40.       }
41.       if (mu > 1.f) {
42.         alfa = a1;
43.       }
44.       while (f(alfa) >= f(a1) && Math.abs((alfa - a1) /alfa) > eps) {
45.         alfa = alfa -(alfa - a1) / 2;
46.       }
47.       cond1 = f_der(alfa);
48.       cond2 = (alfa - a1)/alfa;
49.
50.       if (Math.abs(cond1) > eps || Math.abs(cond2) > eps ) {
51.         if (f_der(alfa) * f_der(a1) <= 0.0f) {
52.           a2 = a1;
53.           a1 = alfa;
54.         }
55.         if (f_der(alfa) * f_der(a2) <= 0.0f) {
56.           a1 = alfa;
57.         }
```

```
58.       }
59.       else {
60.          return alfa;
61.       }
62.
63.    }
64.}
65.
```