Отчет по лабораторной работе № 1
Методы оптимизаций

Прямые методы одномерной оптимизации

Скроба Дмитрий M3234

March 2021

1. Постановка задачи

- Реализовать алгоритмы одномерной оптимизации функции:
  1. Метод дихотомии
  2. Метод золотого сечения
  3. Метод Фиббоначи
  4. Метод парабол
  5. Комбинированный метод Брента
- Протестировать на задаче оптимизации унимодальной функции:

2. Результаты исследований

- **Аналитическое решение функции:**
  $f(x) = e^{3x} + 5e^{-2x} \to min : x \in [0; 1] \Leftrightarrow f'(x) = 0$
  $f'(x) = 0 \Leftrightarrow 3 * e^{3x} - 10e^{-2x} = 0$
  $e^{5x} = \frac{10}{3} \Leftrightarrow x = \frac{1}{5} * \ln \frac{10}{3}$

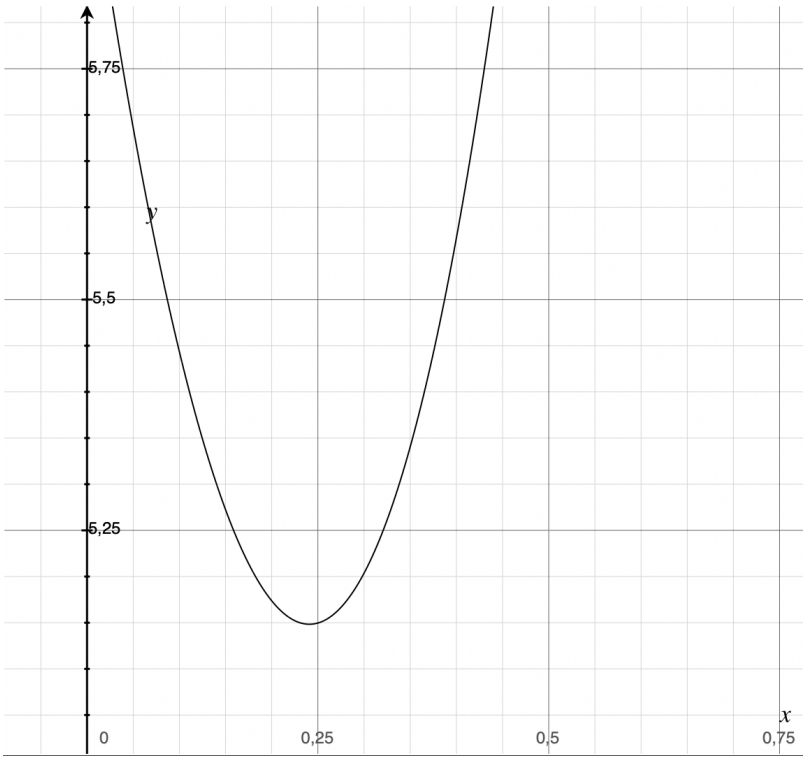


Рис. 1: Аналитическое решение функции и ее график

- **Метод Дихотомии**
  Начальные условия: $left = 0$, $right = 0$, $\epsilon = 0.001$, $\delta = 0.00001$
  Полученный результат: $0,2412$
  Количество операций: 10

№	left-border	right-border	left	left-fun	right	right-fun
1	0,0000	1,0000	0,5000	6,3210	0,5000	6,3211
2	0,0000	0,5000	0,2500	5,1497	0,2500	5,1497
3	0,0000	0,2500	0,1250	5,3490	0,1250	5,3490
4	0,1250	0,2500	0,1875	5,1915	0,1875	5,1915
5	0,1875	0,2500	0,2187	5,1558	0,2188	5,1558
6	0,2187	0,2500	0,2344	5,1490	0,2344	5,1490
7	0,2344	0,2500	0,2422	5,1484	0,2422	5,1484
8	0,2344	0,2422	0,2383	5,1484	0,2383	5,1484
9	0,2383	0,2422	0,2402	5,1483	0,2402	5,1483
10	0,2402	0,2422	0,2412	5,1483	0,2412	5,1483

- **Метод Золотого сечения**
  Начальные условия: $left = 0$, $right = 0$, $\epsilon = 0.001$, $\delta = 0.00001$
  Полученный результат: $0,2413$
  Количество операций: 14

№	left-border	right-border	left	left-fun	right	right-fun
1	0,0000	1,0000	0,3820	5,4744	0,6180	7,8386
2	0,0000	0,6180	0,2361	5,1487	0,3820	5,4744
3	0,0000	0,3820	0,1459	5,2837	0,2361	5,1487
4	0,1459	0,3820	0,2361	5,1487	0,2918	5,1893
5	0,1459	0,2918	0,2016	5,1717	0,2361	5,1487
6	0,2016	0,2918	0,2361	5,1487	0,2574	5,1526
7	0,2016	0,2574	0,2229	5,1533	0,2361	5,1487
8	0,2229	0,2574	0,2361	5,1487	0,2442	5,1485
9	0,2361	0,2574	0,2442	5,1485	0,2492	5,1494
10	0,2361	0,2492	0,2411	5,1483	0,2442	5,1485
11	0,2361	0,2442	0,2392	5,1484	0,2411	5,1483
12	0,2392	0,2442	0,2411	5,1483	0,2423	5,1484
13	0,2392	0,2423	0,2404	5,1483	0,2411	5,1483
14	0,2404	0,2423	0,2411	5,1483	0,2415	5,1483

- **Метод Фиббоначи**
  Начальные условия: $left = 0$, $right = 0$, $\epsilon = 0.001$, $\delta = 0.00001$
  Полученный результат: $0,2408$
  Количество операций: 24

№	left-border	right-border	left	left-fun	right	right-fun
1	0,0000	1,0000	0,3820	5,4744	0,6180	7,8386
2	0,0000	0,6180	0,2361	5,1487	0,3820	5,4744
3	0,0000	0,3820	0,1459	5,2837	0,2361	5,1487
4	0,1459	0,3820	0,2361	5,1487	0,2918	5,1893
5	0,1459	0,2918	0,2016	5,1717	0,2361	5,1487
6	0,2016	0,2918	0,2361	5,1487	0,2574	5,1526
7	0,2016	0,2574	0,2229	5,1533	0,2361	5,1487
8	0,2229	0,2574	0,2361	5,1487	0,2442	5,1485
9	0,2361	0,2574	0,2442	5,1485	0,2492	5,1494
10	0,2361	0,2492	0,2411	5,1483	0,2442	5,1485
11	0,2361	0,2442	0,2392	5,1484	0,2411	5,1483
12	0,2392	0,2442	0,2411	5,1483	0,2423	5,1484
13	0,2392	0,2423	0,2404	5,1483	0,2411	5,1483
14	0,2404	0,2423	0,2411	5,1483	0,2415	5,1483
15	0,2404	0,2415	0,2408	5,1483	0,2411	5,1483
16	0,2404	0,2411	0,2406	5,1483	0,2408	5,1483
17	0,2406	0,2411	0,2408	5,1483	0,2409	5,1483
18	0,2406	0,2409	0,2407	5,1483	0,2408	5,1483
19	0,2407	0,2409	0,2408	5,1483	0,2409	5,1483
20	0,2407	0,2409	0,2408	5,1483	0,2408	5,1483
21	0,2407	0,2408	0,2408	5,1483	0,2408	5,1483
22	0,2408	0,2408	0,2408	5,1483	0,2408	5,1483
23	0,2408	0,2408	0,2408	5,1483	0,2408	5,1483
24	0,2408	0,2408	0,2408	5,1483	0,2408	5,1483

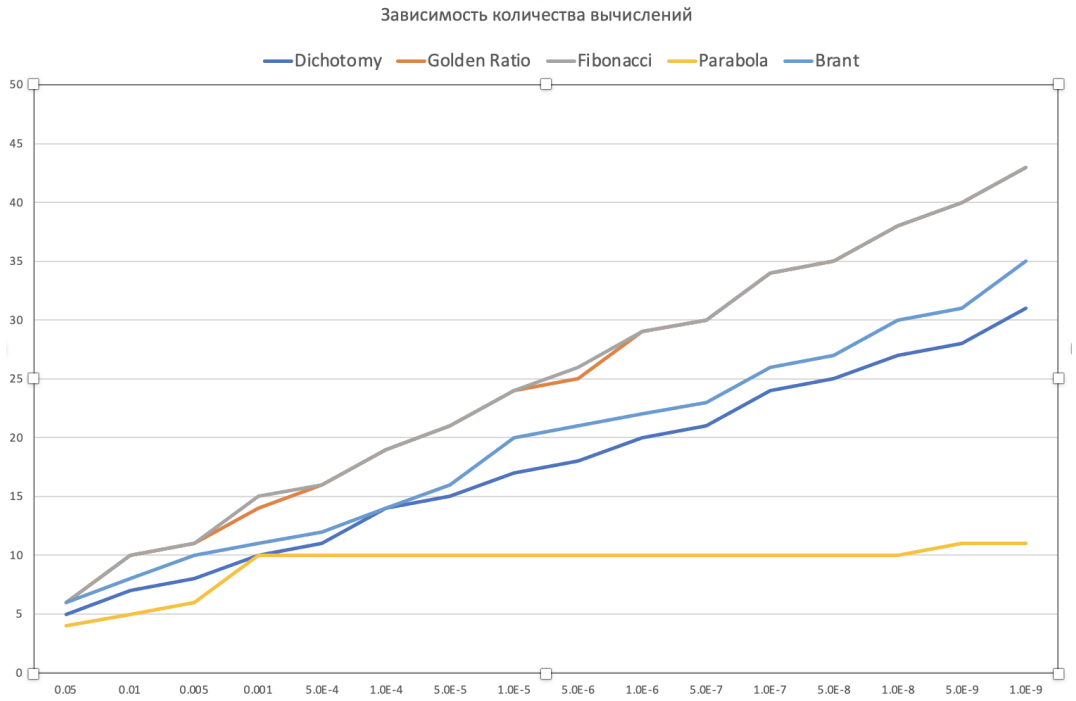
- **Метод Парабол**
  Начальные условия: $left = 0$, $right = 0$, $\epsilon = 0.001$, $\delta = 0.00001$
  Полученный результат: $0,2408$
  Количество операций: 10

№	left-border	right-border	x	x-fun	middle	middle-fun
1	0,0000	1,0000	0,2500	5,1497	0,2500	5,1497
2	0,0000	1,0000	0,2500	5,1497	0,1952	5,1800
3	0,1952	1,0000	0,2500	5,1497	0,2330	5,1493
4	0,1952	0,2500	0,2330	5,1493	0,2408	5,1483
5	0,2330	0,2500	0,2408	5,1483	0,2408	5,1483
6	0,2408	0,2500	0,2408	5,1483	0,2408	5,1483
7	0,2408	0,2500	0,2408	5,1483	0,2408	5,1483
8	0,2408	0,2500	0,2408	5,1483	0,2408	5,1483
9	0,2408	0,2500	0,2408	5,1483	0,2408	5,1483
10	0,2408	0,2408	0,2408	5,1483	0,2408	5,1483

- **Комбинированный метод Брента**
  Начальные условия: $left = 0$, $right = 0$, $\epsilon = 0.001$, $\delta = 0.00001$
  Полученный результат: $0,2409$
  Количество операций: 11

№	left-border	right-border	x	x-fun	w	w-fun	v	v-fun	u	u-fun
1	0,0000	1,0000	0,5000	6,3211	0,5000	6,3211	0,5000	6,3211	0,0000	6,0000
2	0,0000	0,5000	0,1910	5,1861	0,5000	6,3211	0,5000	6,3211	0,1910	5,1861
3	0,0000	0,3090	0,1910	5,1861	0,3090	5,2221	0,5000	6,3211	0,3090	5,2221
4	0,1910	0,3090	0,2318	5,1496	0,1910	5,1861	0,3090	5,2221	0,2318	5,1496
5	0,1910	0,2500	0,2318	5,1496	0,2500	5,1497	0,1910	5,1861	0,2500	5,1497
6	0,2318	0,2500	0,2352	5,1488	0,2318	5,1496	0,2500	5,1497	0,2352	5,1488
7	0,2352	0,2500	0,2409	5,1483	0,2352	5,1488	0,2318	5,1496	0,2409	5,1483
8	0,2352	0,2444	0,2409	5,1483	0,2444	5,1485	0,2352	5,1488	0,2444	5,1485
9	0,2398	0,2444	0,2409	5,1483	0,2398	5,1484	0,2444	5,1485	0,2398	5,1484
10	0,2398	0,2412	0,2409	5,1483	0,2412	5,1483	0,2398	5,1484	0,2412	5,1483
11	0,2405	0,2412	0,2409	5,1483	0,2405	5,1483	0,2412	5,1483	0,2405	5,1483

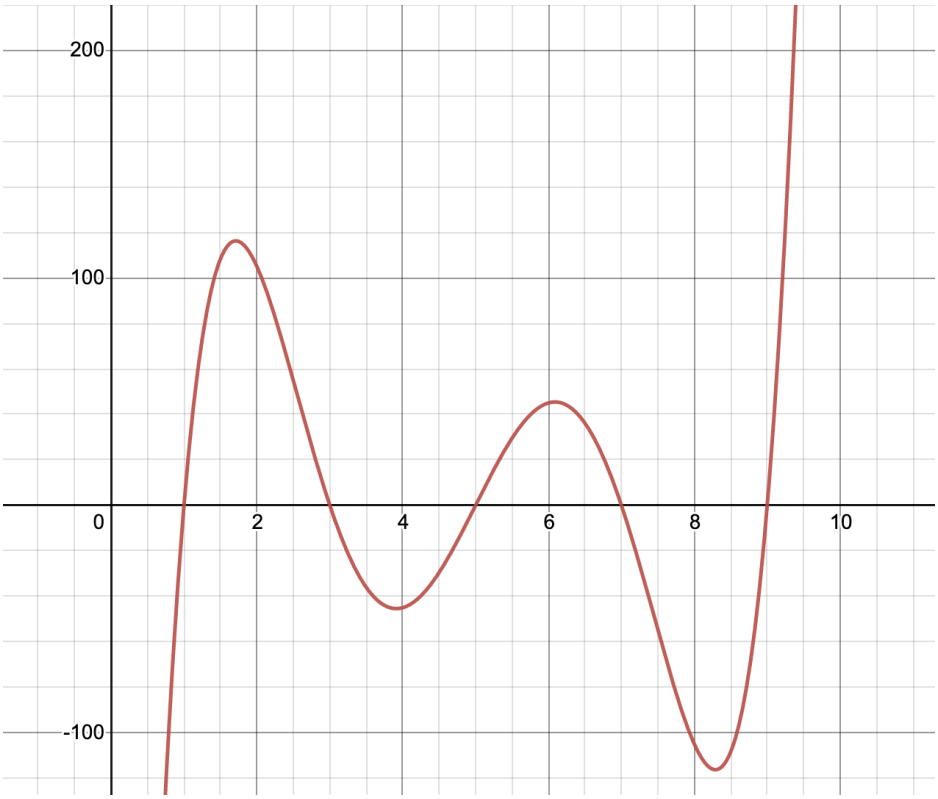
3. Зависимость количества вычислений от точности



4. Тестирование на многомодальных функциях

Тестирование методов на функции: $f(x) = (x-1)\cdot(x-3)\cdot(x-5)\cdot(x-7)\cdot(x-9) \to min : x \in [1; 10]$

Dichotomy Method	3,91213
Golden Ratio Method	3,91220
Fibonacci Method	3,91218
Parabola Method	3,91218
Brant Combine Method	8,28907
Аналитическое решение	8.2889



Вывод: исходя из полученных результатов делаем вывод, что данные методы не могут гарантировать нохождения глобального экстремума на отрезке, в случае если заданная им функция не является унимодальной.

5. Сравнение методов и вывод

- Ожидается, что методу Брента требуется наименьшее количество операций для нахождения минимума, но вследствии того что функция данного варианта ведет себя при рассмотрении на заданном участе похоже на квадратическую метод парабол оказался более эффективным.
- Метод брента окказался вторым по эффективности, требующий значительно меньше операций чем оставшиеся методы.
- Метод Золотого сечения и метод Фиббоначи покказали себя более менее одинакого, но лучше чем метод дихотомии на разных унимодальных функкциях.
- Все эти методы можно использоваться только для поиска мимимума для унимодальных функций.

P.S. На графики зависсимости количества вычислений метод дихотомии имеет малое ккол-ичество вычислений из-за маленькой δ, так как при больших, если $\epsilon < 1.0E-5$, количество операций сильно возрастает.

6. Программный код

Реализация всех методов и всех вспомогательных классов вывовящих резальтаты представленна в репозитории (sdmitrioul). Здесь представлен только интерфейс метода, абстрактный класс метода и для примера метод золотого сечения.

Интерфейс:

```
public interface MinimumSearcher {
    //PRE:  leftBorder < rightBorder
    Answer findMin(double leftBorder, double rightBorder) throws TimeOutException;
    int getOperationCounter();
    //PRE:  findMin must throw exception
    Answer answerWithException();
}
```

Абстрактный класс:

```
public abstract class AbstractMethod implements MinimumSearcher {
    private final int MAX_OPERATION_NUMBER = 1000000;
    private boolean exception = false;
    private final String methodName;
    protected final double EPS;
    protected final double DELTA;
    protected final String nameOfParameters;
    protected final Function<Double, Double> function;
    protected int operationCounter = 1;
    protected LinkedHashMap<Integer, Data> data = new LinkedHashMap<>();


    public AbstractMethod(String methodName, double EPS, double DELTA,
                          String nameOfParameters, Function<Double, Double> function) {
        this.methodName = methodName;
        this.EPS = EPS;
        this.DELTA = DELTA;
        this.nameOfParameters = nameOfParameters;
        this.function = function;
    }

    @Override
    public abstract Answer findMin(double leftBorder, double rightBorder)
     throws TimeOutException;

    @Override
    public int getOperationCounter() {
        return operationCounter;
    }

    protected Answer wrapper(final double min) {
        return new Answer(min, data,
                nameOfParameters, this.methodName);
```

```
    }

    protected void addData(final int operation, Data data) {
        this.data.put(operation, data);
    }

    protected List<Pair<Double, Double>> pointsWrapper(final double ... points) {
        ArrayList<Pair<Double, Double>> list = new ArrayList<>();
        for (double point : points) {
            list.add(new Pair<>(point, function.apply(point)));
        }
        return list;
    }

    protected void checkCondition() throws TimeOutException {
        if (operationCounter > MAX_OPERATION_NUMBER) {
            exception = true;
            throw new TimeOutException(methodName + " can't find operation in needed
             time");
        }
    }

    public Answer answerWithException() {
        if (exception) {
            return wrapper(Double.NaN);
        }
        return null;
    }

    protected Data wrapData(final double leftBorder, final double rightBorder,
     final double ... points) {
        return new Data(leftBorder, rightBorder, rightBorder - leftBorder,
         pointsWrapper(points));
    }

    protected void clear() {
        operationCounter = 1;
        exception = false;
        data = new LinkedHashMap<>();
    }
}
```

Метод золотого сечения:

```
public class GoldenRatioMethod extends AbstractMethod {
    private final BinaryOperator<Double> leftFunction = (left, right)
            -> left + (3 - Math.sqrt(5)) * (right - left) / 2;
    private final BinaryOperator<Double> rightFunction = (left, right)
            -> left + (Math.sqrt(5) - 1) * (right - left) / 2;
```

```
    public GoldenRatioMethod(final double EPS, final double DELTA,
                             final Function<Double, Double> function) {
        super("Golden_Ratio_Method",EPS, DELTA,
                "(left-border) (right-border) " +
                        "(left) (left-fun) (right) (right-fun)", function);
    }

    @Override
    public Answer findMin(double leftBorder, double rightBorder)
            throws TimeOutException {
        clear();
        double left = leftFunction.apply(leftBorder, rightBorder);
        double right = rightFunction.apply(leftBorder, rightBorder);

        addData(operationCounter, wrapData(leftBorder, rightBorder, left, right));

        while ((rightBorder - leftBorder) / 2 > EPS) {
            if (function.apply(left) <= function.apply(right)) {
                rightBorder = right;
                right = left;
                left = leftFunction.apply(leftBorder, rightBorder);
            } else {
                leftBorder = left;
                left = right;
                right = rightFunction.apply(leftBorder, rightBorder);
            }

            addData(++operationCounter,
                    wrapData(leftBorder, rightBorder, left, right));
            checkCondition();
        }

        double min = (leftBorder + rightBorder) / 2;

        return wrapper(min);
    }
}
```
