Университет ИТМО

ФПИиКТ

# Лабораторная работа №2
# по Вычислительной математике

Выполнил: Балтабаев Дамир
Группа: P3210
Вариант: 4

Преподаватель: Малышева Татьяна Алексеевна

Санкт-Петербург
2022

## Цель лабораторной работы:

Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения, выполнить программную реализацию методов.

## Порядок выполнение работы:

1. Отделить корни заданного нелинейного уравнения графически
2. Определить интервалы изоляции корней.
3. **<u>Вычислительная реализация задачи (в отчет):</u>**

Уточнить корни нелинейного уравнения с точностью $\varepsilon=10^{-2}$. Вычисления оформить в виде таблиц , удержать 3 знака после запятой. Представить в отчете заполненные таблицы.

4. **Программная реализация задачи:**

**<u>Для нелинейных уравнений:</u>**

4.1 Все численные методы должны быть реализованы в виде отдельных подпрограмм или классов.

4.2 Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает программа.

4.3 Предусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность вычисления) из файла или с клавиатуры по выбору конечного пользователя.

4.4 Выполнить верификацию исходных данных. Для метода половинного деления (метода хорд) анализировать наличие корня на введенном интервале. Для метода Ньютона (метода секущих) – выбор начального приближения (a или b). Для метода простой итерации – достаточное условие сходимости метода. Программа должна реагировать на некорректные введенные данные.

4.5 Предусмотреть вывод результатов (найденный корень уравнения, значение функции в корне, число итераций) в файл или на экран по выбору конечного пользователя.

4.6 Организовать вывод графика функции, график должен полностью отображать весь исследуемый интервал (с запасом).

**Для систем нелинейных уравнений:**

4.7 Рассмотреть систему двух уравнений.

4.8 Организовать вывод графика функций.

4.9 Для метода простой итерации проверить достаточное условие сходимости.

4.10 Вывод вектора неизвестных: $x_1,\ x_2$

4.11 Вывод количества итераций, за которое было найдено решение.

4.12 Вывод вектора погрешностей: $|x_i^{(k)} - x_i^{(k-1)}|$

## Рабочие формулы используемых методов:

Метод простой итерации:

**Рабочая формула метода: $x_{i+1} = \varphi(x_i)$**

**Достаточное условие сходимости метода:**

$|\varphi'(x)| \leq q < 1$, где $q$ – некоторая константа (коэффициент Липшица или коэффициент сжатия)

**Критерий окончания итерационного процесса:**

$|x_n - x_{n-1}| \leq \varepsilon$ (при $0 < q \leq 0{,}5$)

$|x_n - x_{n-1}| < \frac{1-q}{q}\varepsilon$ (при $0{,}5 < q < 1$)

$$\varphi(x) = x + \lambda f(x), \varphi'(x) = 1 + \lambda f'(x)$$

$$\lambda = -\frac{1}{\max\limits_{[a,b]} f'(x)}$$

Метод половинного деления:

**Рабочая формула метода:** $x_i = \frac{a_i + b_i}{2}$

Критерий окончания итерационного процесса: $|b_n - a_n| \le \varepsilon$ или $|f(x_n)| \le \varepsilon$.

Метод секущих:

**Рабочая формула метода:**

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i) \qquad i = 1,2 \ldots$$

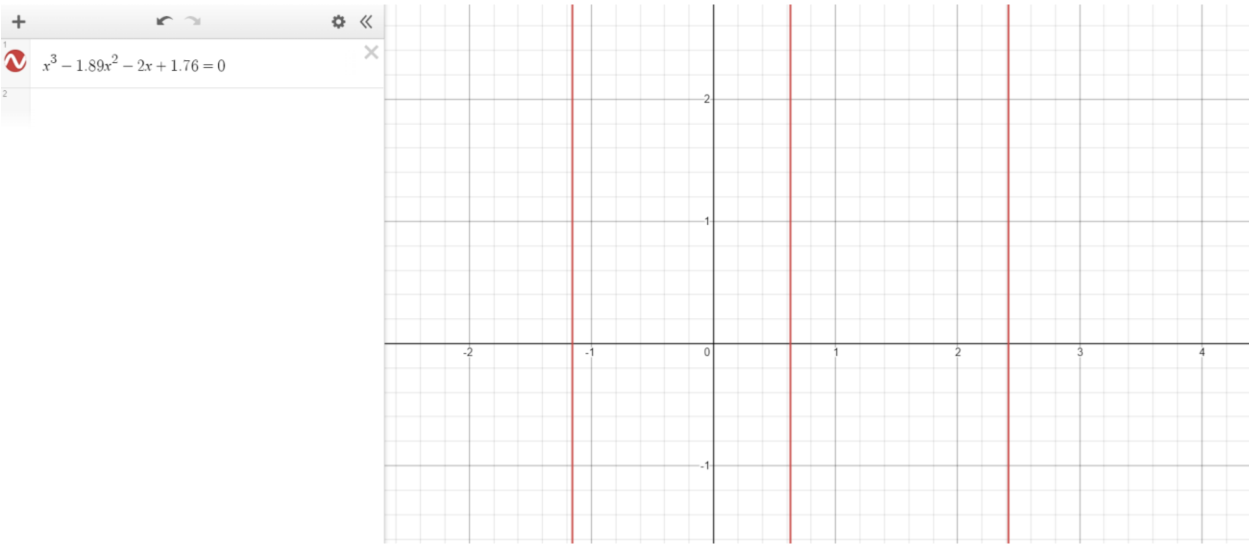
Критерий окончания итерационного процесса:

$$|x_n - x_{n-1}| \le \varepsilon \text{ или } |f(x_n)| \le \varepsilon.$$

## Отделение корней заданного нелинейного уравнения графически и определение интервалов изоляции корней:

$x^3 - 1{,}89x^2 - 2x + 1{,}76$ - заданное нелинейное уравнение

Построим график функции и найдем точки пересечения с осью абсцисс.




Глядя на график видно, что крайний левый корень находится на отрезке [-2; -1];

Центральный: [0; 1]

Крайний правый: [2; 3]

# Заполненные таблицы

$x^3 - 1{,}89x^2 - 2x + 1{,}76$ - заданное нелинейное уравнение

Точность $\varepsilon=10^{-2}$

1. Крайний правый корень: Метод простой итерации

Крайний правый корень расположен на отрезке [2; 3]

| № итерации | $x_k$ | $f(x_k)$ | $x_{k+1}$ | $\varphi(x_k)$ | $\lvert x_k - x_{k+1} \rvert$ |
|---|---|---|---|---|---|
| 1 | 2.000 | -1.800 | 2.230 | 2.230 | 0.230 |
| 2 | 2.230 | -1.009 | 2.359 | 2.359 | 0.129 |
| 3 | 2.359 | -0.348 | 2.403 | 2.403 | 0.044 |
| 4 | 2.403 | -0.084 | 2.414 | 2.414 | 0.011 |
| 5 | 2.414 | -0.014 | 2.416 | 2.416 | 0.002 |

2. Крайний левый корень: Метод половинного деления

Крайний левый корень расположен на отрезке [-2; -1]

| № шага | a | b | x | f(a) | f(b) | f(x) | \|a-b\| |
|---|---|---|---|---|---|---|---|
| 1 | -2.000 | -1.000 | -1.500 | -9.800 | 0.870 | -2.867 | 1.000 |
| 2 | -1.500 | -1.000 | -1.250 | -2.867 | 0.870 | -0.646 | 0.500 |
| 3 | -1.250 | -1.000 | -1.125 | -0.646 | 0.870 | 0.194 | 0.25 |
| 4 | -1.250 | -1.125 | -1.188 | -0.646 | 0.194 | -0.208 | 0.125 |
| 5 | -1.188 | -1.125 | -1.157 | -0.208 | 0.194 | -0.005 | 0.063 |
| 6 | -1.157 | -1.125 | -1.141 | -0.005 | 0.194 | 0.096 | 0.032 |
| 7 | -1.157 | -1.141 | -1.149 | -0.005 | 0.096 | 0.046 | 0.016 |
| 8 | -1.157 | -1.149 | -1.153 | -0.005 | 0.046 | 0.021 | 0.008 |

3. Центральный корень: Метод секущих

Центральный корень расположен на отрезке [0; 1]

| № итерации | $x_{k-1}$ | $f(x_{k-1})$ | $x_k$ | $f(x_k)$ | $x_{k+1}$ | $f(x_{k+1})$ | $\lvert x_k - x_{k+1} \rvert$ |
|---|---|---|---|---|---|---|---|
| 1 | 0.000 | 1.760 | 0.250 | 1.157 | 0.730 | -0.318 | 0.480 |
| 2 | 0.250 | 1.157 | 0.730 | -0.318 | 0.627 | 0.009 | 0.103 |
| 3 | 0.730 | -0.318 | 0.627 | 0.009 | 0.630 | 0.000 | 0.003 |

# Листинг программы

- Метод простых итераций для решения нелинейных уравнений

```
    public void mainIterationMethod() throws IOException {

        lambdaCalculation();

        // Проверка на сходимость

        if (q >= 0 && q < 1) {
            messenger.convergenceConditionIsMetMessage();
        } else {
            messenger.convergenceConditionIsNotMetMessage();
            System.exit(0);
        }


        writer.write("Метод простых итераций:");
        int iterationCounter = 0;
        Double xi = null;
        if (computingFunctional.equationArgument(firstBorder) *
computingFunctional.equationSecondDerivative(firstBorder) > 0) {
            xi = firstBorder;
        } else if (computingFunctional.equationArgument(secondBorder) *
computingFunctional.equationSecondDerivative(secondBorder) > 0) {
            xi = secondBorder;
        } else xi = findMax(false);
        while (true) {
            writer.write("Итерация № " + iterationCounter);
            writer.write("Xi = " + xi);
            Double xi_1 = computingFunctional.equationFiX(lambda, xi);
            writer.write("Xi+1 = " + xi_1);
            Double fi_xi_1 = computingFunctional.equationFiX(lambda, xi_1);
            writer.write("\uD835\uDF4B(Xi+1) = " + fi_xi_1);
            Double f_xi_1 = computingFunctional.equationArgument(xi_1);
            writer.write("f(Xi+1) = " + f_xi_1);
            Double fault = Math.abs(xi_1 - xi);
            writer.write("|Xi+1-Xi| = " + fault);

            if (q >= 0 && q <= 0.5) {
```

```
                if (fault <= getEpsilon() &&
(Math.abs(computingFunctional.equationArgument(xi_1)) <= getEpsilon())) {
                    writer.write("\nОтвет: " + xi_1);
                    writer.write("Количество итераций: " + iterationCounter);
                    writer.write("Значение функции в точке x = " +
computingFunctional.equationArgument(xi_1));
                    writer.write("Fi'(a) = " +
computingFunctional.equationTransformedDerivative(lambda, firstBorder));
                    writer.write("Fi'(b) = " +
computingFunctional.equationTransformedDerivative(lambda, secondBorder));

                    return;
                }
            }

            if (q > 0.5 && q < 1) {
                if (fault <= ((1 - q) / q) * getEpsilon() &&
(Math.abs(computingFunctional.equationArgument(xi_1)) <= getEpsilon())) {
                    writer.write("\nОтвет: " + xi_1);
                    writer.write("Количество итераций: " + iterationCounter);
                    writer.write("Значение функции в точке x = " +
computingFunctional.equationArgument(xi_1));
                    writer.write("Fi'(a) = " +
computingFunctional.equationTransformedDerivative(lambda, firstBorder));
                    writer.write("Fi'(b) = " +
computingFunctional.equationTransformedDerivative(lambda, secondBorder));

                    return;
                }
            }

            iterationCounter++;
            xi = xi_1;
        }
    }
```

- Метод хорд для решения нелинейных уравнений

```
 public void mainChordMethod() throws IOException {
        Double a = null;
        Double b = null;
        Double x0 = null;
        int iteration_counter = 0;

        if (computingFunctional.equationArgument(firstBorder) *
computingFunctional.equationSecondDerivative(firstBorder) > 0) {
            x0 = firstBorder;
            a = firstBorder;
            b = secondBorder;
        } else if (computingFunctional.equationArgument(secondBorder) *
computingFunctional.equationSecondDerivative(secondBorder) > 0) {
            x0 = secondBorder;
            a = secondBorder;
            b = firstBorder;
        } else {
            messenger.convergenceConditionIsNotMetMessage();
            System.exit(0);
        }

        if
(computingFunctional.equationArgument(firstBorder)*computingFunctional.equation
Argument(secondBorder)<0){
            messenger.convergenceConditionIsMetMessage();
        }else {
            messenger.convergenceConditionIsNotMetMessage();
            System.exit(0);
```

```
        }

        writer.write("Метод хорд:");
        while (true) {
            writer.write("Итерация №" + iteration_counter);
            writer.write("a = " + a);
            writer.write("b = " + b);
            double x = (a * computingFunctional.equationArgument(b) - b *
computingFunctional.equationArgument(a)) /
                    (computingFunctional.equationArgument(b) -
computingFunctional.equationArgument(a));

            writer.write("x = " + x);
            writer.write("F(a) = " + computingFunctional.equationArgument(a));
            writer.write("F(b) = " + computingFunctional.equationArgument(b));
            writer.write("F(x) = " + computingFunctional.equationArgument(x));

            Double fault = Math.abs(x - x0);
            writer.write("|Xn-1-Xn| = " + fault);
            if (fault <= getEpsilon() &&
Math.abs(computingFunctional.equationArgument(x)) <= getEpsilon()) {
                writer.write("\nОтвет: " + x);
                writer.write("Количество итераций = " + iteration_counter);
                writer.write("Значение функции в точке x =
"+computingFunctional.equationArgument(x));
                return;
            }

            if (computingFunctional.equationArgument(a) *
computingFunctional.equationArgument(x) > 0) {
                a = x;
                x0 = a;
            }

            if (computingFunctional.equationArgument(b) *
computingFunctional.equationArgument(x) > 0) {
                b = x;
                x0 = b;
            }

            iteration_counter++;
        }


    }
```

- Метод простых итераций для решения систем нелинейных уравнений

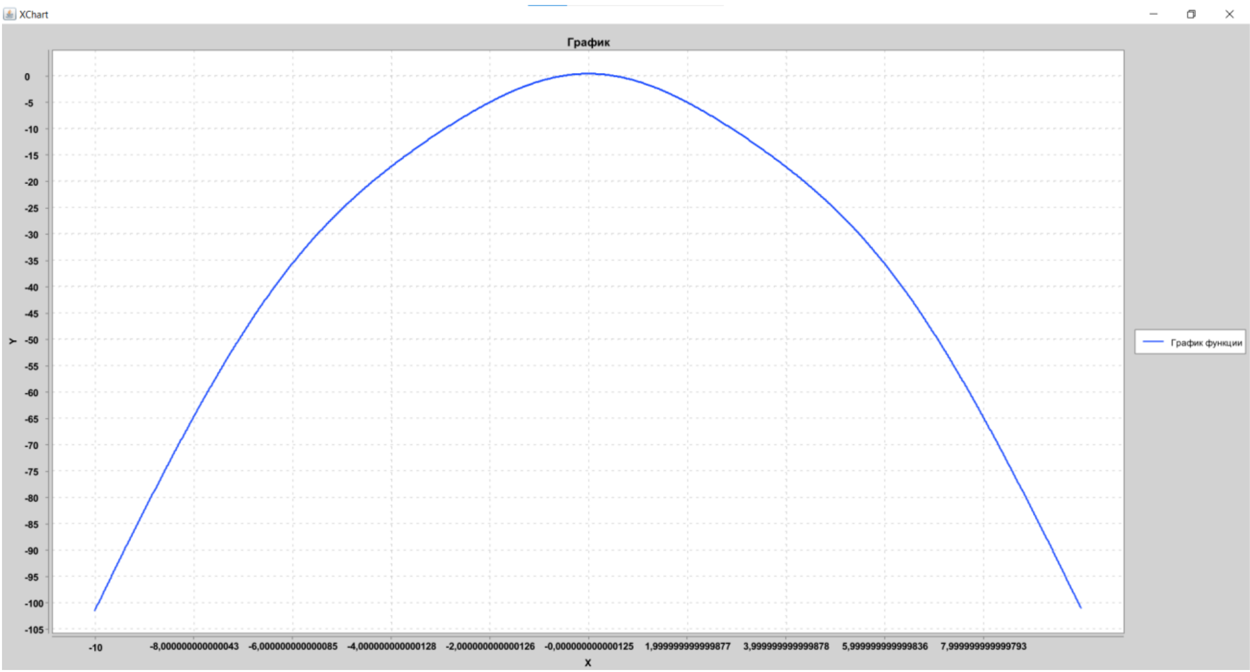
```
public void mainIterationMethod() throws IOException {

        checkOnConvergence();
        Double x0_1 = secondBorderOfFirstValue;
        Double x0_2 = secondBorderOfSecondValue;


        Double x1_new;
        Double x2_new;


        int iteration_counter = 0;
        writer.write("Метод простых итераций:\n");

        while (true) {
```

```
            x1_new = computingFunctional.firstSystemEquationArgument(x0_1,
x0_2);
            x2_new = computingFunctional.secondSystemEquationArgument(x0_1,
x0_2);

            writer.write("Итерация № " + iteration_counter);
            writer.write("x1 = " + x1_new);
            writer.write("x2 =" + x2_new);

            Double fault1 = Math.abs(x1_new - x0_1);
            Double fault2 = Math.abs(x2_new - x0_2);

            writer.write("xi^k-xi^(k-1) = " + fault1);
            writer.write("xi^k-xi^(k-1) = " + fault2);

            if ((fault1 <= getEpsilon() && fault2 <= getEpsilon()) ||

Math.abs(computingFunctional.firstSystemEquationArgument(x1_new, x2_new)) <=
getEpsilon() ||

Math.abs(computingFunctional.secondSystemEquationArgument(x1_new, x2_new)) <=
getEpsilon()) {
                writer.write("Количество итераций: " + iteration_counter);
                return;
            }

            x0_1 = x1_new;
            x0_2 = x2_new;
            iteration_counter++;

        }

    }
```

# Результаты выполнения программы

```
Введите цифру 1 для решения нелинейного уравнения
Введите цифру 2 для решение системы нелинейных уравнений
1
Выберите уравнение:
Введите цифру 1 для решения уравнения x^3-1.89x^2-2x+1.76
Введите цифру 2 для решения уравнения x^3-0.12x^2-1.475x+0.192
Введите цифру 3 для решения уравнения -0.5-x^2+cos(x)
3
Выберите метод для решения нелинейного уравнения:
Введите цифру 1 для решения методом хорд
Введите цифру 2 для решения методом простых итераций
1
Выберите метод ввода значений:
Введите цифру 1 в консоль для ввода с клавиатуры или цифру 2 для ввода с файла
1
Вы выбрали возможность ввода данных с КЛАВИАТУРЫ
Выберите метод вывода значений:
Введите цифру 1 для вывода ответа в консоль или цифру 2 для вывода в файл
1
Вы выбрали возможность вывода ответа в КОНСОЛЬ
Введите границы интервала в формате [a;b]:
[0;1]
Введите точность:
0.01
```

```
Метод хорд:
Итерация №0
a = 1.0
b = 0.0
x = 0.34253667866302256
F(a) = -0.9596976941318602
F(b) = 0.5
F(x) = 0.32457430742675786
Итерация №1
a = 1.0
b = 0.34253667866302256
x = 0.5086975090186359
F(a) = -0.9596976941318602
F(b) = 0.32457430742675786
F(x) = 0.1146064579476751
Итерация №2
a = 1.0
b = 0.5086975090186359
x = 0.5611095174459058
F(a) = -0.9596976941318602
F(b) = 0.1146064579476751
F(x) = 0.031821338714604464
Итерация №3
a = 1.0
b = 0.5611095174459058
x = 0.5751950591665483
F(a) = -0.9596976941318602
F(b) = 0.031821338714604464
F(x) = 0.008236850405879736
Итерация №4
a = 1.0
b = 0.5751950591665483
x = 0.5788100295889489
F(a) = -0.9596976941318602
F(b) = 0.008236850405879736
F(x) = 0.002093139449993453

Ответ: 0.5788100295889489
Количество итераций = 4
```




## Вывод

В ходе данной лабораторной работы я изучил методы уточнения корней нелинейных уравнений. Поработал с методами для вычисления алгебраических и трансцендентных нелинейных уравнений, а также систем. Изучил условия сходимости того или иного метода, а также научился строить графики функций.