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. **Вычислительная реализация задачи (в отчет):**

Уточнить корни нелинейного уравнения с точностью ε=10-2. Вычисления оформить в виде таблиц , удержать 3 знака после запятой.Представить в отчете заполненные таблицы.

1. **Программная реализация задачи:**

**Для нелинейных уравнений:**

* 1. Все численные методы должны быть реализованы в виде отдельных подпрограмм или классов.
  2. Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает программа.
  3. Предусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность вычисления) из файла или с клавиатуры по выбору конечного пользователя.
  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 (а или b). Для метода простой итерации – достаточное условие сходимости метода. Программа должна реагировать на некорректные введенные данные.
  5. Предусмотреть вывод результатов (найденный корень уравнения, значение функции в корне, число итераций) в файл или на экран по выбору конечного пользователя.
  6. Организовать вывод графика функции, график должен полностью отображать весь исследуемый интервал (с запасом).

**Для систем нелинейных уравнений:**

* 1. Рассмотреть систему двух уравнений.
  2. Организовать вывод графика функций.
  3. Для метода простой итерации проверить достаточное условие сходимости.
  4. Вывод вектора неизвестных:
  5. Вывод количества итераций, за которое было найдено решение.
  6. Вывод вектора погрешностей:

**Рабочие формулы используемых методов:**

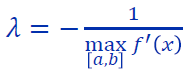
Метод простой итерации:



Изображение выглядит как текст

Автоматически созданное описание



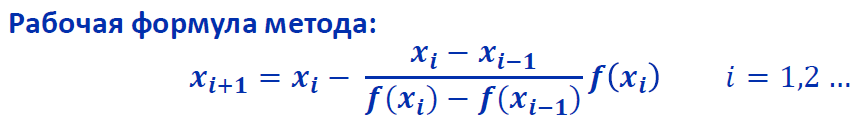


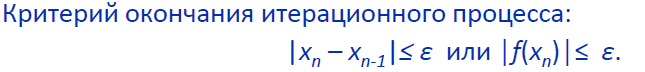
Метод половинного деления:





Метод cекущих:

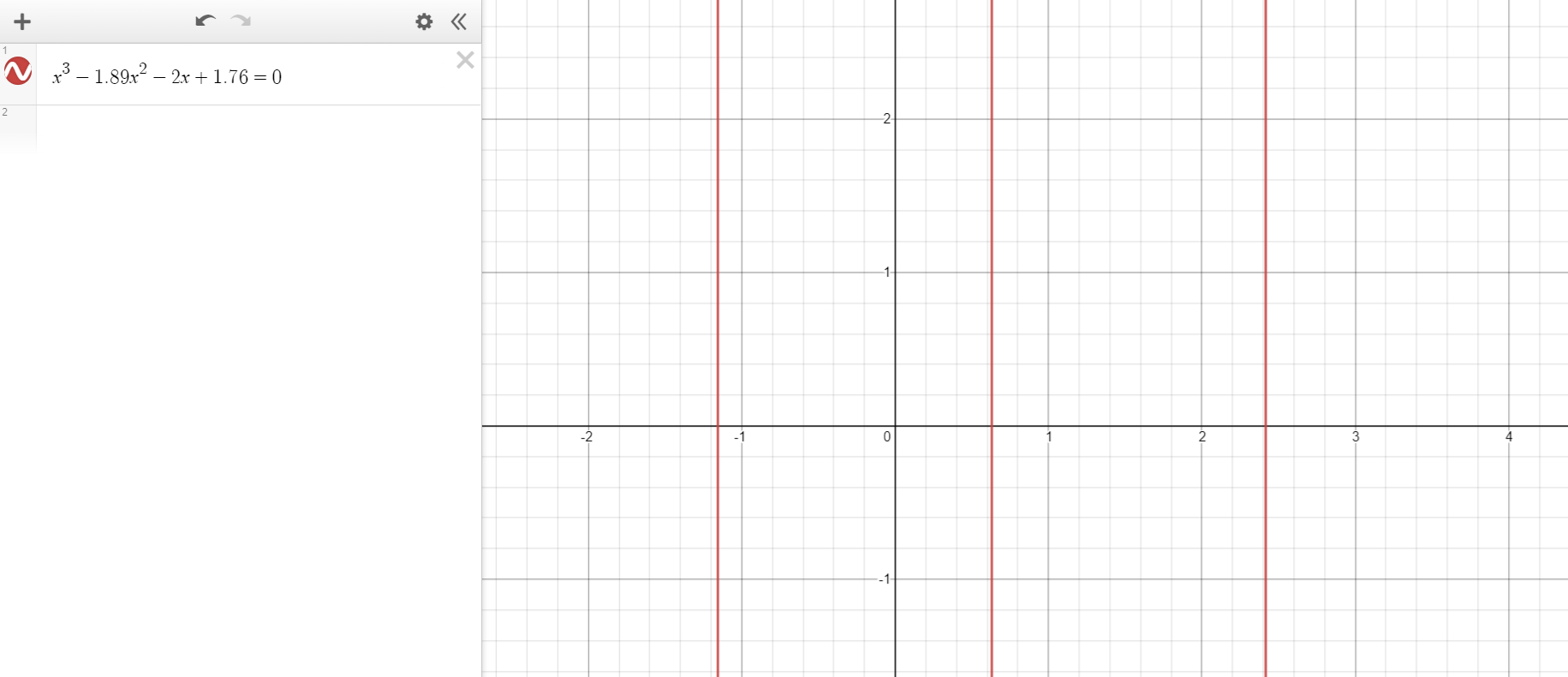




**Отделение корней заданного нелинейного уравнения графически и определение интервалов изоляции корней:**

- 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**

- заданное нелинейное уравнение

Точность ε=10-2

1. Крайний правый корень: Метод простой итерации

Крайний правый корень расположен на отрезке [2; 3]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации | *xk* | *f*(*xk* ) | *xk*+1 |  | │*xk* − *xk*+1│ |
| 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 |

1. 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 |

1. Центральный корень: Метод секущих

Центральный корень расположен на отрезке [0; 1]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № итерации | *xk-1* | *f*(*xk-1* ) | *xk* | *f*(*xk*) | *xk*+1 | *f*(*xk+1*) | │*xk* − *xk*+1│ |
| 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.equationArgument(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++;**

**}**

**}**

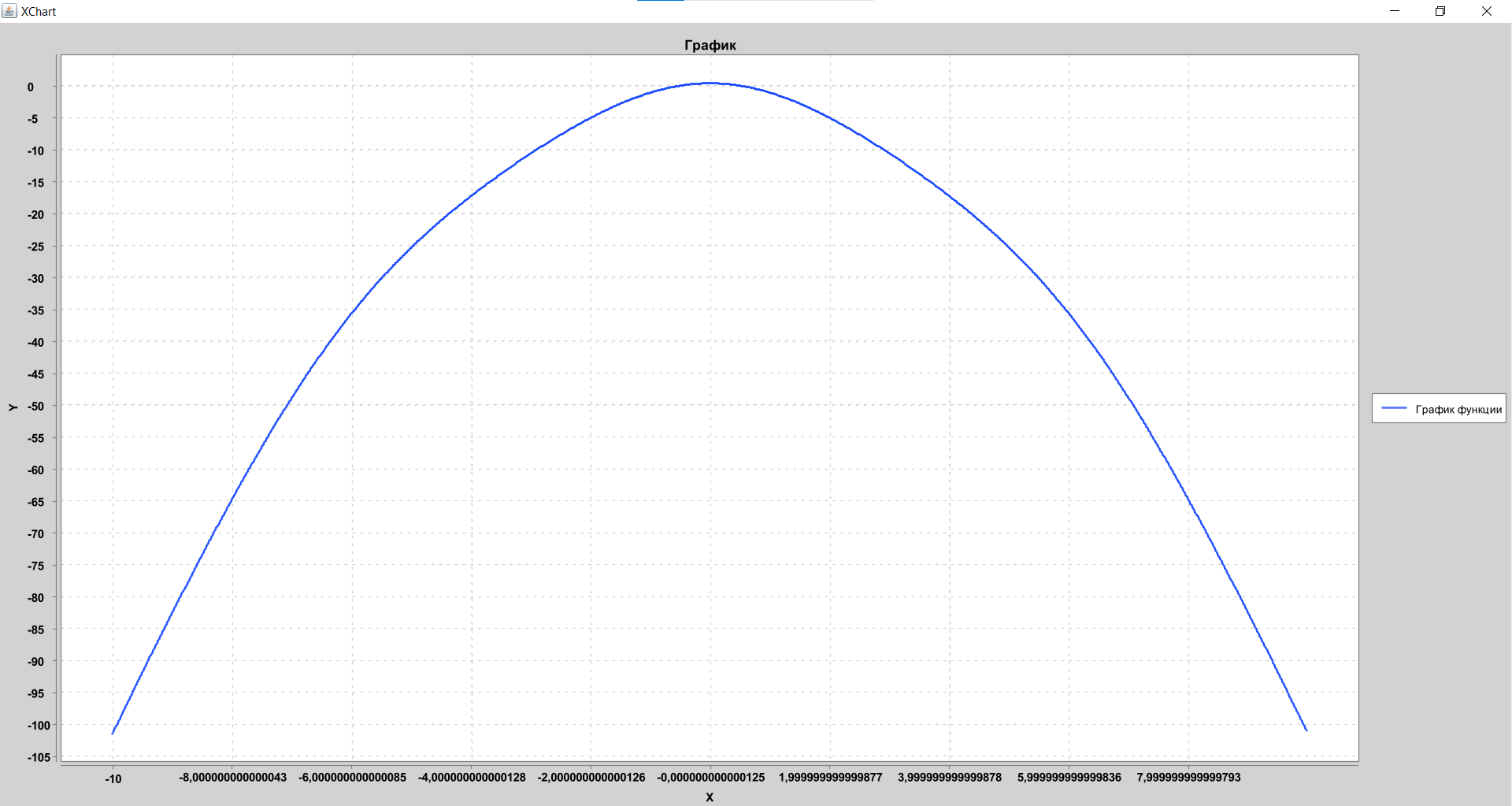
**Результаты выполнения программы**

**Изображение выглядит как текст

Автоматически созданное описание**

**Изображение выглядит как текст

Автоматически созданное описание**



**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.