Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №2**

по «Вычислительной математике»

**Численное решение нелинейных уравнений и систем**

Вариант 4

Выполнил:

Студент группы P3213

Ваховский П.А.

Преподаватель:

Малышева Т.А.

Санкт-Петербург

2022

**Цель работы**: написать программу, решающую нелинейные уравнения методом хорд, либо методом простых итераций по выбору пользователя. А также написать программу, решающую системы нелинейных уравнений из двух неизвестных методом простых итераций.

**Задание**

Уточнить *крайний правый корень* нелинейного уравнения методом половинного деления (или методом хорд, см. вариант задания) с точностью ε=10-2. Вычисления оформить в виде таблицы, удержать 3 знака после запятой (см. табл. 1).

Уточнение корня уравнения методом половинного деления (хорд)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | a | b | x | f(a) | f(b) | f(x) | |a-b| |
| 1 | 2 | 4 | 4 | -1.8 | 27.5 | 27.5 | 2 |
| 2 | 2 | 4 | 2.123 | -1.8 | 27.5 | -1.437 | 2 |
| 3 | 2.123 | 4 | 2.216 | -1.437 | 27.5 | -1.071 | 1.87 |
| 4 | 2.216 | 4 | 2.283 | -1.071 | 27.5 | -0.759 | 1.78 |
| 5 | 2.283 | 4 | 2.329 | -0.759 | 27.5 | -0.518 | 1.72 |
| 6 | 2.329 | 4 | 2.360 | -0.518 | 27.5 | -0.344 | 1.67 |
| 7 | 2.360 | 4 | 2.380 | -0.344 | 27.5 | X\_-0.225 | 1.64 |
| 8 | 2.380 | 4 | 2.393 | -0.225 | 27.5 | -0.115 | 1.62 |
| 9 | 2.393 | 4 | 2.402 | -0.115 | 27.5 | -0.093 | 1.61 |

Уточнить *центральный корень* нелинейного уравнения методом простой итерации с точностью ε=10-2. Вычисления оформить в виде таблицы, удержать 3 знака после запятой (см. табл. 4).  
 Таблица 4

Уточнение корня уравнения методом простой итерации

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации | *xk* | *f(xk )* | *xk+1* |  | *│xk − xk+1│* |
| 1 | 0.500 | 0.413 | 0.706 | 0.706 | 0.206 |
| 2 | 0.706 | -0.243 | 0.585 | 0.585 | 0.121 |
| 3 | 0.585 | 0.144 | 0.657 | 0.657 | 0.072 |
| 4 | 0.657 | -0.086 | 0.614 | 0.614 | 0.043 |
| 5 | 0.614 | 0.051 | 0.639 | 0.639 | 0.025 |
| 6 | 0.639 | -0.030 | 0.624 | 0.624 | 0.015 |
| 7 | 0.624 | -0.018 | 0.633 | 0.633 | 0.009 |
| 8 | 0.633 | -0.011 | 0.628 | 0.628 | 0.005 |

**Программная реализация задачи:**

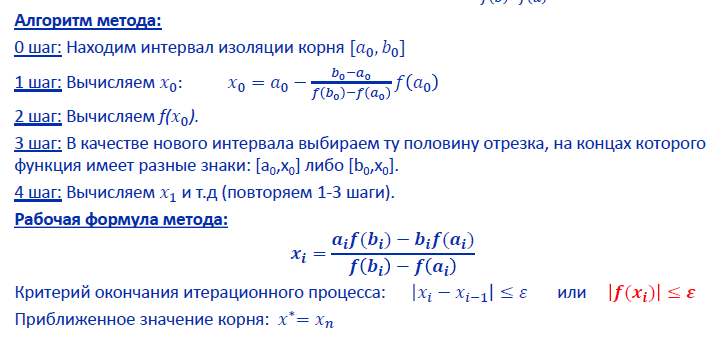
**Для нелинейных уравнений:**

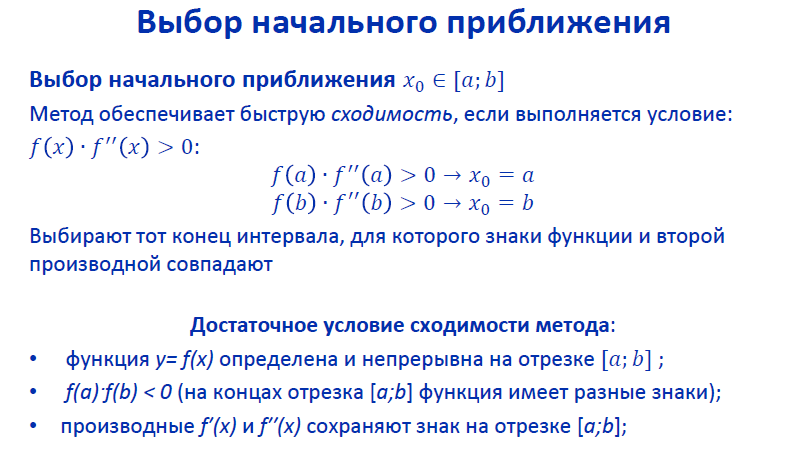
* 1. Все численные методы (см. табл. 6) должны быть реализованы в виде отдельных подпрограмм или классов.
  2. Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает программа.
  3. Предусмотреть ввод исходных 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:

**Описание метода хорд:**





**Листинг класса ChordMethod:**

public class ChordMethod {  
 Function f, df, ddf;  
 double a, b, eps;  
 public ChordMethod(Function f, Function df, Function ddf, double a, double b, double eps){  
 this.f = f;  
 this.df = df;  
 this.ddf = ddf;  
 this.a = a;  
 this.b = b;  
 this.eps = eps;  
 }  
 double calcX(){  
 return a - (b-a)/(f.calcFunction(b)-f.calcFunction(a)) \* f.calcFunction(a);  
 }  
  
 private String result;  
  
 public String getResult() {  
 return result;  
 }  
  
 public int checkInterval(){  
 if (f.calcFunction(a) \* f.calcFunction(b) >= 0) {  
 System.*out*.println("На выбранном интервале нет корня, либо корень не единственный.");  
 return -1;  
 }  
 double prevSignDf = Math.*signum*(df.calcFunction(a));  
 double prevSignDdf = Math.*signum*(ddf.calcFunction(a));  
 for(double i = a; i <= b; i+=0.05){  
 if(prevSignDf != Math.*signum*(df.calcFunction(i)) || prevSignDdf != Math.*signum*(ddf.calcFunction(i))){  
 System.*out*.println("Не выполняется условие константности знака первой или второй производной ф-и.");  
 // System.out.println(i);  
 return 0;  
 }  
 prevSignDf = Math.*signum*(df.calcFunction(i));  
 prevSignDdf = Math.*signum*(ddf.calcFunction(i));  
 }  
 return 1;  
 }  
  
 public double solution() {  
 double prevX = calcX();  
 if(f.calcFunction(a)\*ddf.calcFunction(a) > 0)  
 prevX = a;  
 if(f.calcFunction(b)\*ddf.calcFunction(b) > 0)  
 prevX = b;  
 int count = 0;  
 while (Math.*abs*(prevX) > eps) {  
 count++;  
 if (f.calcFunction(a) \* f.calcFunction(prevX) < 0)  
 b = prevX;  
 else  
 a = prevX;  
 if (Math.*abs*(calcX() - prevX) <= eps) {  
 prevX = calcX();  
 break;  
 }  
 prevX = calcX();  
 }  
 result = String.*format*("Найдено решение x\* = %.5f за %d итераций\n", prevX, count) +  
 String.*format*("Значение ф-и в корне f(x\*) = %.5f\n", f.calcFunction(prevX));  
 return prevX;  
 }  
}

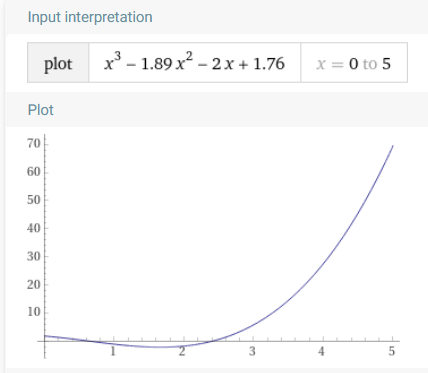
**Пример:**

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



**Описание метода простой итерации:**

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

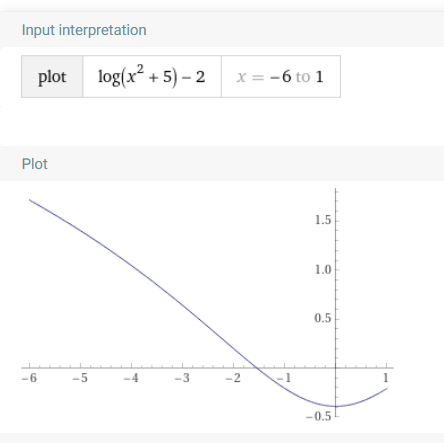
**Листинг класса SimpleIteration:**

public class SimpleIterationMethod {  
 Function f, df ;  
 double a, b, eps;  
 public SimpleIterationMethod(Function f, Function df, double a, double b, double eps){  
 this.f = f;  
 this.df = df;  
 this.a = a;  
 this.b = b;  
 this.eps = eps;  
 }  
  
 private double calcLambda(){  
 return -1d/Double.*max*(df.calcFunction(a), df.calcFunction(b));  
 }  
  
 private String result;  
  
 public double solution(){  
 double lambda = calcLambda();  
 Function dphi = x->1+lambda\*df.calcFunction(x);  
 double q = 0d;  
 for(double i = a; i <= b; i += 0.05){  
 q = Double.*max*(q, Math.*abs*(dphi.calcFunction(i)));  
 }  
 if(q >= 1){  
 System.*out*.printf("q = %.2f Достаточное условие сходимости не выполняется\n", q);  
 return Double.*NaN*;  
 }  
 Function phi = x->x+lambda\*f.calcFunction(x);  
 double prevX = (a+b)/2;  
 int count = 0;  
 while(count < 1000){  
 count++;  
 double curX = phi.calcFunction(prevX);  
 double delta = Math.*abs*(curX - prevX);  
 if (0 < q && q <= 0.5 && delta <= eps ||  
 0.5 < q && q < 1 && delta < (1 - q) / q \* eps) {  
 result = String.*format*("Найдено решение x\* = %.5f за %d итераций\n", curX, count) +  
 String.*format*("Значение ф-и в корне f(x\*) = %.5f\n", f.calcFunction(curX));  
 return curX;  
 }  
 prevX = curX;  
 }  
 return Double.*NaN*;  
 }  
  
 public String getResult() {  
 return result;  
 }  
}

**Пример:**

Изображение выглядит как текст

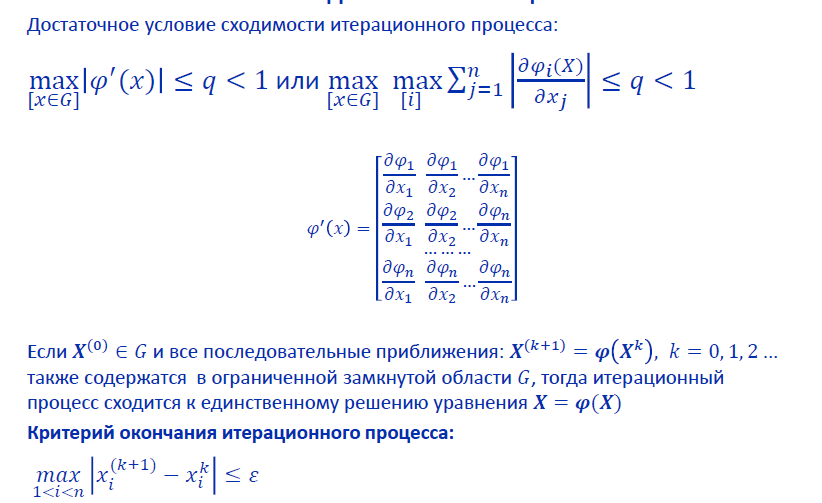
Автоматически созданное описание



**Описание метода простой итерации для систем нелинейных уравнений:**

Изображение выглядит как текст

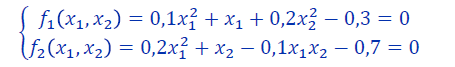
Автоматически созданное описание

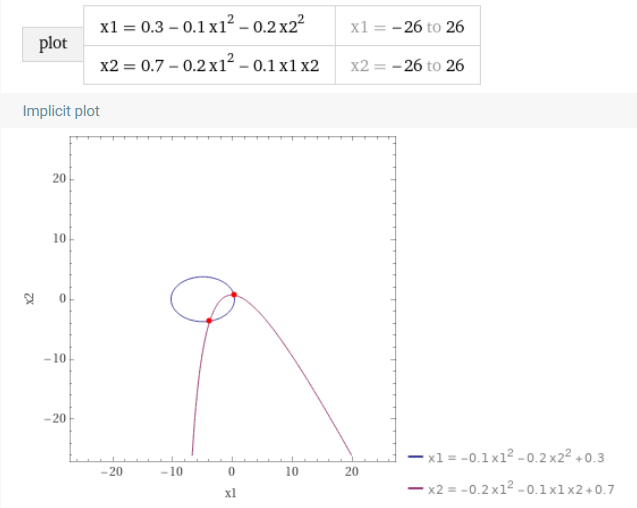


**Листинг класса SimpleIterationForSystem:**

public class SimpleIterationForSystem {  
 FunctionWithTwoParameters f1, f2, phi1, phi2;  
 double prevX1, prevX2;  
 double x1, x2;  
 double eps;  
 public SimpleIterationForSystem(FunctionWithTwoParameters f1, FunctionWithTwoParameters f2,  
 FunctionWithTwoParameters phi1, FunctionWithTwoParameters phi2,  
 double eps) {  
 this.f1 = f1;  
 this.f2 = f2;  
 this.phi1 = phi1;  
 this.phi2 = phi2;  
 this.eps = eps;  
 }  
  
 public void setFirstXs(double prevX1, double prevX2){  
 this.prevX1 = prevX1;  
 this.prevX2 = prevX2;  
 }  
  
 String result;  
  
 public String getResult() {  
 return result;  
 }  
  
 public void solution(){  
 int count = 0;  
 while(true){  
 count++;  
 x1 = phi1.calcFunction(prevX1, prevX2);  
 x2 = phi2.calcFunction(prevX1, prevX2);  
 double delta1 = Math.*abs*(prevX1-x1), delta2 = Math.*abs*(prevX2-x2);  
 if(Double.*min*(delta1, delta2) < eps){  
 result = String.*format*("Найдено решение {x1\* = %.5f, x2\* = %.5f} за %d итераций\n", x1, x2, count) +  
 String.*format*("Погрешности: {|x1\_k+1 - x1\_k| = %.5f, |x2\_k+1 - x2\_k| = %.5f}\n", delta1, delta2);  
 return;  
 }  
 prevX1 = x1;  
 prevX2 = x2;  
 }  
 }  
}

**Пример:**





Изображение выглядит как текст

Автоматически созданное описание**Вывод: написал две программы, для решения нелинейных уравнений и их систем соответственно. Узнал про минимум пять итерационных методов для решения подобных уравнений.**