МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ
ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине
«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА»

Вариант “2ав”

Выполнил:
Студент группы Р32312
Лысенко А.К.
Преподаватель:
Перл О.В.

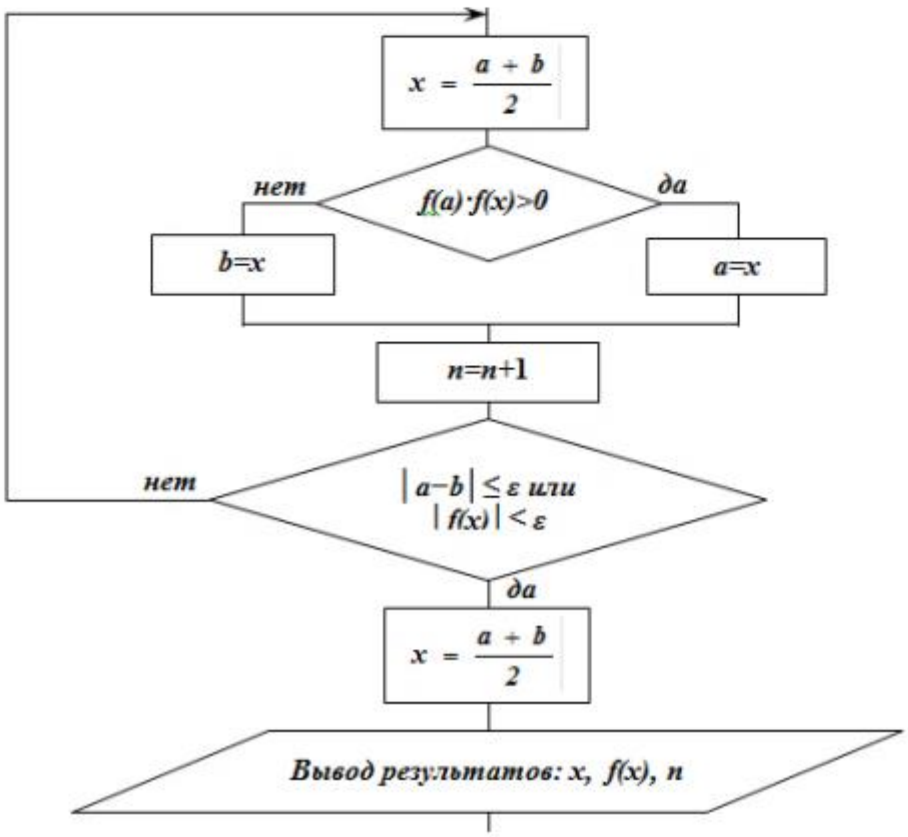
Санкт-Петербург, 2023

Описание метода, расчётные формулы

Решения нелинейных уравнений:

1 метод: метод деления пополам. Начальный интервал изоляции корня делим пополам, получаем начальное приближение к корню: $x_0 = \frac{(a_0+0)}{2}$. Вычисляем f(x0). В качестве нового интервала выбираем ту половину отрезка, на концах которого функция имеет разные знаки: [a0, x0], либо [b0,x0]. Другую половину отрезка [a0,b0], на которой функция f(x) знает не меняет, отбрасываем. Новый интервал вновь делим, получаем очередное приближение к корню: $x_1 = \frac{(a_1+b_1)}{2}$. Рабочая формула метода $x_i = \frac{(a_i+b_i)}{2}$. Критерий окончания итерационного процесса: $|b_i - a_i| \leq$ e.

Блок-схема



Достоинства: обладает абсолютной сходимостью, а также устойчив к ошибкам округления.

Недостатки: если интервал содержит несколько корней, то неизвестно к какому вычислительному относится вычислительный процесс.

Листинг численного метода

```
public double solveCubicEquation() {
    double r = Math.abs(super.getRight() - super.getLeft());
    double middle = 0;
    while (!(r < super.getEpsilon())) {
        middle = (super.getLeft() + super.getRight()) / 2;
        double k = super.getValueForCubicEquation(super.getLeft()) *
super.getValueForCubicEquation(middle);
        if (k > 0) {
            super.setLeft(middle);
        } else {
            super.setRight(middle);
        }
        r = Math.abs(super.getRight() - super.getLeft());
    }
    System.out.println("Нашли корень методом деления пополам  - " + middle);
    return middle;
}
```

Примеры и результаты работы

```
Что вы хотите сделать?
1 - решить систему линейных уравнений методом простой итерации
2 - решить нелинейные уравнения
2
1. ax^2 + bx + c = 0
2. ax^3 + bx^2 + cx + d = 0
3. sin(x) + c = 0
1
Введите коэффициенты перед переменными. Их должно быть 3 штуки.
1 -6 -6
Введите точность нахождения корня
0,001
Введите границы для нахождения корня
2 10
Нашли корень методом касательных - 6.872983363173406
Нашли корень методом деления пополам - 6.8720703125
Разница между решениями разными способами = 9.130506734056354E-4
Время работы алгоритма - 13 мс
```

2 метод – метод касательных: функция y = f(x) на отрезке [a,b] заменяется касательной и в качестве приближенного значения корня x* = x_n принимается точка пересечения касательной с осью абсцисс. Рабочая формула метода:

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}$$

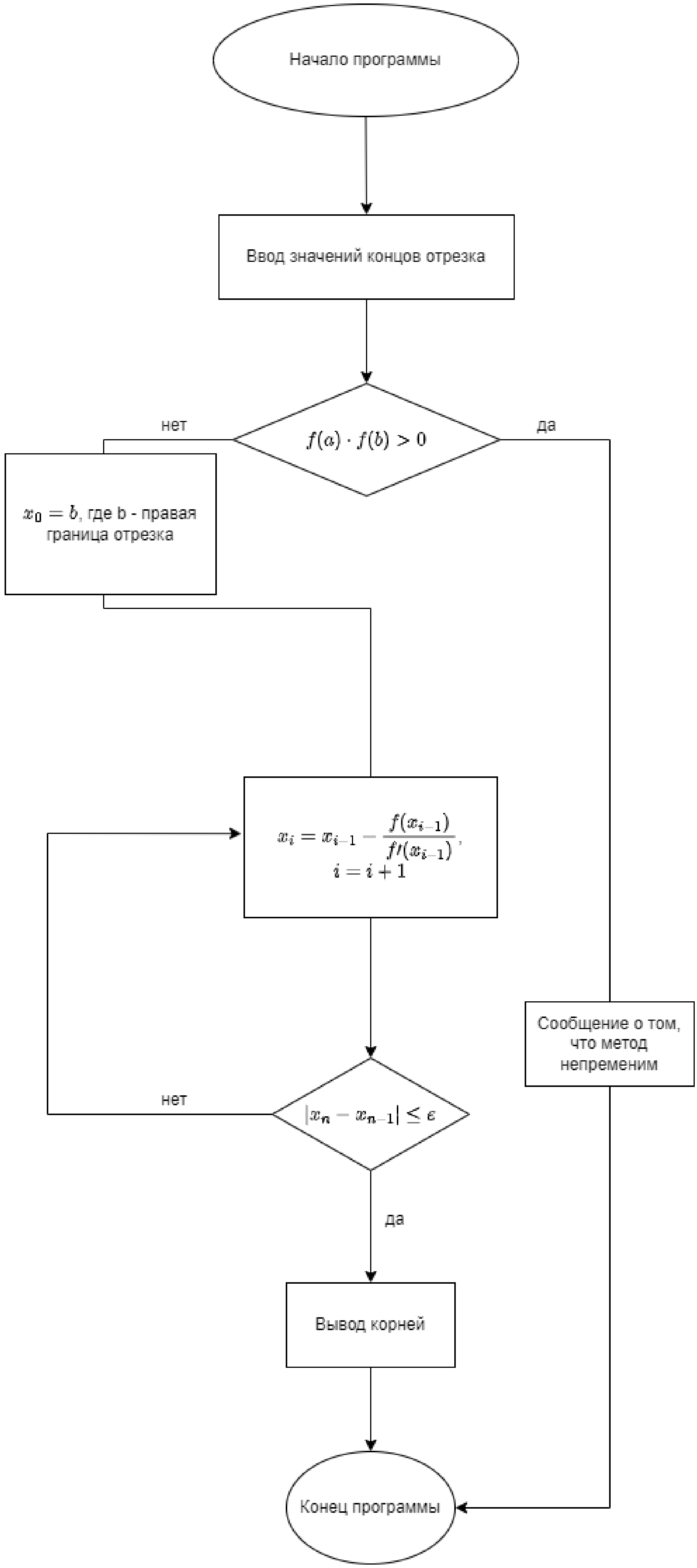
Критерий окончания итерационного процесса: $|x_i - x_{i-1}|$≤ e
Метод Ньютона применяется в том случае, если выполняются условия:

- Функция y=f(x) определена и непрерывна на отрезке [a;b]
- f(a)*f(b) < 0
- производные $f'(x)$ и $f''(x)$ сохраняют знак на отрезке [a;b]
- $f'(x) \,! = 0$

Выбор начального приближения x_0 в отрезке $[a; b]$
Метод обеспечивает быструю сходимость, если выполняется условие:
$f(x_0) * f''(x_0) > 0$

Недостатки: необходимость вычисления производной на каждой итерации
Достоинства: квадратичная сходимость

Блок-схема


Начало программы
Ввод значений концов отрезка
$f(a) \cdot f(b) > 0$
нет
да
$x_0 = b$, где b - правая граница отрезка
$x_i = x_{i-1} - \frac{f(x_{i-1})}{f\prime(x_{i-1})}$,
$i = i + 1$
Сообщение о том, что метод непременим
$|x_n - x_{n-1}| \leq \epsilon$
нет
да
Вывод корней
Конец программы

Листинг численного метода

```
public double solveQuadraticEquation() {
    double zeroApproximation = super.getRight();
    double nextApproximation = zeroApproximation -
((super.getValueForQuadraticEquation(zeroApproximation)) /
(getDerivativeForQuadraticEquation(zeroApproximation)));
    while (!(Math.abs(nextApproximation - zeroApproximation) <
super.getEpsilon())){
        zeroApproximation = nextApproximation;
        nextApproximation = zeroApproximation -
((super.getValueForQuadraticEquation(zeroApproximation)) /
(getDerivativeForQuadraticEquation(zeroApproximation)));
    }
    System.out.println("Нашли корень методом касательных - " +
nextApproximation);
    return nextApproximation;
}
```

Метод простой итерации

Данный метод помогает нам найти корни у системы нелинейных уравнений. Его суть заключается в том, что изначально мы приводим систему к эквивалентному виду:

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, \dots, x_n) \\ x_2 = \varphi_2(x_1, x_2, \dots, x_n) \\ \quad \dots\dots\dots\dots\dots\dots\dots. \\ x_n = \varphi_n(x_1, x_2, \dots, x_n) \end{cases}$$

Далее мы выбираем начальное приближение, а последующие приближения находятся по формулам:

$$\begin{cases} x_1^{(k+1)} = \varphi_1(x_1^k, x_2^k, \dots, x_n^k) \\ x_2^{(k+1)} = \varphi_2(x_1^k, x_2^k, \dots, x_n^k) \\ \quad \dots\dots\dots\dots\dots\dots\dots. \\ x_n^{(k+1)} = \varphi_n(x_1^k, x_2^k, \dots, x_n^k) \end{cases} \qquad k = 0, 1, 2, \dots$$

Достаточное условие сходимости итерационного процесса:

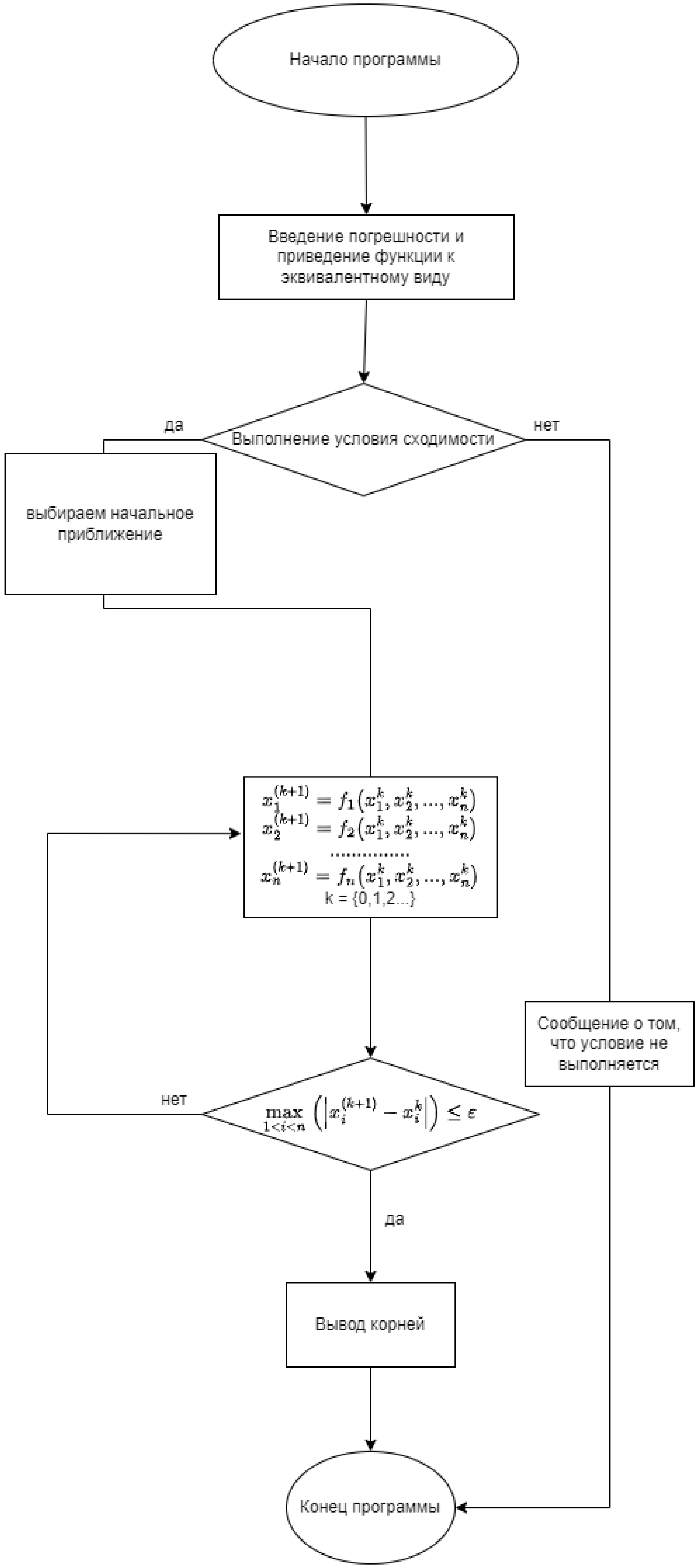
$$\max_{[x \in G]} |\varphi'(x)| \le q < 1$$

Если $X^{(0)} \in G$ и все последовательные приближения: $X^{(k+1)} = \varphi(X^k)$, $k = 0, 1, 2 \dots$ также содержатся в ограниченной замкнутой области G, тогда итерационный процесс сходится к единственному решению уравнения $X = \varphi(X)$

Критерий окончания итерационного процесса:

$$\max_{1<i<n} \left| x_i^{(k+1)} - x_i^k \right| \le \varepsilon$$

Блок-схема


Начало программы
Введение погрешности и приведение функции к эквивалентному виду
Выполнение условия сходимости
да
нет
выбираем начальное приближение
$x_1^{(k+1)} = f_1(x_1^k, x_2^k, ..., x_n^k)$
$x_2^{(k+1)} = f_2(x_1^k, x_2^k, ..., x_n^k)$
....
$x_n^{(k+1)} = f_n(x_1^k, x_2^k, ..., x_n^k)$
k = {0,1,2...}
Сообщение о том, что условие не выполняется
нет
$\max\limits_{1<i<n} \left(\left|x_i^{(k+1)} - x_i^k\right|\right) \leq \varepsilon$
да
Вывод корней
Конец программы

Листинг численного метода

```java
public void solveFirstSystem(){
    this.x0 = new double[]{-1.6, -3.7};
    this.x = new double[2];
    this.F = new double[2];
    doZeroApproximationForTheFirst();
    while (!(Math.max(Math.abs(x[0] - x0[0]), Math.abs(x[1] - x0[1])) <
super.getEpsilon())){
        functionsForTheFirstSystem(F, x);
        x0[0] = x[0];
        x0[1] = x[1];
        x[0] = F[0];
        x[1] = F[1];
    }
    System.out.println("Корни данной системы - " + Arrays.toString(x));
}
```

Примеры и результаты работы

```
Что вы хотите сделать?
1 - решить систему линейных уравнений методом простой итерации
2 - решить нелинейные уравнения
1
Выберите систему :
1. y = x^3
   y = x^2 - 6
2. x^3 + y^3 - 6x + 3 = 0
   x^3 - y^3 - 6y + 2 = 0
2
Введите погрешность
0,0001
Корни данной системы - [0.5323753111063038, 0.3512582521396515]
Время работы алгоритма - 5 мс
```

Вывод

Метод половинного деления позволяет нам найти корни на отрезке от a до b, при условии, что функция в данных точках имеет разный знак. При этом функция должна быть непрерывная и монотонна. Идея заключается в том, что мы на каждом шаге уменьшаем область поиска в 2 раза. В целом, метод очень напоминает бинарный поиск. Из достоинств можно отметить простоту, а также близость получаемого численного решения задачи к истинному решению. Из недостатков можно отметить, что при наличии на интервале нескольких корней, будет неясно к какому из них мы придем.

Метод касательных также позволяет найти нам найти корни в промежутке от a до b. Функция в данном методе заменяется касательной и в качестве приближенного значения корня принимается точка пересечения с осью абсцисс. Данный метод используется, если функция определена и непрерывна на отрезке, на концах отрезков функция имеет разные знаки, первая и вторая производные сохраняют знаки и первая производная не равна 0. Из недостатков можно отметить необходимость подсчета производной на каждом шаге.

Метод простой итерации для СНАУ помогает найти приближенные значения корней данной системы, так как метод является итерационным. Из больших недостатков можно отметить слабое условие сходимости – первое приближение должно быть достаточно близко к корням, чтобы решение точно нашлось. Также коэффициенты перед неизвестными должны быть меньше 1.