Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2 по "Вычислительной математике"

Численное решение нелинейных уравнений

Вариант 4

Метод хорд, метод секущих и метод простых итераций

Выполнил: Дьяконов Михаил Павлович
Группа: P3211
Преподаватель: Малышева Татьяна Алексеевна

Университет ИТМО, 2021

Цель работы

Реализовать 3 из 5 методов численного решения уравнений. Предусмотреть ввод и верификацию входных данных. Понять, как они работают и визуализировать каждый из них.

Описание методов

Метод хорд:
Функция y=f(x) на отрезке [a, b] заменяется хордой и в качестве приближенного значения корня принимается точка пересечения хорды с осью абсцисс. Рабочая формула метода:

$$x_i = \frac{a_i f(b_i) - b_i f(a_i)}{f(b_i) - f(a_i)}$$

Критерий окончания итерационного процесса: $|x_i - x_{i-1}| \leq \varepsilon$

Метод секущих:
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. f'(x) заменяем разностным приближением. Рабочая формула метода:

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

Критерий окончания итерационного процесса: $|x_i - x_{i-1}| \leq \varepsilon$

Метод простой итерации:
Уравнение $f(x) = 0$ приведем к эквивалентному виду $x = \phi(x)$.

$$x_1 = \phi(x_0), x_2 = \phi(x_1)...$$

Критерий окончания итерационного процесса: $|x_i - x_{i-1}| \leq \varepsilon$ (0<q$\leq$ 0.5),$|x_i - x_{i-1}| \leq \frac{1-q}{q}\varepsilon$ (0.5<q<1)

Листинг программы

ChordMethod.ts:

```
export class ChordMethod {

static calculate(func: MathFunction, a0: number, b0: number, fault: number): ChordMethodResult {
    let x0: number;
    const verificationResult = VerificationUtils.completeVerification(func, a0, b0);
    if (verificationResult !== undefined) return {errorMessage: verificationResult};

    if(func.fnc(a0)*func.secondDerivative(a0) > 0) x0 = a0;
    else if(func.fnc(b0)*func.secondDerivative(b0) > 0) x0 = b0;
    else {
        x0 = a0 - (b0 - a0)*func.fnc(a0)/(func.fnc(b0) - func.fnc(a0));
    }

    const aValues = [], bValues = [], xValues = [], funcA = [], funcB = [], funcX = [], faults =
        [], functions = [];
    let a = a0, b = b0, x = x0;
    do {
        aValues.push(a);
        bValues.push(b);
        funcA.push(func.fnc(a));
        funcB.push(func.fnc(b));
        x = a - (b - a)*func.fnc(a)/(func.fnc(b) - func.fnc(a));
        xValues.push(x);
        funcX.push(func.fnc(x));

        // change interval
        if(func.fnc(x)*func.fnc(a) < 0) b = x;
```

```
            else a = x;

            // замыкания )0))
            const aCopy = a, bCopy = b;
            functions.push((x: number) => (x - aCopy)*(func.fnc(bCopy) -
                func.fnc(aCopy))/(bCopy-aCopy) + func.fnc(aCopy));
            const currentFault = xValues.length > 1 ?
               Math.abs(xValues[xValues.length - 1] - xValues[xValues.length - 2]) : null;
            faults.push(currentFault);
        } while (faults[faults.length - 1] == null || faults[faults.length - 1] > fault);

        return {aValues: aValues, bValues: bValues, xValues: xValues, funcA: funcA, funcB: funcB,
            funcX: funcX, faults: faults, functions: functions};
    }
}
```

SecantMethod.ts:

```
    export class SecantMethod {

    static calculate(func: MathFunction, a0: number, b0: number, x0: number, fault: number):
        SecantMethodResult {
        const verificationResult = VerificationUtils.completeVerification(func, a0, b0);
        if (verificationResult !== undefined) return {errorMessage: verificationResult};

        if (func.fnc(x0) * func.secondDerivative(x0) < 0)
            return {errorMessage: 'Начальное приближениевыбранонеправильно'};

        const x1 = a0 + (b0 - a0)/2;
        const xValues = [], nextXValues = [], funcXNext = [], prevXValues = [], faults = [], functions
            = [];
        let x = x1, prevX = x0;
        do {
            prevXValues.push(prevX);
            xValues.push(x);
            const nextX = x - func.fnc(x) * (x - prevX)/(func.fnc(x) - func.fnc(prevX));
            nextXValues.push(nextX);
            funcXNext.push(func.fnc(nextX));

            const xCopy = x;
            functions.push((x: number) => func.fnc(xCopy) + func.derivative(xCopy) * (x - xCopy));
            faults.push( Math.abs(nextXValues[nextXValues.length - 1] - xValues[xValues.length - 1]));
            prevX = x;
            x = nextX;
        } while (faults[faults.length - 1] > fault);

        return {xValues: xValues, nextXValues: nextXValues, funcXNext: funcXNext, prevXValues:
            prevXValues,
            faults: faults, functions: functions};
    }
}
```

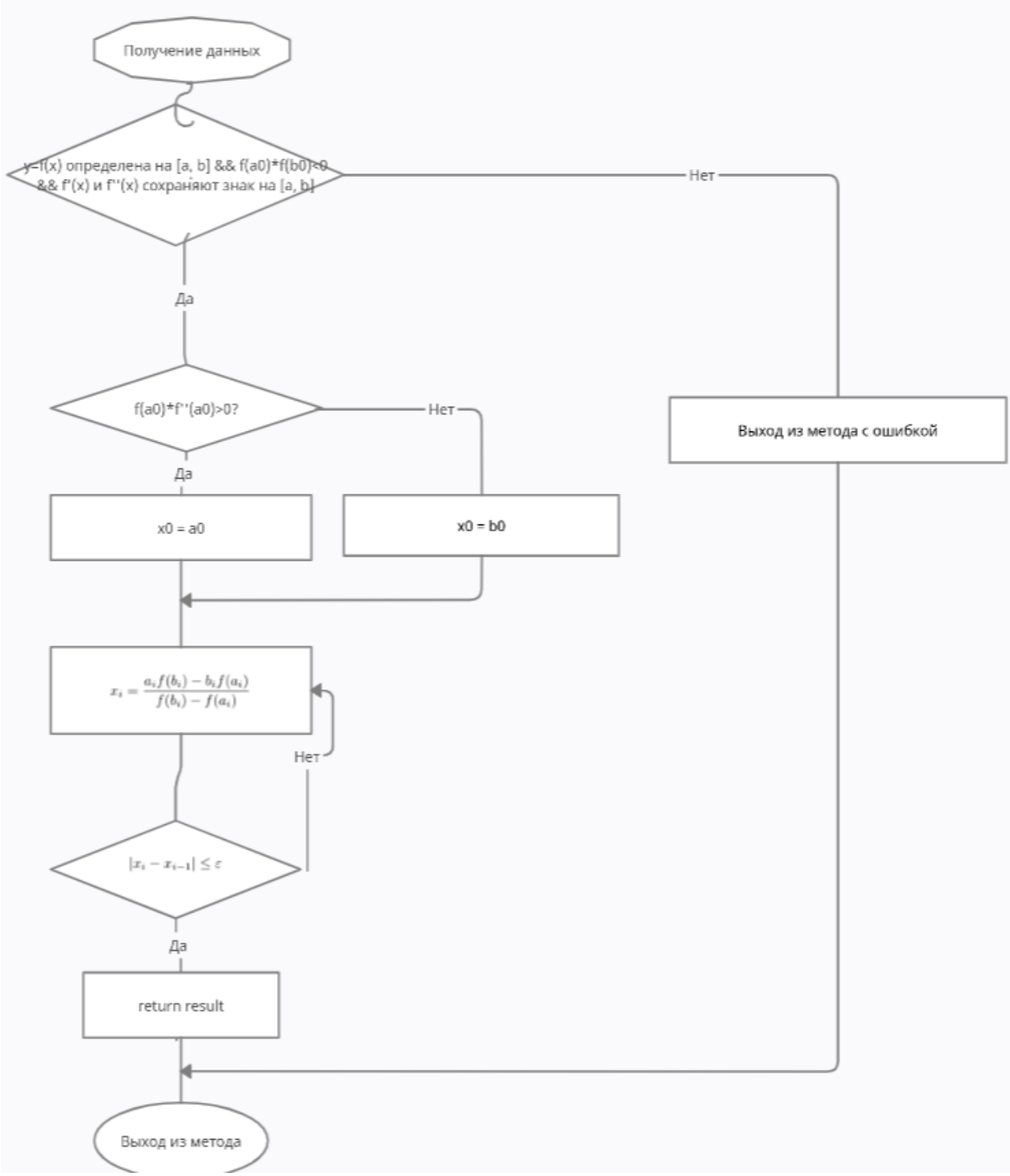
SimpleIterationMethod.ts:

```
export class SimpleIterationMethod {

    static calculate(func: MathFunction, a0: number, b0: number, x0: number, fault: number):
        SimpleIterationMethodResult {
        const verificationResult = VerificationUtils.completeVerification(func, a0, b0);
        if (verificationResult !== undefined) return {errorMessage: verificationResult};

        const lambda = -1/Math.max(Math.abs(func.derivative(a0)), Math.abs(func.derivative(b0)));
        const xFunc = (x: number) => x + lambda*func.fnc(x);
```

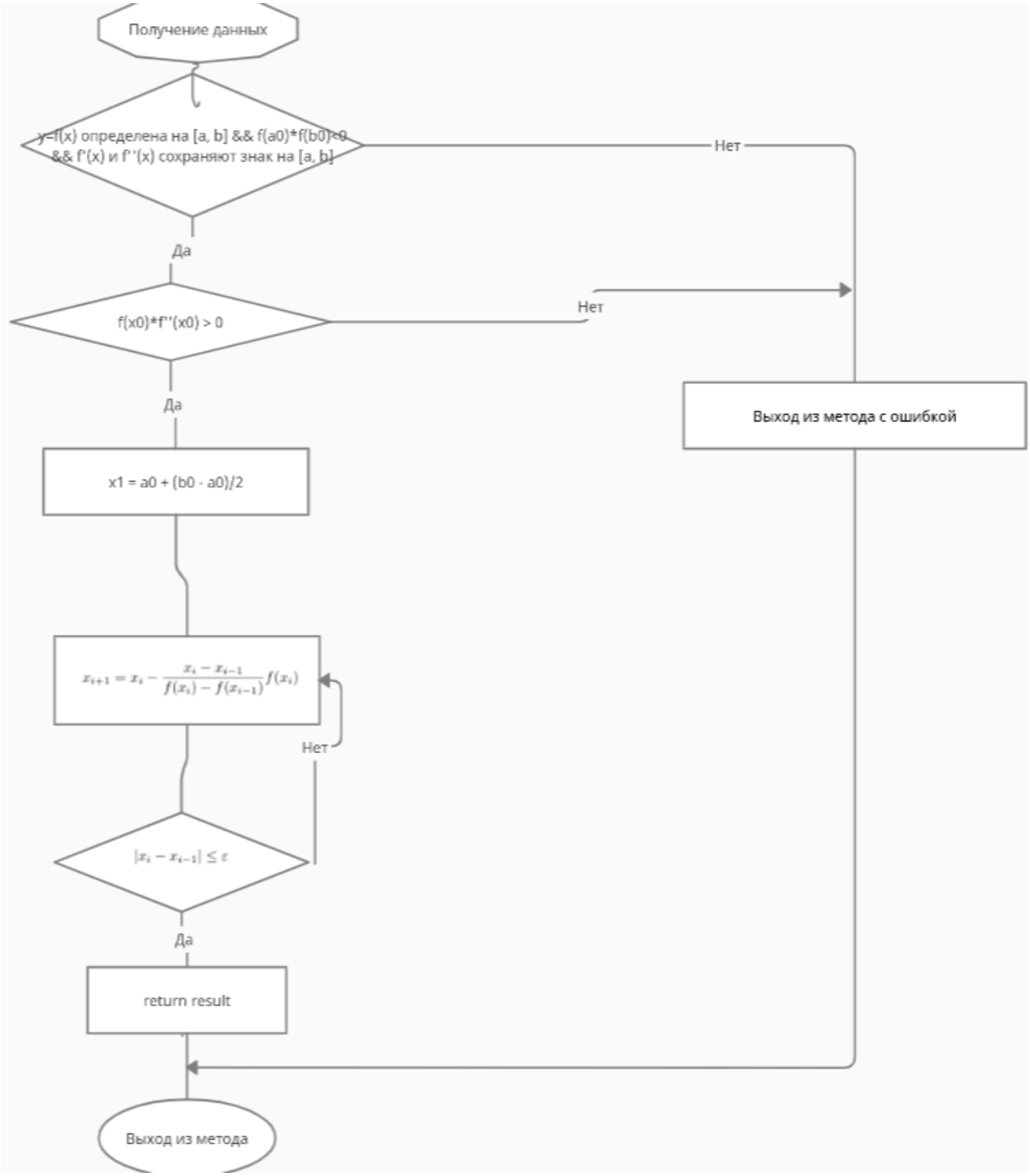
```
        const xFuncDerivative = (x: number) => 1 + lambda*func.derivative(x);

        const q = Math.max(Math.abs(xFuncDerivative(a0)), Math.abs(xFuncDerivative(b0)));
        if(q >= 1) return {errorMessage: "Достаточное условиеметоданевыполняется"};
        fault = q <= 0.5 ? fault : (1-q)*fault/q;

        const xValues = [], nextXValues = [], xFuncNext = [], funcNext = [], faults = [],
            functions = [(x: number) => x, xFunc];
        let x = x0;
        do {
            xValues.push(x);
            const nextX = xFunc(x);
            nextXValues.push(nextX);
            xFuncNext.push(xFunc(nextX));
            funcNext.push(func.fnc(nextX));
            faults.push(Math.abs(nextX - x));
            x = nextX;
        } while (faults[faults.length - 1] > fault);

        return {xValues: xValues, nextXValues: nextXValues, xFuncNext: xFuncNext, funcNext: funcNext,
            faults: faults, functions: functions};
    }
}
```

Блок-схемы методов

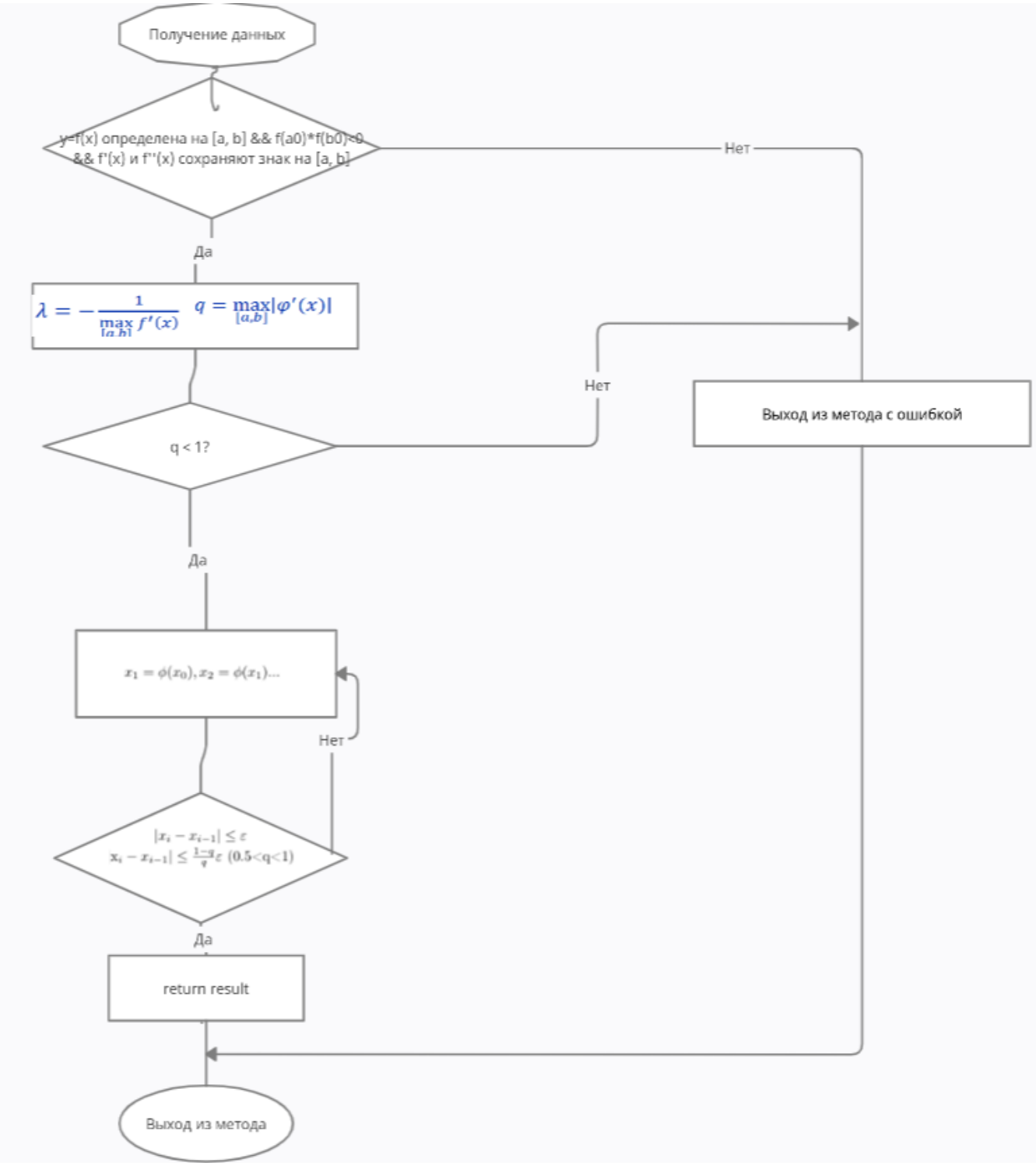
Метод хорд:



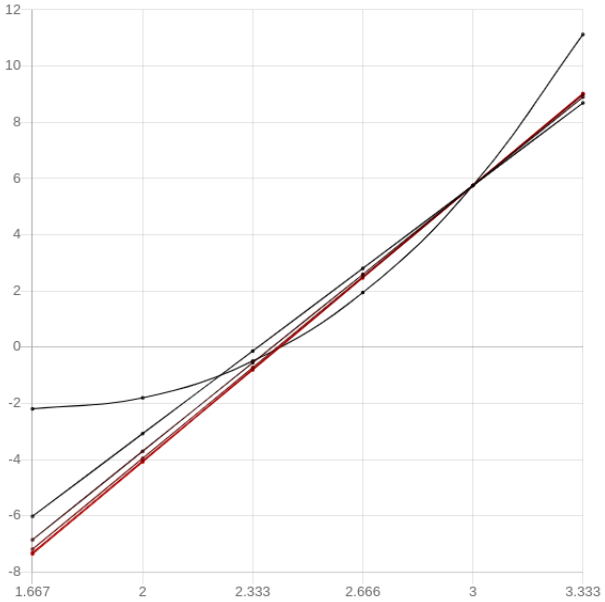
Метод секущих:



Метод простой итерации:



Примеры



Выберите метод

- (•) Метод хорд
- ( ) Метод секущих
- ( ) Метод простой итерации

Выберите функцию

- (•) x^3 - 1.89x^2 - 2x + 1.76
- ( ) 3sin(3x) + 1.5
- ( ) ln(2.5x) - 3.7
- ( ) -x^5 + 1.8x^4 - 2x^3 + 3x + 6

Выберите формат ввода и вывода

- ( ) Из файла
- (•) Из формы

Введите границы интервала

От 2 До 3

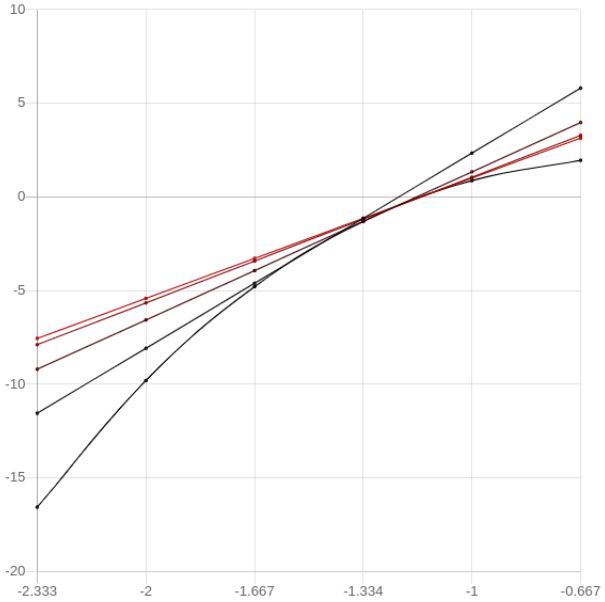
Введите начальное приближение

Введите погрешность

0.01

[Найти корень]

№ итерации	a	b	x	F(a)	F(b)	F(x)	Δ
0	2	3	2.23841	-1.8	5.75	-0.97112	-
1	2.23841	3	2.34845	-0.97112	5.75	-0.40845	0.11004
2	2.34845	3	2.39166	-0.40845	5.75	-0.15379	0.043213
3	2.39166	3	2.40751	-0.15379	5.75	-0.05548	0.015846
4	2.40751	3	2.41317	-0.05548	5.75	-0.01971	0.005662



Выберите метод

- ○ Метод хорд
- ◉ Метод секущих
- ○ Метод простой итерации

Выберите функцию

- ◉ x^3 - 1.89x^2 - 2x + 1.76
- ○ 3sin(3x) + 1.5
- ○ ln(2.5x) - 3.7
- ○ -x^5 + 1.8x^4 - 2x^3 + 3x + 6

Выберите формат ввода и вывода

- ○ Из файла
- ◉ Из формы

Введите границы интервала

От -2 До -1

Введите начальное приближение

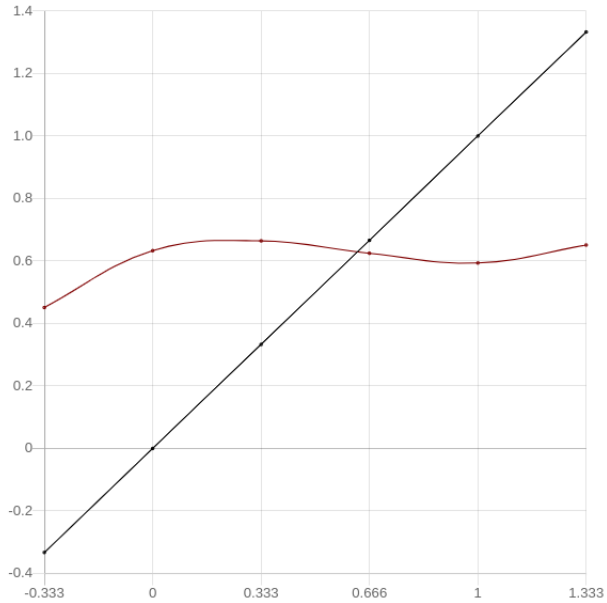
-2

Введите погрешность

0.01

Найти корень

№ итерации	x_{i-1}	x_i	x_{i+1}	$F(x_{i+1})$	Δ
0	-2	-1.5	-1.29318	-0.97695	0.206816
1	-1.5	-1.29318	-1.18631	-0.19677	0.106873
2	-1.29318	-1.18631	-1.15936	-0.01994	0.026955
3	-1.18631	-1.15936	-1.15632	-0.00049	0.00304



Выберите метод

- ○ Метод хорд
- ○ Метод секущих
- ◉ Метод простой итерации

Выберите функцию

- ◉ x^3 - 1.89x^2 - 2x + 1.76
- ○ 3sin(3x) + 1.5
- ○ ln(2.5x) - 3.7
- ○ x^4 - 5x^3 + 8x^2 - 5x + 1

Выберите формат ввода и вывода

- ○ Из файла
- ◉ Из формы

Введите границы интервала

От 0 До 1

Введите начальное приближение

0.5

Введите погрешность

0.01

Найти корень

№ итерации	x_i	x_{i+1}	$\varphi(x_{i+1})$	$F(x_{i+1})$	Δ
0	0.5	0.64838	0.62725	-0.05874	0.148381
1	0.64838	0.62725	0.63037	0.00867	0.021128
2	0.62725	0.63037	0.62991	-0.00128	0.003119

Вывод

В результате выполнения данной лабораторной работы я узнал какие есть численные методы решения уравнений. Также узнал о преимуществах и недостатках каждого из них и реализовал три из них.