

Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Дисциплина: «Вычислительная математика»

Лабораторная работа №2

Вариант:

метод деления пополам

метод касательных

метод Ньютона

Выполнил: Кизилев Степан Александрович,
группа Р32312

Преподаватель: Перл Ольга Вячеславовна

1 Описание методов

1.1 Метод деления пополам

Идея заключается в том, что если на концах отрезка функция имеет разные знаки, то где-то на промежутке она имеет корень. Поэтому мы почти что бинарным поиском ищем промежуток, каждый раз деля прошлый пополам.

1.2 Метод касательных

Идея заключается в том, мы постепенно приближаем значение корня x_k , проводя касательные в точке $(x_k, f(x_k))$. После этого, x_{k+1} будет располагаться в точке пересечения касательной и оси Ох:

$$f' = \frac{\Delta f}{\Delta x}$$
$$\Delta x = \frac{\Delta f}{f'}$$
$$x_{k+1} = x_k - \Delta x = x_k - \frac{f(x_k) - 0}{f'(x_k)}$$

1.3 Метод Ньютона

Идея как в методе касательных, только у нас много функций.
Сотворим разложение в Тейлора:

$$F_1(x_1, y_1) = F_1(x_0, y_0) + \frac{\partial F_1}{\partial x} \Delta x + \frac{\partial F_1}{\partial y} \Delta y$$
$$F_2(x_1, y_1) = F_2(x_0, y_0) + \frac{\partial F_2}{\partial x} \Delta x + \frac{\partial F_2}{\partial y} \Delta y$$

Так как мы ищем корни, то левая часть должна быть нулевой. Тогда:

$$-F_1(x_0, y_0) = \frac{\partial F_1}{\partial x} \Delta x + \frac{\partial F_1}{\partial y} \Delta y$$
$$-F_2(x_0, y_0) = \frac{\partial F_2}{\partial x} \Delta x + \frac{\partial F_2}{\partial y} \Delta y$$

Решая систему, получим:

матр. эквив.

$$\begin{pmatrix} \frac{\partial F_1}{\partial x} & \frac{\partial F_1}{\partial y} \\ \frac{\partial F_2}{\partial x} & \frac{\partial F_2}{\partial y} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} F_1 \\ F_2 \end{pmatrix}$$

$$= \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} \frac{\partial F_1}{\partial x} & \frac{\partial F_1}{\partial y} \\ \frac{\partial F_2}{\partial x} & \frac{\partial F_2}{\partial y} \end{pmatrix}^{-1} \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} =$$

$$= - \frac{1}{J} \cdot \begin{pmatrix} \frac{\partial F_2}{\partial y} & - \frac{\partial F_1}{\partial y} \\ - \frac{\partial F_2}{\partial x} & \frac{\partial F_1}{\partial x} \end{pmatrix} \begin{pmatrix} F_1 \\ F_2 \end{pmatrix}$$

$$\Delta x = - \frac{1}{J} \left(F_1 \cdot \frac{\partial F_2}{\partial y} - F_2 \cdot \frac{\partial F_1}{\partial y} \right)$$

$$\Delta y = - \frac{1}{J} \left(F_1 \cdot \frac{\partial F_2}{\partial x} - F_2 \cdot \frac{\partial F_1}{\partial x} \right) \cdot (-1)$$

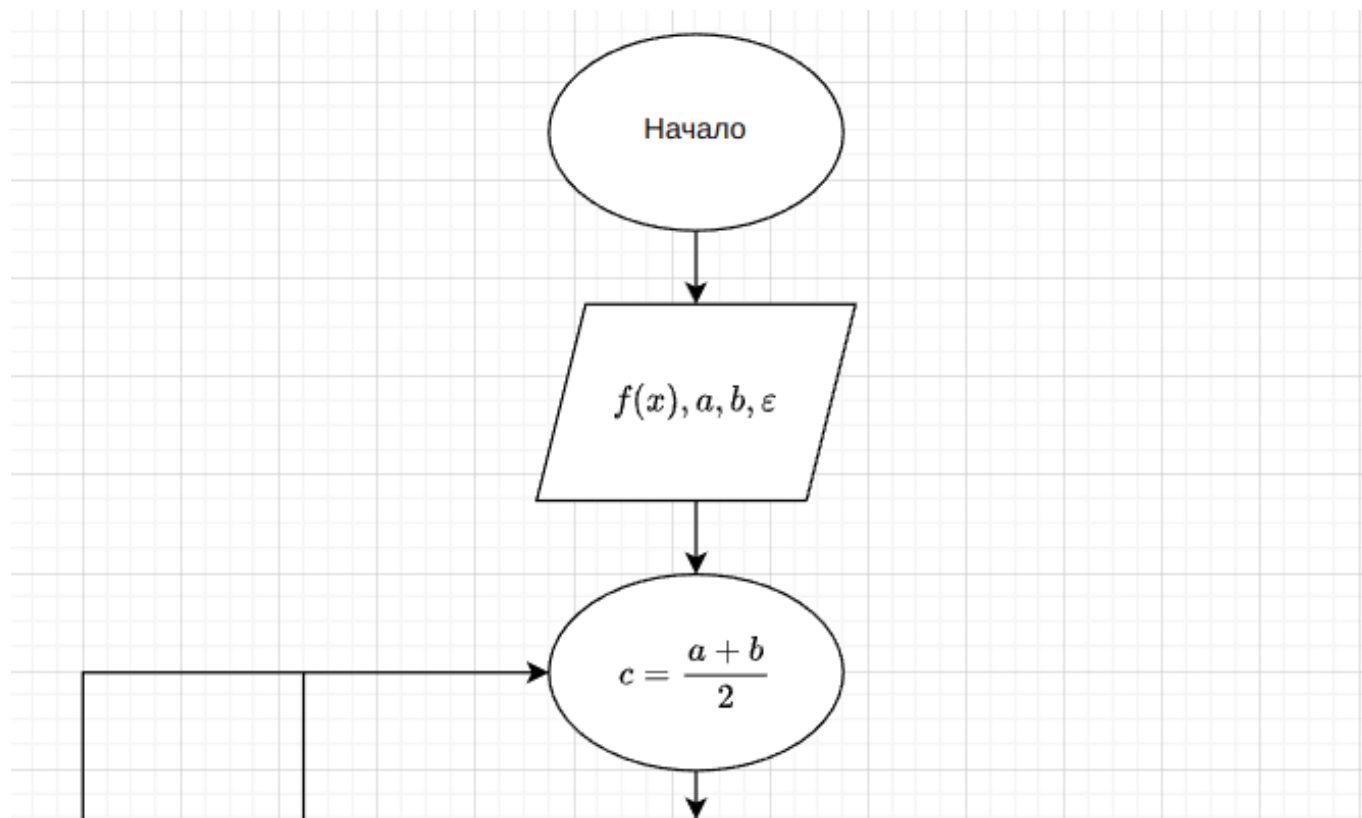
Таким образом,

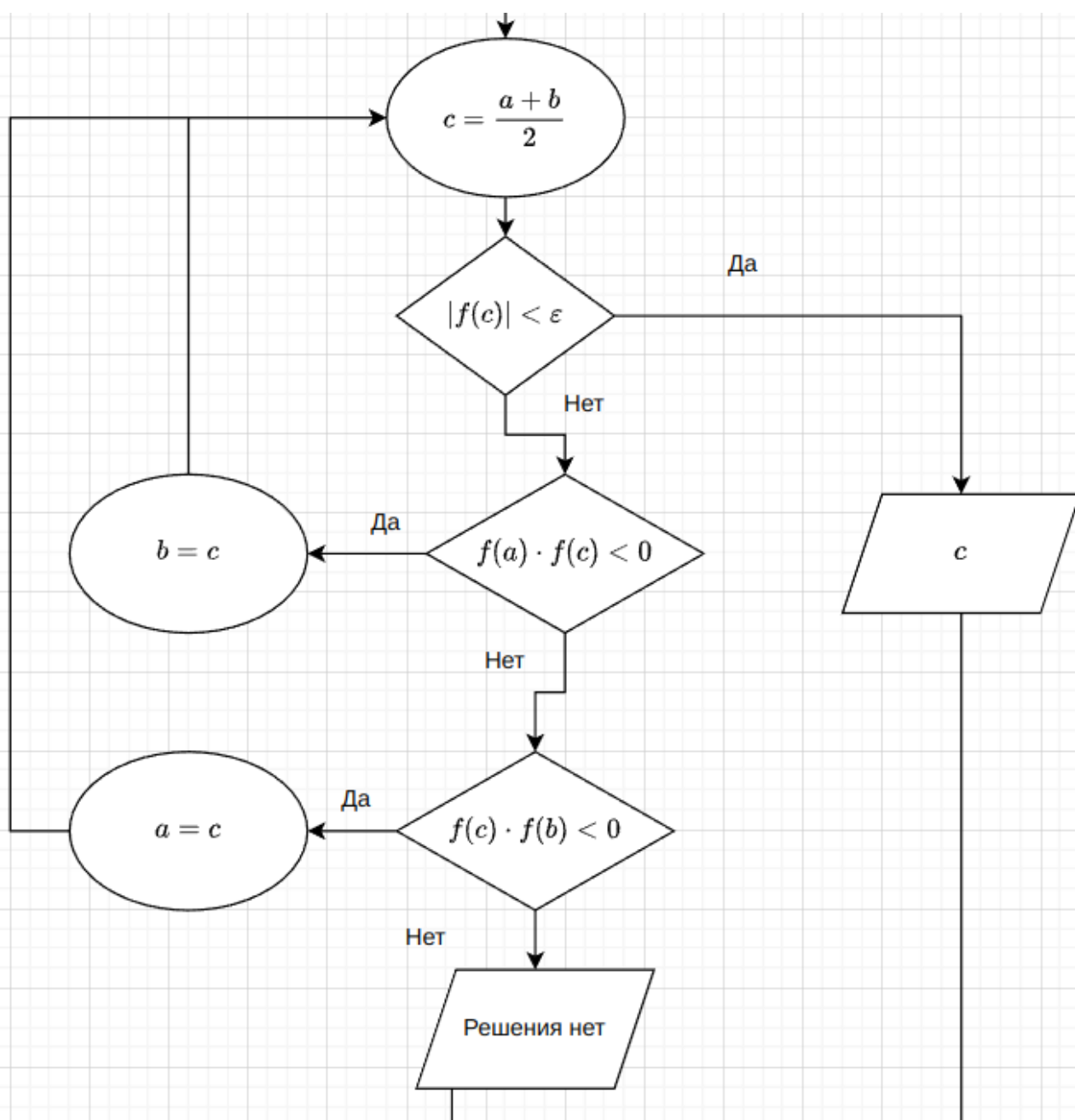
$$x_{k+1} = x_k + \Delta x$$

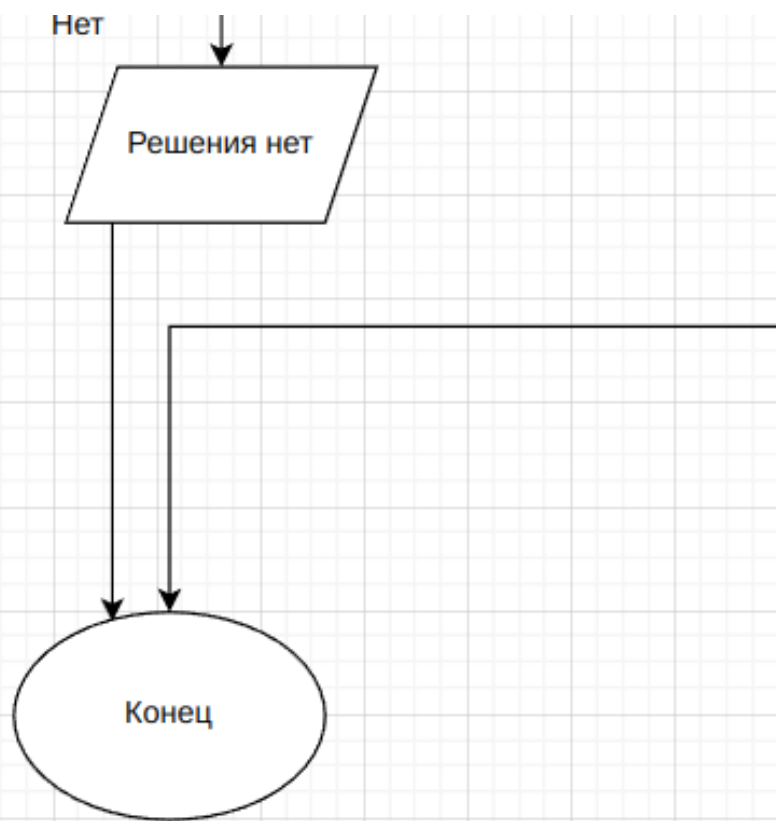
$$y_{k+1} = y_k + \Delta y$$

2 Блок-схема метода

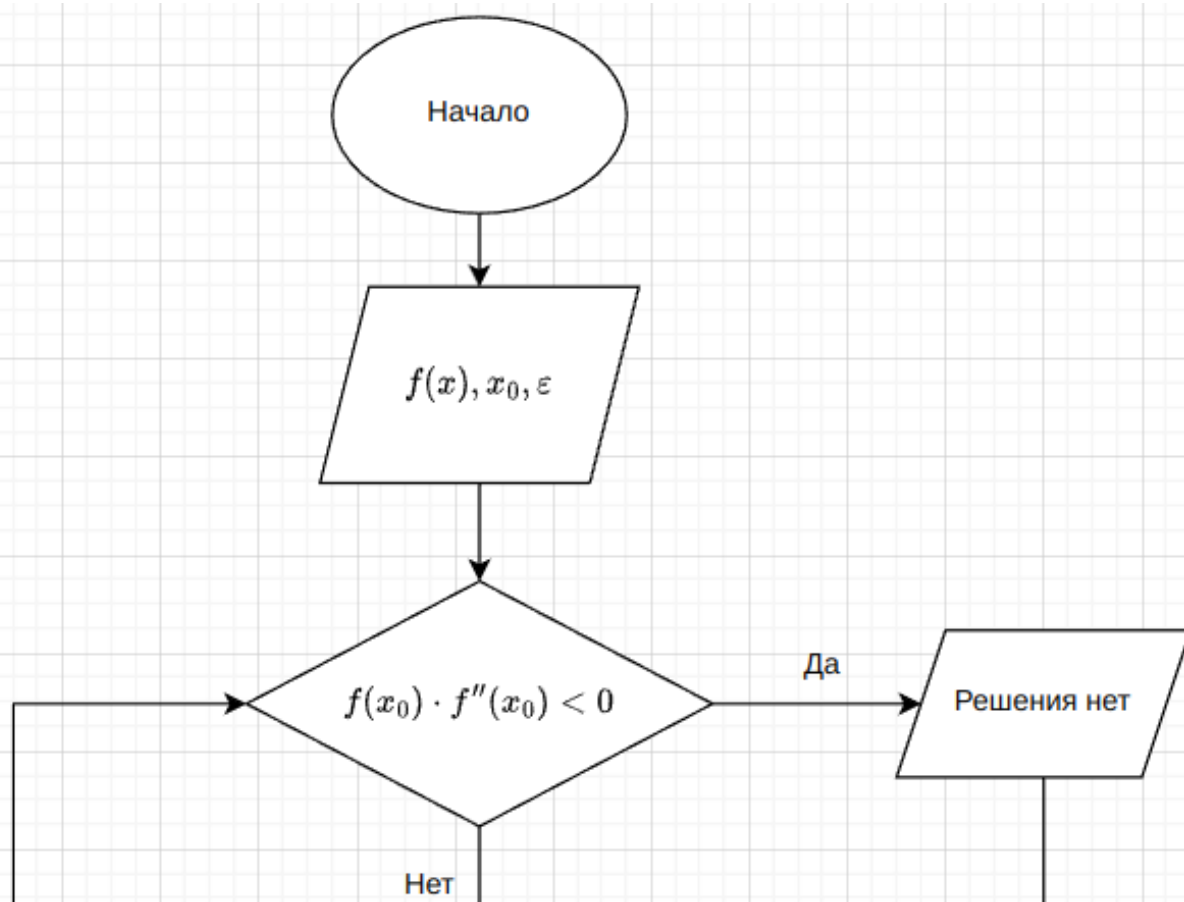
2.1 Метод деления пополам

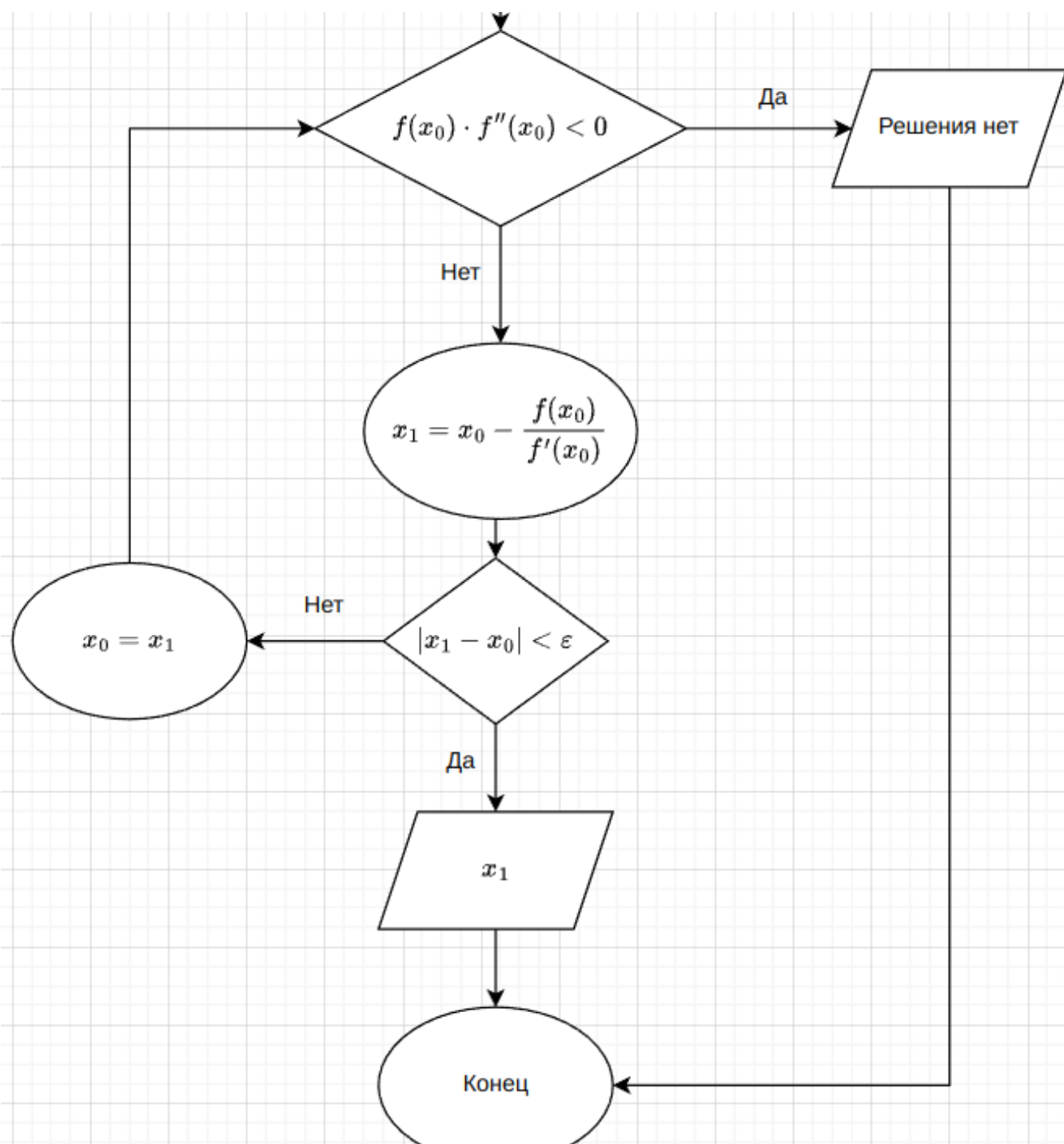




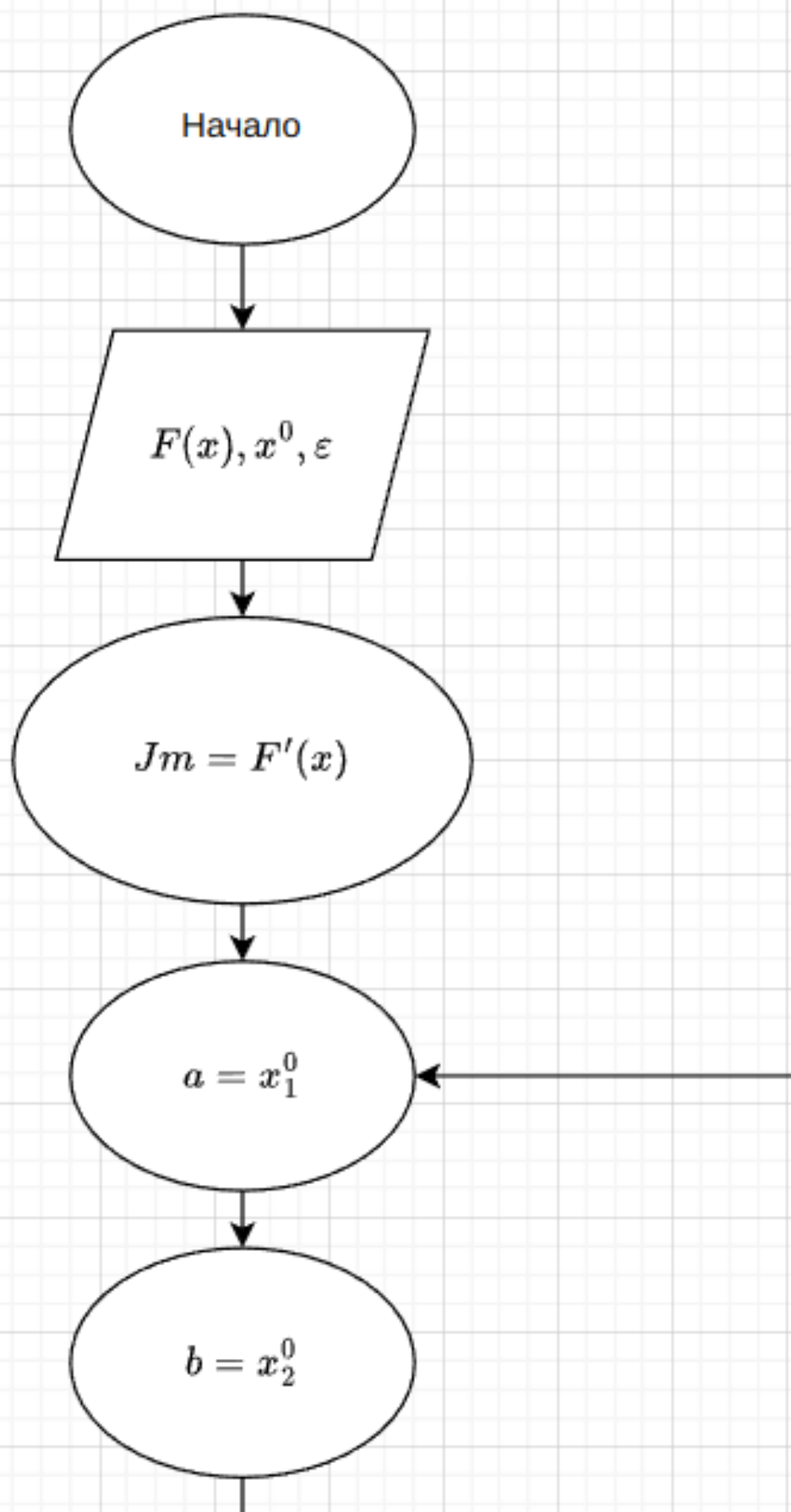


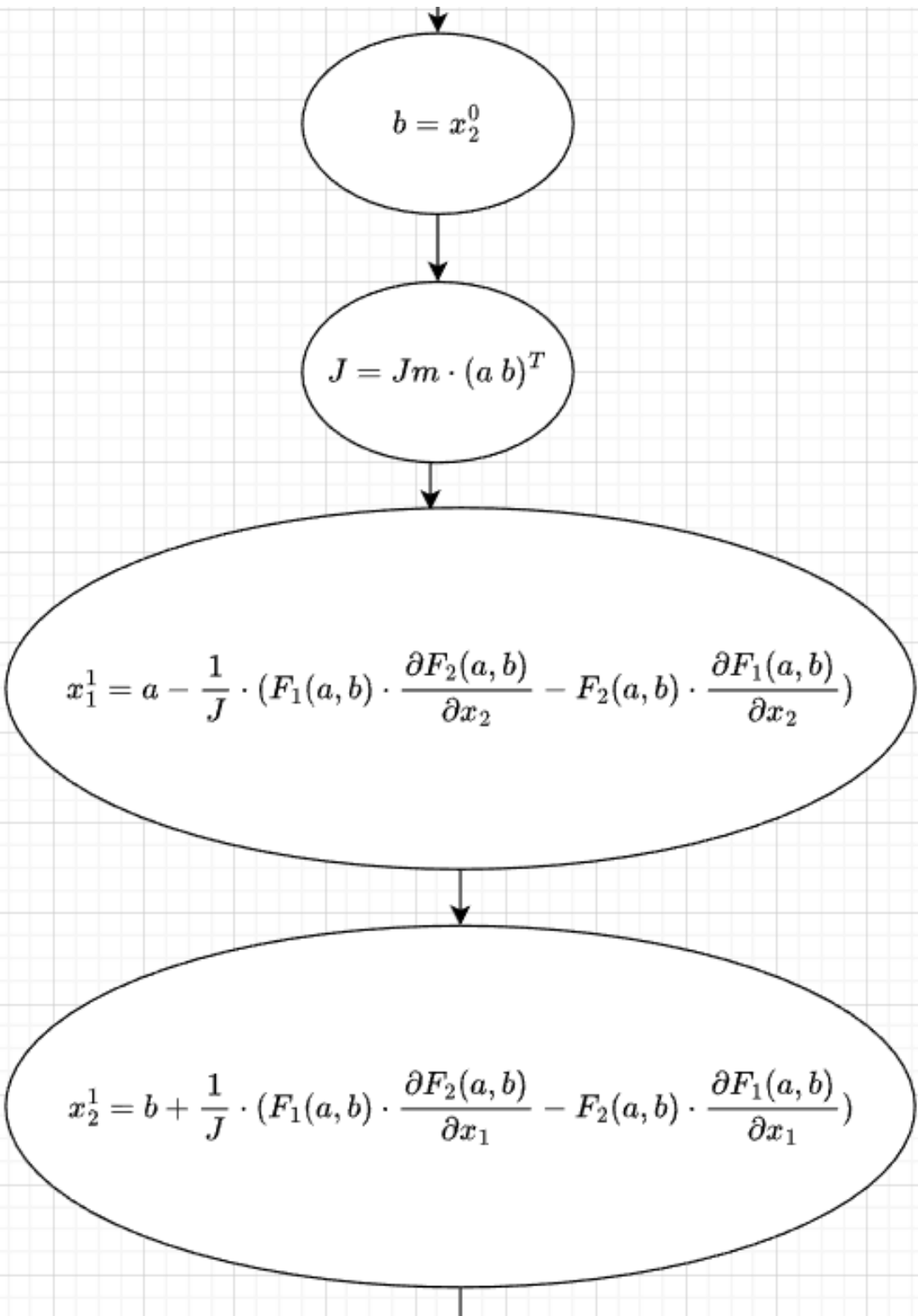
2.2 Метод касательных





2.3 Метод Ньютона





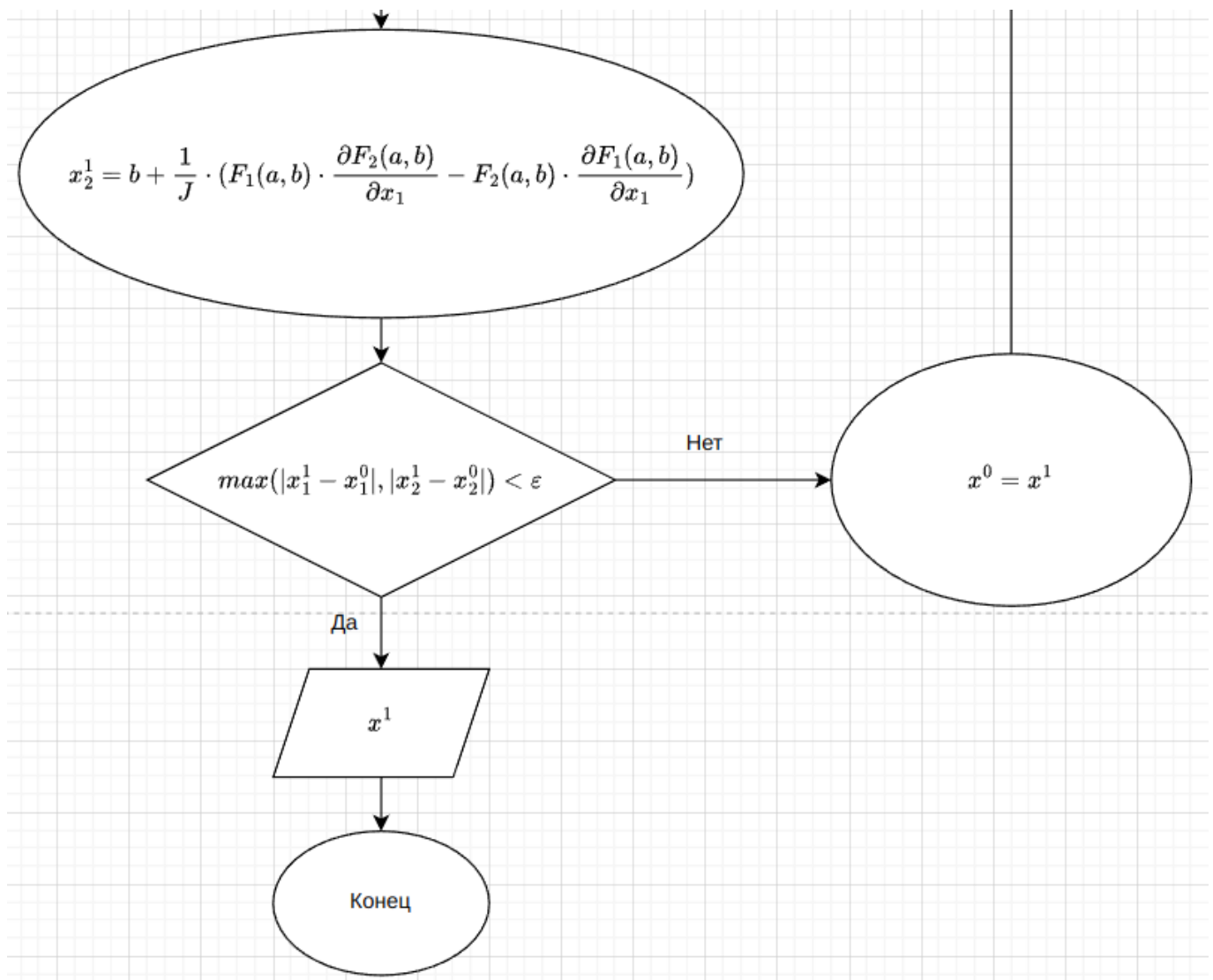
```
graph TD; A([b = x_2^0]) --> B([J = Jm · (a b)^T]); B --> C([x_1^1 = a - 1/J · (F_1(a, b) · ∂F_2(a, b)/∂x_2 - F_2(a, b) · ∂F_1(a, b)/∂x_2)]); C --> D([x_2^1 = b + 1/J · (F_1(a, b) · ∂F_2(a, b)/∂x_1 - F_2(a, b) · ∂F_1(a, b)/∂x_1)]); D --> E[|];
```

$b = x_2^0$

$J = Jm \cdot (a \ b)^T$

$$x_1^1 = a - \frac{1}{J} \cdot (F_1(a, b) \cdot \frac{\partial F_2(a, b)}{\partial x_2} - F_2(a, b) \cdot \frac{\partial F_1(a, b)}{\partial x_2})$$

$$x_2^1 = b + \frac{1}{J} \cdot (F_1(a, b) \cdot \frac{\partial F_2(a, b)}{\partial x_1} - F_2(a, b) \cdot \frac{\partial F_1(a, b)}{\partial x_1})$$



3 Исходный код

3.1 Метод деления пополам

```

def bisection_method(eq: Equation, a_value: float, b_value: float, eps: float = 10 ** (-14)) -> float | None:
    func = eq.function

    while True:
        med = (a_value + b_value) / 2

        if -eps < func(med) < eps:
            return med

        if func(a_value) * func(med) < 0:
            b_value = med
            continue

        elif func(med) * func(b_value) < 0:
            a_value = med
            continue

        else:
            return
  
```

3.2 Метод касательных

```
def tangent_method(eq: Equation, x0: float, eps: float = 10 ** (-10)) -> float | None:
    func = eq.function

    if func(x0) * numeric_derivative2(func, x0) < 0:
        return None

    prev_result = x0
    while True:
        result = prev_result - func(prev_result) / numeric_derivative(func, prev_result)

        if abs(result - prev_result) < eps:
            return result

        prev_result = result
```

3.3 Метод Ньютона

```
def newton_method(equation_list: list[SysEquation], x0: list[float], eps: float = 10 ** (-5)) -> list[float] | None:
    jacobian_matrix = [sys_eq.jacobian_row for sys_eq in equation_list]

    prev_result = copy.deepcopy(x0)

    counter = 0

    while True:
        a_val, b_val = prev_result
        J = get_jacobian(jacobian_matrix, prev_result)
        new_a = a_val - (1 / J) * (
            equation_list[0].function(a_val, b_val) * equation_list[1].jacobian_row[1](a_val, b_val) -
            equation_list[1].function(a_val, b_val) * equation_list[0].jacobian_row[1](a_val, b_val)
        )

        new_b = b_val + (1 / J) * (
            equation_list[0].function(a_val, b_val) * equation_list[1].jacobian_row[0](a_val, b_val) -
            equation_list[1].function(a_val, b_val) * equation_list[0].jacobian_row[0](a_val, b_val)
        )

        result = [new_a, new_b]

        if is_vector_diff_less_than_eps(result, prev_result, eps):
            return result

        counter = counter + 1
        if counter > 300_000:
            return None

        prev_result = copy.deepcopy(result)
```

4 Примеры и результаты работы

4.1 Метод Ньютона

```
Choose task:
1: solve equation
2: solve system
>>> 2
Choose equations:
0)  $x^2 - y = 0$ 
1)  $5*x - y^3 = 0$ 
2)  $1/x + y^2 = 0$ 
>>> 0 1
Enter x_0 and y_0, like:
1.25 3.21
>>> 0.5 0.5
System:
 $x^2 - y = 0$ 
 $5*x - y^3 = 0$ 
Solution:
x = 0.0000
y = -0.0000
```

4.2 Уравнение 1

Choose task:

1: solve equation

2: solve system

>>> 1

Choose equation:

0) $x^2 - 4 = 0$

1) $e^x - 17 = 0$

2) $x^3 - x + 11 = 0$

3) $\sin(x) * x^2 - 3 = 0$

>>> 0

Enter a and b, like:

1.25 3.21

>>> 1 5

Equation: $x^2 - 4 = 0$

Bisection method solution: 2.0000

Tangent method solution: 2.0000

4.3 Уравнение 2

```
Choose task:
1: solve equation
2: solve system
>>> 1

Choose equation:
0)  $x^2 - 4 = 0$ 
1)  $e^x - 17 = 0$ 
2)  $x^3 - x + 11 = 0$ 
3)  $\sin(x) * x^2 - 3 = 0$ 
>>> 1

Enter a and b, like:
1.25 3.21
>>> -5 5

Equation:  $e^x - 17 = 0$ 
Bisection method solution: 2.8332
Tangent method solution: 2.8332
```

5 Вывод

В ходе выполнения работы реализовали 3 метода для решения НАО и СНАО.

Для методов существуют ограничения:

Метод деления пополам требует разных знаков функции на концах отрезка

Метод касательных требует положительности второй производной функции

Метод Ньютона требует, чтобы якобиан не был равен нулю. Методы мне очень понравились, было интересно их реализовывать.