МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

**МОДЕЛИРОВАНИЕ СЛУЧАЙНЫХ ВЕЛИЧИН С ЗАДАННЫМ ЗАКОНОМ РАСПРЕДЕЛЕНИЯ**

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к лабораторной работе №1

по дисциплине

**Вычислительная Математика**

РУКОВОДИТЕЛЬ:

Суркова Анна Сергеевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

СТУДЕНТ:

Цветков Николай Максимович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

19-ИВТ-3

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2020

Оглавление

[Цель 3](#_Toc65613027)

[Постановка задачи 4](#_Toc65613028)

[Теоретические сведения 5](#_Toc65613029)

[Критерий остановки 5](#_Toc65613030)

[Метод биссекции 6](#_Toc65613031)

[Метод хорд 8](#_Toc65613032)

[Метод Ньютона 10](#_Toc65613033)

[Метод простых итераций 13](#_Toc65613034)

[Расчетные данные 15](#_Toc65613035)

[Листинг разработанной программы 17](#_Toc65613036)

[Результат работы программы 20](#_Toc65613037)

[Вывод 21](#_Toc65613038)

# Цель

Закрепление знаний и умений по нахождению решений нелинейных уравнений различными способами.

# Постановка задачи

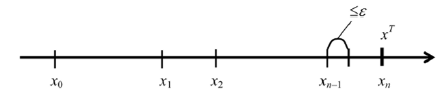
Решить нелинейное уравнение с одним неизвестным с использованием четырех методов (метод биссекции, метод хорд, метод Ньютона, метод простой итерации). Задание по вариантам. Номер варианта – номер студента в списке группы. ε=0.001

**Вариант № 7:**

# Теоретические сведения

# Критерий остановки

Процесс нахождения оптимального решения чаще всего имеет итерационный характер, т.е. последовательность {𝑥0, 𝑥1, …, 𝑥𝑛 → 𝑥} стремится к точному решению при увеличении кол-ва итераций n.



Весьма важным элементом всех итерационных методов является критерий (правило) остановки итерационного процесса. Именно критерий определяет точность достижения решения, а соответственно и эффективность метода.

Наиболее распространённые критерии остановки:

1.𝑓(𝑥)=0 – найдено точное решение

2.|𝑓(𝑥𝑛)|< 𝜀 – найдена заданная точность функции

3.|𝑓(𝑥𝑛− 𝑥𝑛+1)|< 𝜀 – значение двух последовательных приближений отличаются меньше, чем на 𝜀

# Метод биссекции

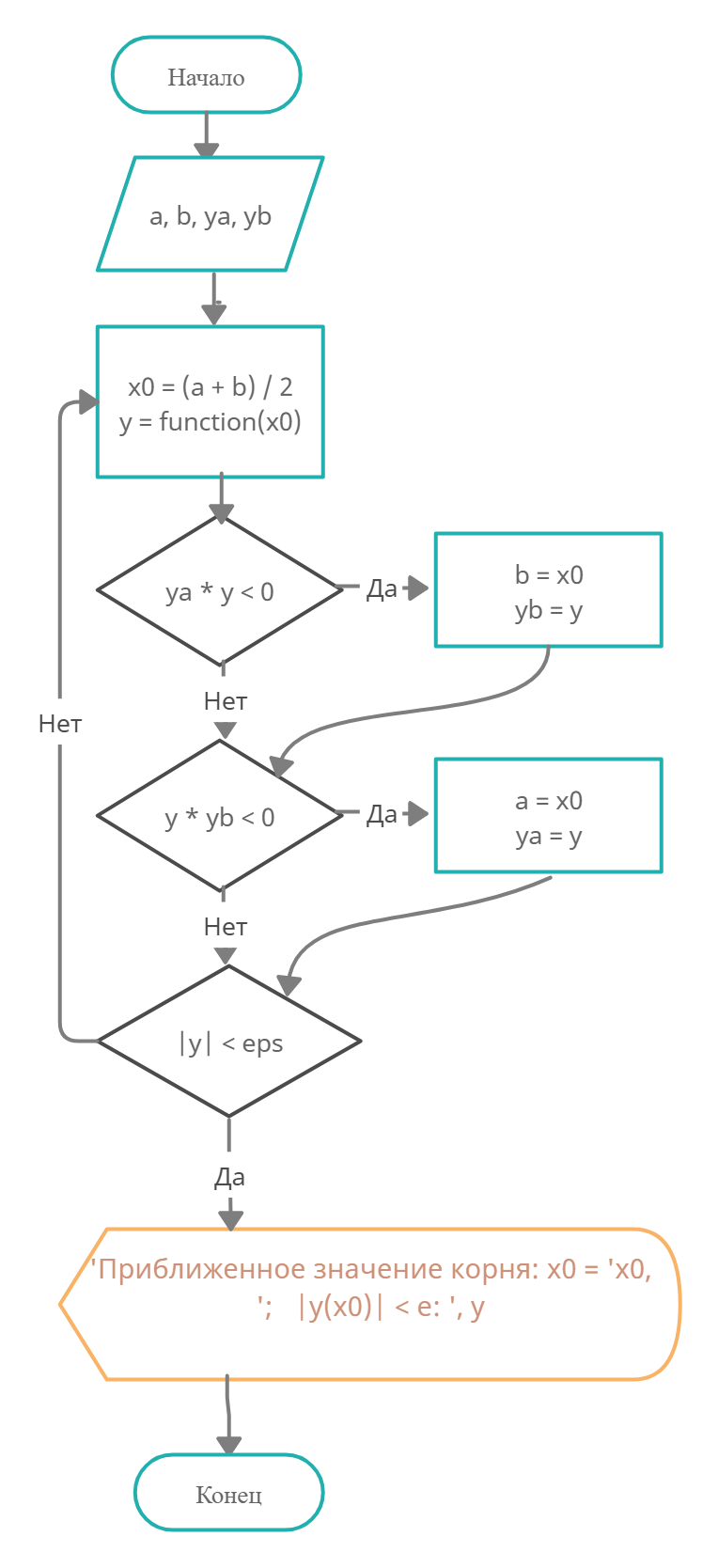
Пусть был выбран интервал изоляции [𝑎;𝑏]. Примем за первое приближение корня точку c, которая является серединой отрезка [𝑎;𝑏]. Далее будем действовать по следующему алгоритму:

1. Находим точку ;

2. Находим значение 𝑓(𝑥𝑖);

3. Если 𝑓(𝑎) ∗ 𝑓(𝑥𝑖) < 0, то корень лежит на интервале [𝑎; 𝑥𝑖], иначе корень лежит на интервале [𝑥𝑖;𝑏];

4. Если величина интервала меньше либо равна 𝜀, либо разница двух последовательных приближений меньше, либо равна 𝜀, то найдено решение с точностью до 𝜀 иначе возвращаемся к п.1.



# Метод хорд

Этот метод отличается от метода биссекции тем, что очередное приближение берём не в середине отрезка, а в точке пересечения с осью X прямой, соединяющей точки (𝑎, 𝑓(𝑎)) и (𝑏, 𝑓(𝑏)).

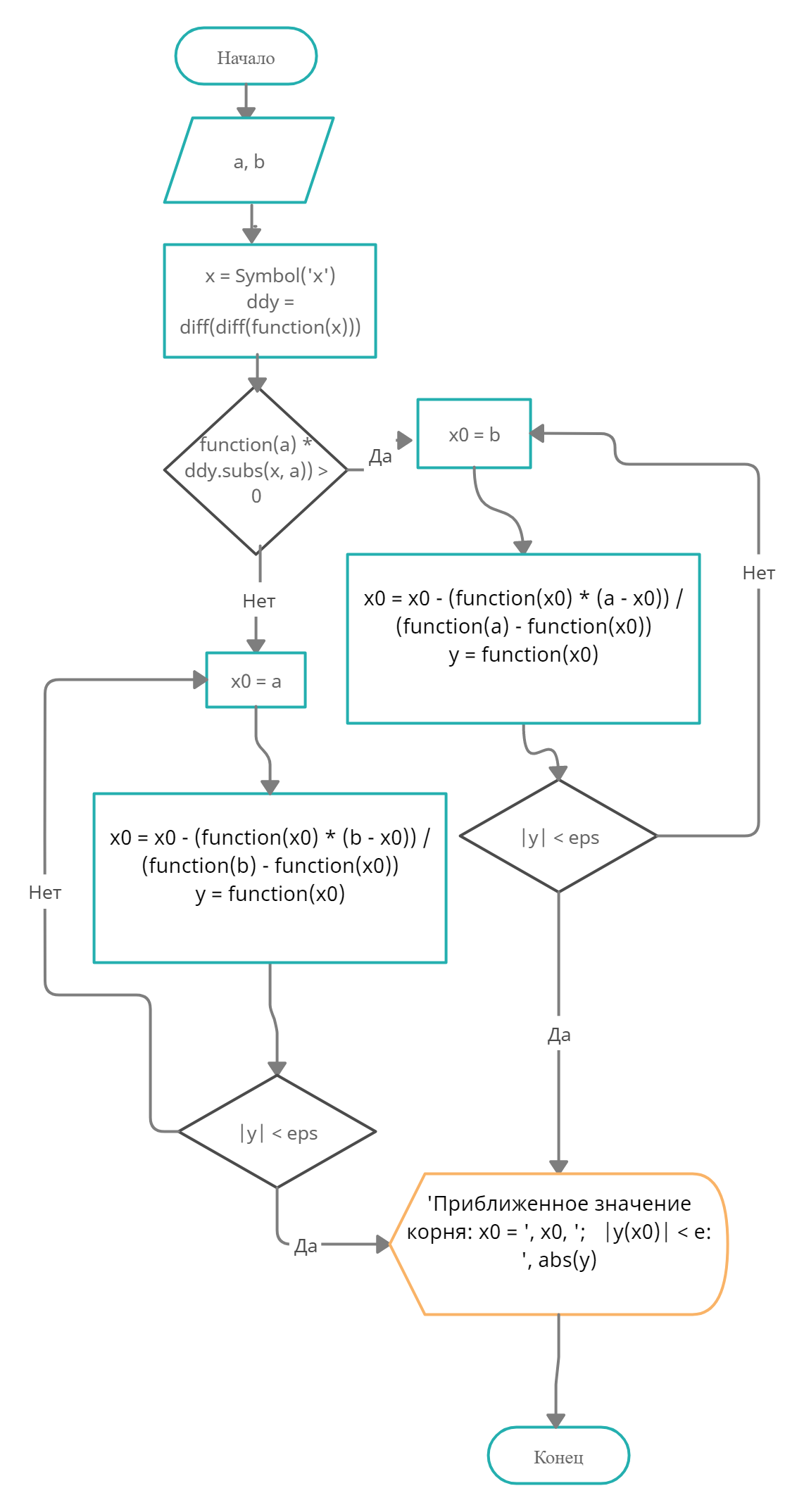
Запишем уравнение прямой, проходящей через точки с координатами точки (𝑎, 𝑓(𝑎)) и (𝑏, 𝑓(𝑏)):

Прямая, заданная уравнением, пересекает ось X при условии y = 0.

Найдём точку пересечения хорды с осью X:

 Далее необходимо вычислить значение функции в точке 𝑋𝑖. Это и будет приближённое значение корня уравнения.

Для вычисления одного из корней уравнения методом хорд достаточно знать интервал изоляции корня и точность вычисления 𝜀.



# Метод Ньютона

В одной из точек интервала [a; b], пусть это будет точка a, проведём касательную. Запишем уравнение этой прямой:

Так как эта прямая является касательной, и она проходит через точку (𝑥𝑖, 𝑓(𝑥𝑖)), то 𝑘 = 𝑓′(𝑥𝑖).

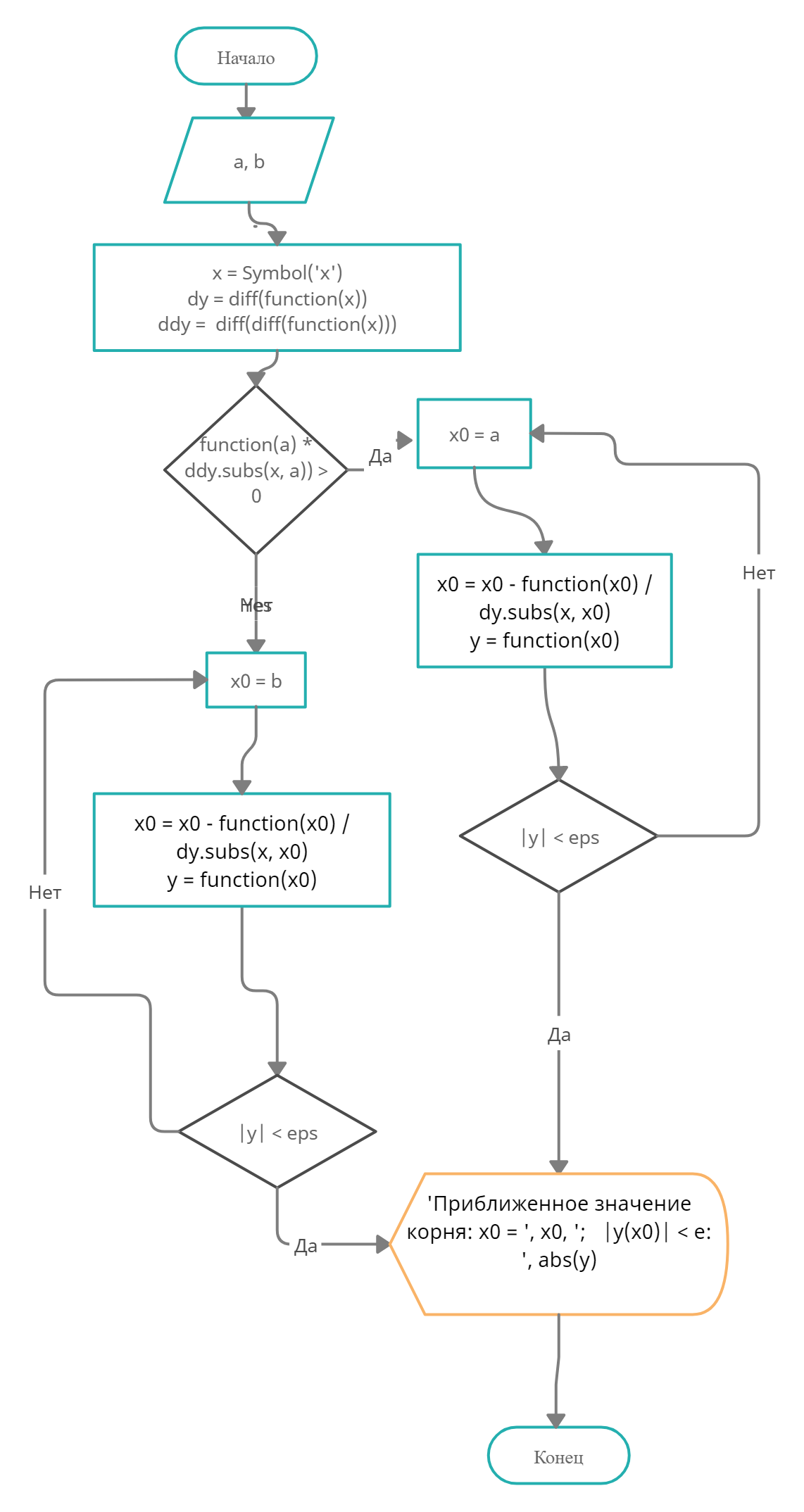
Следовательно,

Найдём точку пересечения касательной с осью X:

Если |𝑓(𝑥) < 𝜀|, то точность достигнута, и точка X — решение; иначе необходимо переменной c присвоить значение X и провести касательную через новую точку 𝑋𝑖; так продолжать до тех пор, пока |𝑓(𝑥)| не станет меньше 𝜀. Осталось решить вопрос, что выбрать в качестве точки начального приближения 𝑋𝑖.

В этой точке должны совпадать знаки функции и её второй производной. А так как нами было сделано допущение, что вторая и первая производные не меняют знак, то можно проверить условие 𝑓(𝑥)𝑓′′(𝑥)>0 на обоих концах интервала, и в качестве начального приближения взять ту точку, где это условие выполняется.

Здесь, как и в предыдущих методах, для вычисления одного из корней уравнения достаточно знать интервал изоляции корня и точность вычисления 𝜀.

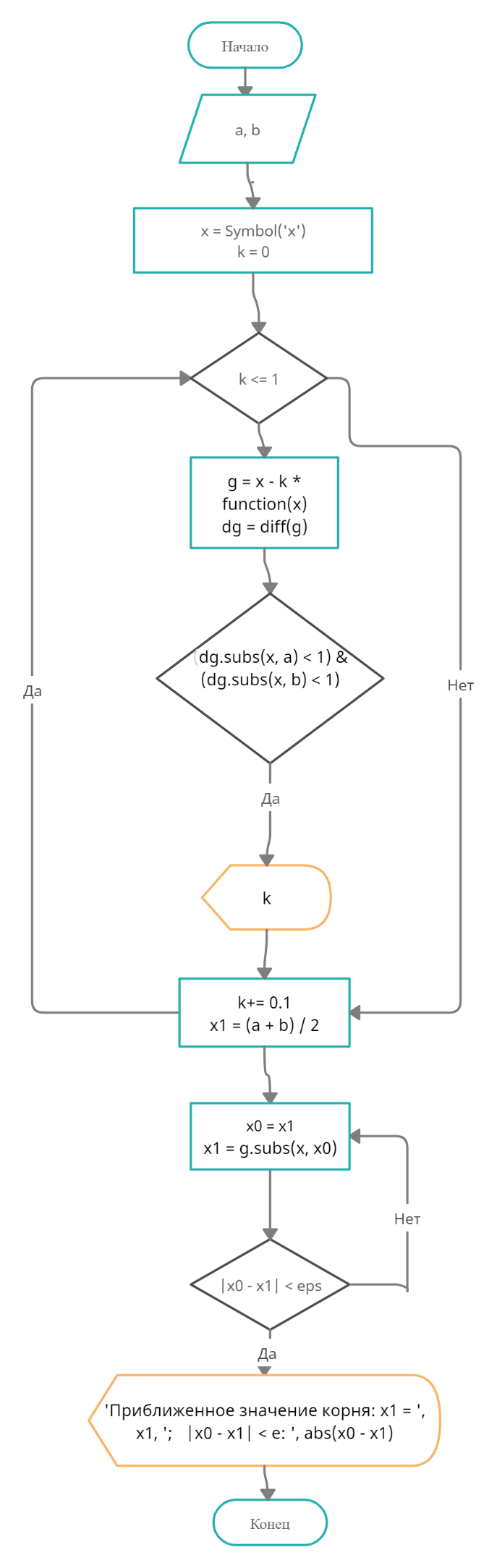


# Метод простых итераций

Для решения уравнения этим методом необходимо записать уравнение в виде 𝑥=𝜑(𝑥), задать начальное приближение 𝑥0 ∈[𝑎; 𝑏] и организовать следующий итерационный вычислительный процесс: 𝑥𝑖= 𝑐(𝑥),𝑖=0,1,2,…,𝑛 сходится к решению 𝑋∗

Если неравенство |𝜑′(𝑥)| < 1 выполняется на всём интервале [a; b], то последовательность 𝑥0,𝑥1,𝑥2,…,𝑥𝑛 сходится к решению.

Уравнение можно привести к виду 𝑥= 𝜑(𝑥) следующим образом. Умножить обе части уравнения 𝑓(𝑥)=0 на число 𝛽. К обеим частям уравнения 𝛽 𝑓(𝑥)=0 добавить число x. Получим 𝑥=𝑥+ 𝛽 𝑓(𝑥).

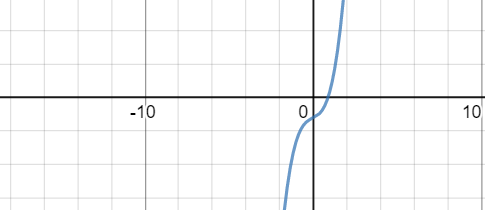


# Расчетные данные

Исходное уравнение:

Сжимающее уравнение:

График функции:

**

|  |  |
| --- | --- |
| **Метод** | **Приближенное значение корня** |
| Метод биссекции | 0.85547 |
| Метод хорд | 0.85512 |
| Метод Ньютона | 0.85566 |
| Метод простой итерации | 0.85363 |

# Листинг разработанной программы

**Lab\_1.py**

import SolvingNonlinearEquations as sne

x = -100

y = sne.function(x)

for x in range(-100,101):# берем значения x от -100 до 100, так как считаем, что функция заданна графически

    x0 = x-1 # x0 и y0 - временные переменные для хранения предыдущих значений

    y0 = y

    y = sne.function(x)

    if y0 \* y < 0: # проверка, где функция сменит знак

        print( 'Функция меняет знак в точках:\n', x0, ' ', y0, '\n', x, ' ', y)

        a = x0 #a, b - границы полученного интервала

        b = x

        ya = y0 # значения функции в данных точках

        yb = y

# Метод биссекции

sne.BisectionMethod(a, b, ya, yb)

# Метод хорд

sne.ChordMethod(a, b)

# Метод Ньютона

sne.NewtonsMethod(a, b)

# Метод простой итерации

sne.SimpleIterationMethod(a, b)

**SolvingNonlinearEquations.py**

from sympy import \*

def function (x): # изначально заданная функция

     y = x\*\*3 + 0.2 \* x\*\*2 + 0.5 \* x - 1.2

     return y

eps = 0.001 # заданное значение погрешности

# Метод биссекции

def BisectionMethod(a, b, ya, yb):

    print("Метод биссекции:")

    while True:

        x0 = (a + b) / 2  # начальное приближение x0

        y = function(x0)

        if ya \* y < 0: #Выбираем нужный отрезок

            b = x0

            yb = y

        if y \* yb < 0:

            a = x0

            ya = y

        if abs(y) < eps: # условие выхода из цикла |y(x0)| < e

            break

    print('Приближенное значение корня: x0 = ', x0, ';   |y(x0)| < e: ', y)

# Метод хорд

def ChordMethod(a, b):

    print("Метод хорд:")

    x = Symbol('x')

    ddy = diff(diff(function(x))) #берем вторую производную по заданной функции

    if (function(a) \* ddy.subs(x, a)) > 0: # проверяем неподвижность точки a

        x0 = b

        while True:

            x0 = x0 - (function(x0) \* (a - x0)) / (function(a) - function(x0)) # вычисляем начальное приближение по заданной формуле

            y = function(x0) # значение функции в данной точке

            if abs(y) < eps: # условие выхода из цикла |y(x0)| < e

                break

    else: # если точка b неподвижна, то двигается точка a

        x0 = a

        while True:

            x0 = x0 - (function(x0) \* (b - x0)) / (function(b) - function(x0)) # вычисляем начальное приближение по заданной формуле

            y = function(x0) # значение функции в данной точке

            if abs(y) < eps: # условие выхода из цикла |y(x0)| < e

                break

    print('Приближенное значение корня: x0 = ', x0, ';   |y(x0)| < e: ',"%f" %abs(y))

# Метод Ньютона

def NewtonsMethod(a, b):

    print("Метод Ньютона:")

    x = Symbol('x')

    dy = diff(function(x)) # берем производную по y

    ddy = diff(diff(function(x))) #берем вторую производную по заданной функции

    if (function(a) \* ddy.subs(x, a)) > 0: # проверяем неподвижность точки a

        x0 = a

        while True:

            x0 = x0 - function(x0) / dy.subs(x, x0) # вычисляем начальное приближение по заданной формуле

            y = function(x0) # значение функции в данной точке

            if abs(y) < eps: # условие выхода из цикла |y(x0)| < e

                break

    else: # проверяем неподвижность точки b

        x0 = b

        while True:

            x0 = x0 - function(x0) / dy.subs(x, x0) # вычисляем начальное приближение по заданной формуле

            y = function(x0) # значение функции в данной точке

            if abs(y) < eps: # условие выхода из цикла |y(x0)| < e

                break

    print('Приближенное значение корня: x0 = ', x0, ';   |y(x0)| < e: ', '%0.15f' %y)

#Метод простой итерации

def SimpleIterationMethod(a, b):

    print('Метод простой итерации:')

    x = Symbol('x')

    k = 0

    while (k <= 1): # подборка коэффициента

        g = x - k \* function(x) # сжимающее уравнение

        dg = diff(g)

        if (dg.subs(x, a) < 1) & (dg.subs(x, b) < 1) : # удовлетворение условию g'(x) < 1 [a, b]

            print('Коффициент подобран: k = ', k)

            break

        k+= 0.1

    x1 = (a + b) / 2

    while True:

        x0 = x1

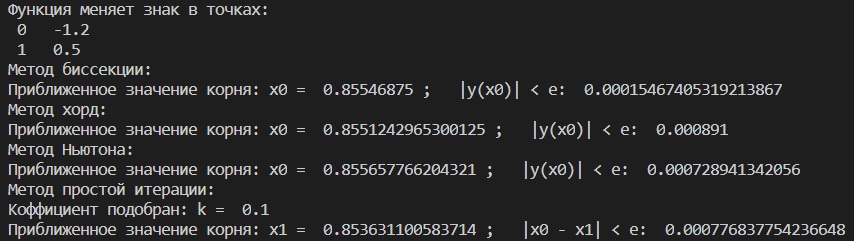
        x1 = g.subs(x, x0) # подстановка в сжимающее уравнение x0

        if abs(x0 - x1) < eps: # условие выхода из цикла |x0 - x1| < e

            break

    print('Приближенное значение корня: x1 = ', x1, ';   |x0 - x1| < e: ', abs(x0 - x1))

# Результат работы программы



# Вывод

Итерационные методы решения нелинейных уравнений удобно применять, когда удается выделить промежуток, на котором находится ровно один корень (значения функции на концах отрезка имеют разные знаки) и нас интересует его значение с некоторой заданной точностью ε.

В ходе данной лабораторной работы были рассмотрены четыре итерационных метода решений уравнений: метод биссекции, метод хорд, метод Ньютона и метод простых итераций.

Самым простым с точки зрения вычислений является метод биссекции, но приближение в данном метода происходит медленнее других способов.

Метод хорд работает быстрее метода биссекции, но его алгоритм осложняется условием выбора неподвижного конца.

Самым простым с точки зрения реализации алгоритма является метод Ньютона (метод касательных), и он же является самым быстрым.

Сложнейшим является метод простых итераций, т.к. для его работы необходимо вывести доп. уравнение 𝑥= 𝜑(𝑥). От того насколько “качественным” окажется выведенное уравнение зависит точность полученного результат и кол-во итераций необходимых для его получения.