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

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

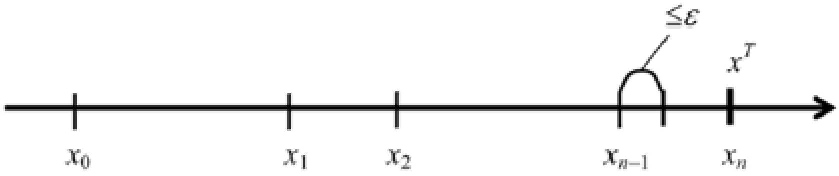
Вариант № 7:

$$x^3 + 0.2x^2 + 0.5x - 1.2 = 0$$

Теоретические сведения

Критерий остановки

Процесс нахождения оптимального решения чаще всего имеет итерационный характер, т.е. последовательность $\{x0,\ x1,\ \dots,\ xn\ \to\ x\}$ 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.$|f(xn)|<\varepsilon$ – найдена заданная точность функции

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

Метод биссекции

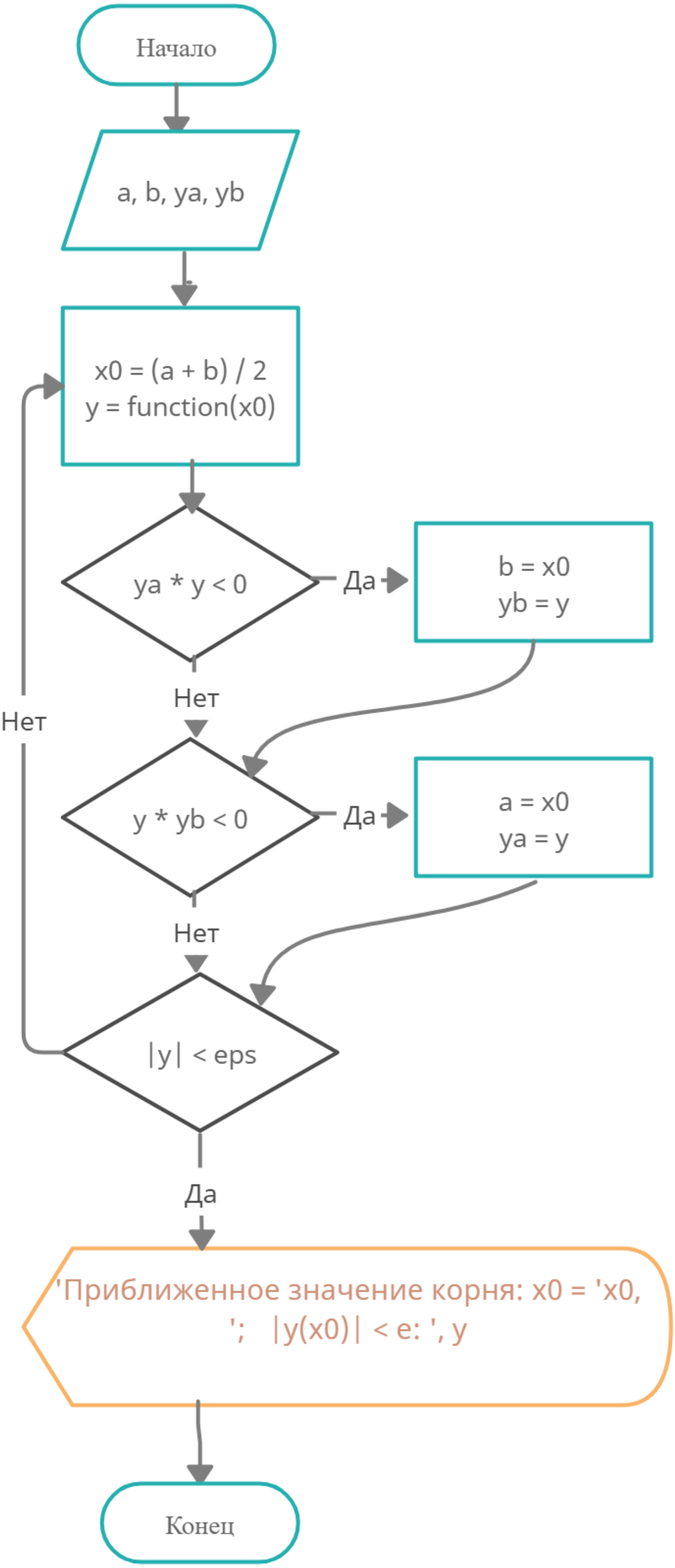
Пусть был выбран интервал изоляции $[a;b]$. Примем за первое приближение корня точку с, которая является серединой отрезка $[a;b]$. Далее будем действовать по следующему алгоритму:

1. Находим точку $x_i = \frac{a_i+b_i}{2}$;

2. Находим значение $f(x_i)$;

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

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.


Начало
a, b, ya, yb
x0 = (a + b) / 2
y = function(x0)
ya * y < 0
Да
b = x0
yb = y
Нет
y * yb < 0
Да
a = x0
ya = y
Нет
Нет
|y| < eps
Да
'Приближенное значение корня: x0 = 'x0, '; |y(x0)| < e: ', y
Конец

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 $(b, f(b))$:

$$\frac{y - f(a)}{f(b) - f(a)} = \frac{x - a}{b - a}$$

$$y = \frac{f(b) - f(a)}{b - a}(x - a) + f(a)$$

Прямая, заданная уравнением, пересекает ось X при условии y = 0. Найдём точку пересечения хорды с осью X:

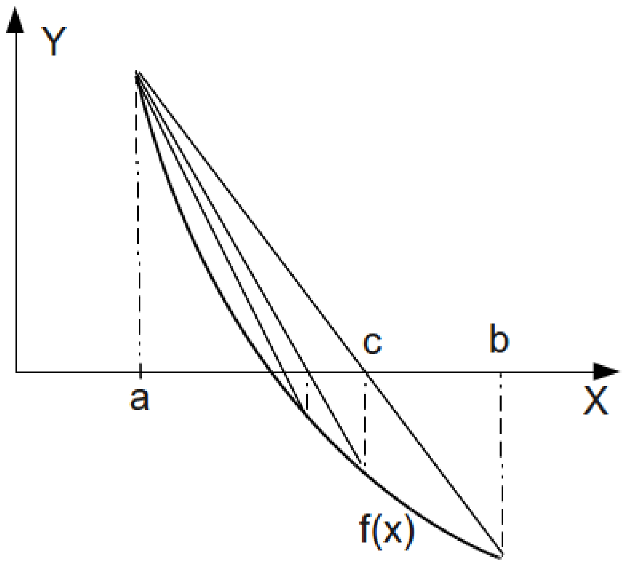
$$y = \frac{f(b) - f(a)}{b - a}(x - a) + f(a)$$

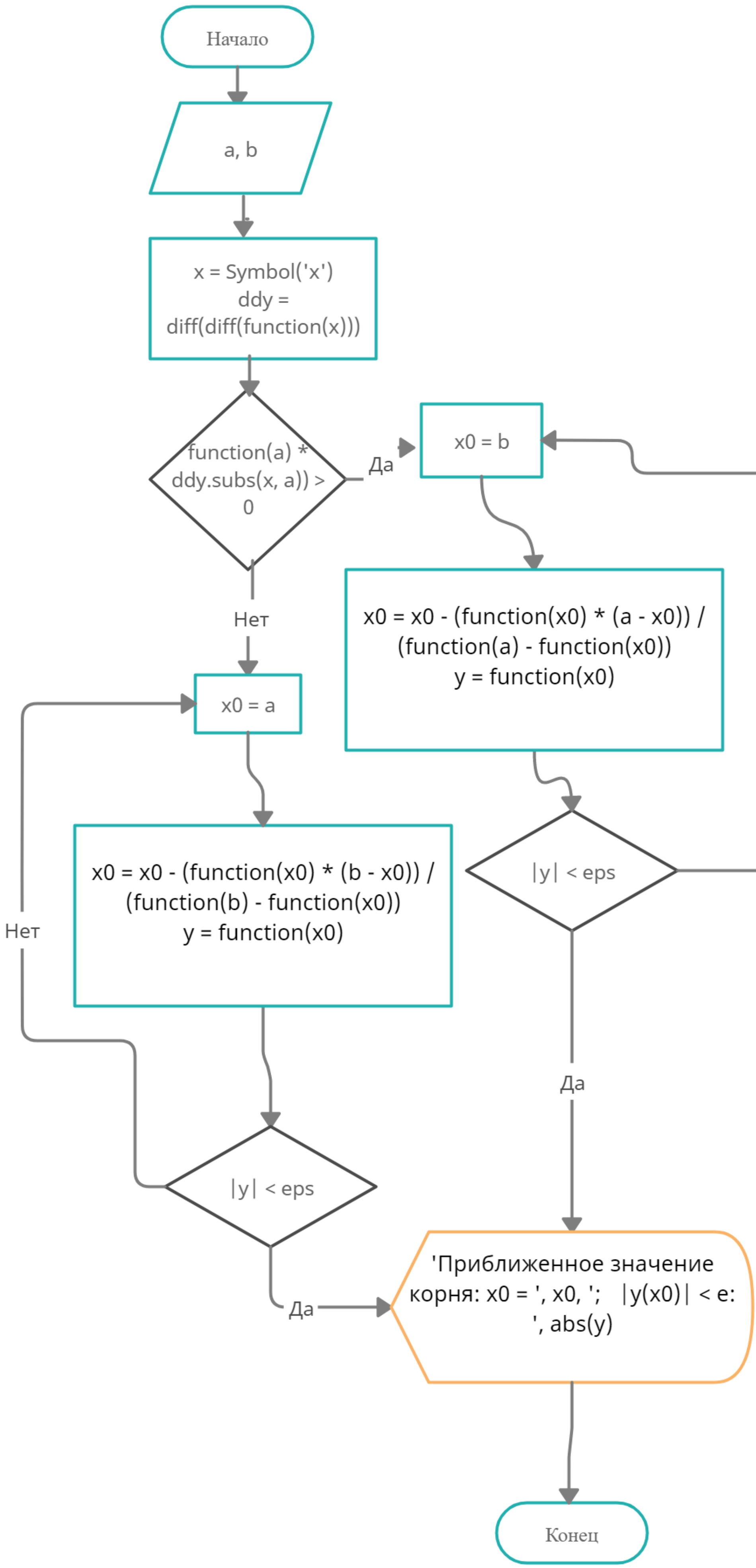
$$x = a - \frac{f(a)(b - a)}{f(b) - f(a)}$$

$$x_i = a - \frac{f(a)}{f(b) - f(a)}(b - a)$$

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.




Начало
a, b
x = Symbol('x')
ddy =
diff(diff(function(x)))
function(a) *
ddy.subs(x, a)) >
0
Да
x0 = b
Нет
x0 = x0 - (function(x0) * (a - x0)) /
(function(a) - function(x0))
y = function(x0)
Нет
x0 = a
x0 = x0 - (function(x0) * (b - x0)) /
(function(b) - function(x0))
y = function(x0)
Нет
|y| < eps
|y| < eps
Да
Да
'Приближенное значение
корня: x0 = ', x0, '; |y(x0)| < e:
', abs(y)
Конец

Метод Ньютона

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

$$y = kx + m$$

Так как эта прямая является касательной, и она проходит через точку (x_i, $f(x_i)$), то $k = f'(x_i)$.

Следовательно,

$$y = f'(x)x + m$$

$$f(x_i) = f'(x_i)x_i + m$$

$$m = f(x_i) - x_i f'(x_i)$$

$$y = f'(x_i)x + f(x_i) - x_i f'(x_i)$$

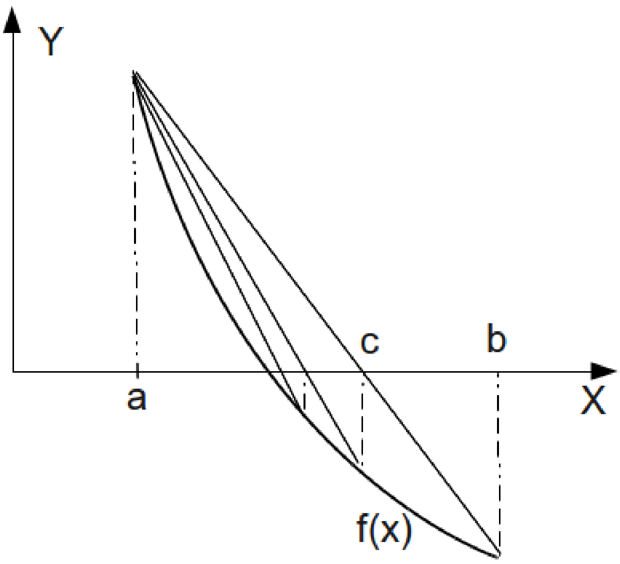
$$y = f'(x_i)(x - x_i) + f(x_i)$$

Найдём точку пересечения касательной с осью X:

$$f'(x)(x - x_i) + f(x_i) = 0$$

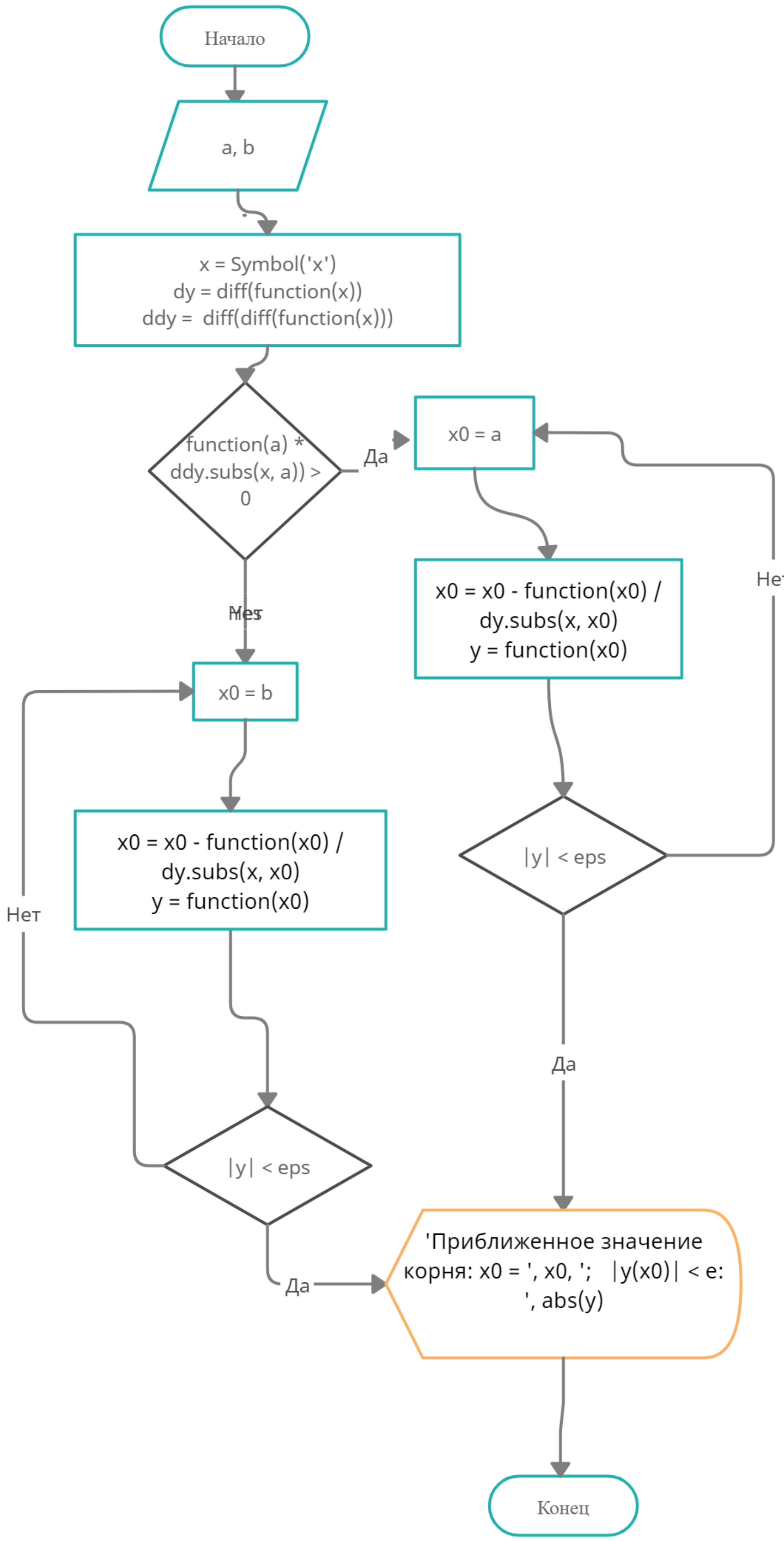
$$x = x_i - \frac{f(x_i)}{f'(x_i)}$$

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



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.



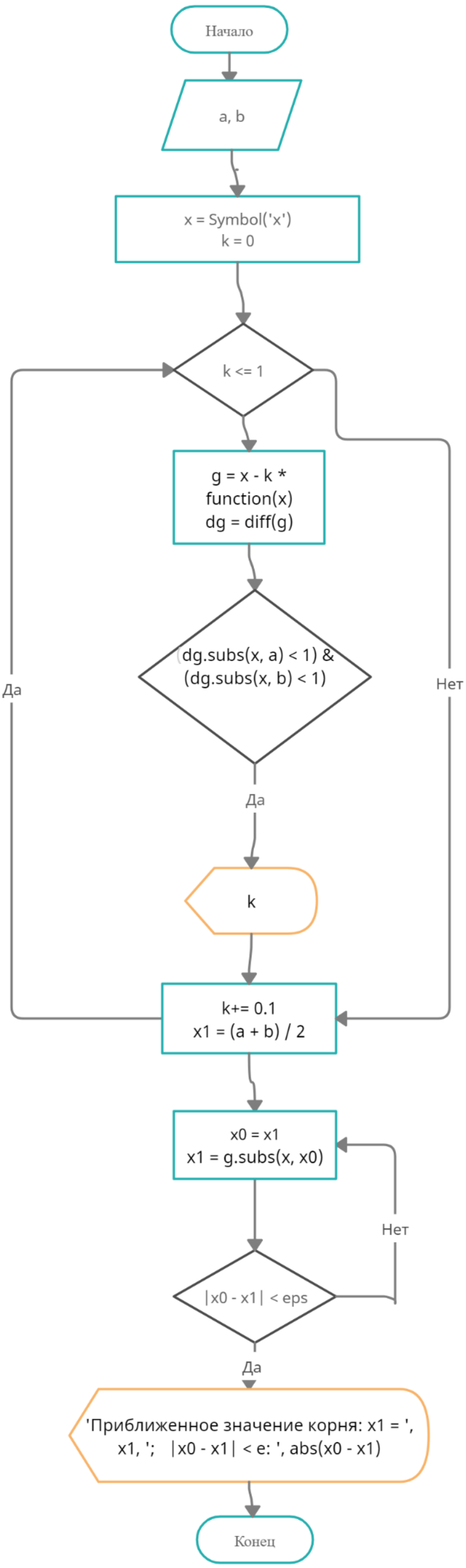
Начало
a, b
x = Symbol('x')
dy = diff(function(x))
ddy = diff(diff(function(x)))
function(a) * ddy.subs(x, a)) > 0
Да
Нет
x0 = a
x0 = x0 - function(x0) / dy.subs(x, x0)
y = function(x0)
Нет
x0 = b
x0 = x0 - function(x0) / dy.subs(x, x0)
y = function(x0)
Нет
|y| < eps
|y| < eps
Да
Да
'Приближенное значение корня: x0 = ', x0, '; |y(x0)| < e: ', abs(y)
Конец

Метод простых итераций

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

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

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



Начало
a, b
x = Symbol('x')
k = 0
k <= 1
g = x - k *
function(x)
dg = diff(g)
(dg.subs(x, a) < 1) &
(dg.subs(x, b) < 1)
Да
Да
Нет
k
k+= 0.1
x1 = (a + b) / 2
x0 = x1
x1 = g.subs(x, x0)
Нет
|x0 - x1| < eps
Да
'Приближенное значение корня: x1 = ',
x1, '; |x0 - x1| < e: ', abs(x0 - x1)
Конец

Расчетные данные

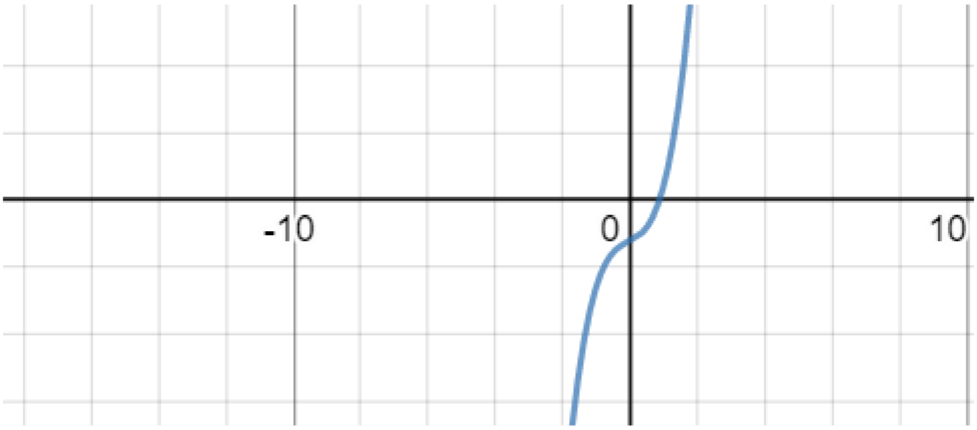
Исходное уравнение:

$$x^3 + 0.2x^2 + 0.5x - 1.2 = 0$$

Сжимающее уравнение:

$$x = x - 0.1(x^3 + 0.2x^2 + 0.5x - 1.2)$$

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 приближ
ение 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 приближ
ение 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

```
Функция меняет знак в точках:
 0   -1.2
 1   0.5
Метод биссекции:
Приближенное значение корня: x0 =  0.85546875 ;   |y(x0)| < e:  0.00015467405319213867
Метод хорд:
Приближенное значение корня: x0 =  0.8551242965300125 ;   |y(x0)| < e:  0.000891
Метод Ньютона:
Приближенное значение корня: x0 =  0.855657766204321 ;   |y(x0)| < e:  0.000728941342056
Метод простой итерации:
Коффициент подобран: k =  0.1
Приближенное значение корня: x1 =  0.853631100583714 ;   |x0 - x1| < e:  0.000776837754236648
```

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 $x= \varphi(x)$. От того насколько “качественным” окажется выведенное уравнение зависит точность полученного результат и кол-во итераций необходимых для его получения.