МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра вычислительные системы и технологии

Решение нелинейных уравнений

Отчет

по лабораторной работе №1

по дисциплине

Вычислительная математика

РУКОВОДИТЕЛЬ:

_____ Суркова А. С.

СТУДЕНТ:

_____ Соляник Д. Р.

19-ИВТ-2

Работа защищена «___» _____

С оценкой _____

Нижний Новгород 2020

1. Тема лабораторной работы:

Решение нелинейных уравнений

Цель лабораторной работы

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

2. Вариант задания на лабораторную работу

Вариант № 18

Решить нелинейное уравнение с одним неизвестным с использованием четырех методов (метод бисекций, метод хорд, метод Ньютона, метод простой итерации). ε=0.001

$$x^3 - 3x^2 + 12x - 12 = 0$$

3. Теоретические сведения и описание лабораторной работы

- *Метод бисекций*

Пусть необходимо решить уравнение f(x) = 0, где функция непрерывна на отрезке [a; b], и единственный корень x заключен в том же интервале:

1. Разделим отрезок [a; b] пополам и найдем x = (a + b) / 2.

2. Вычислим значение f(x) в этой точке.

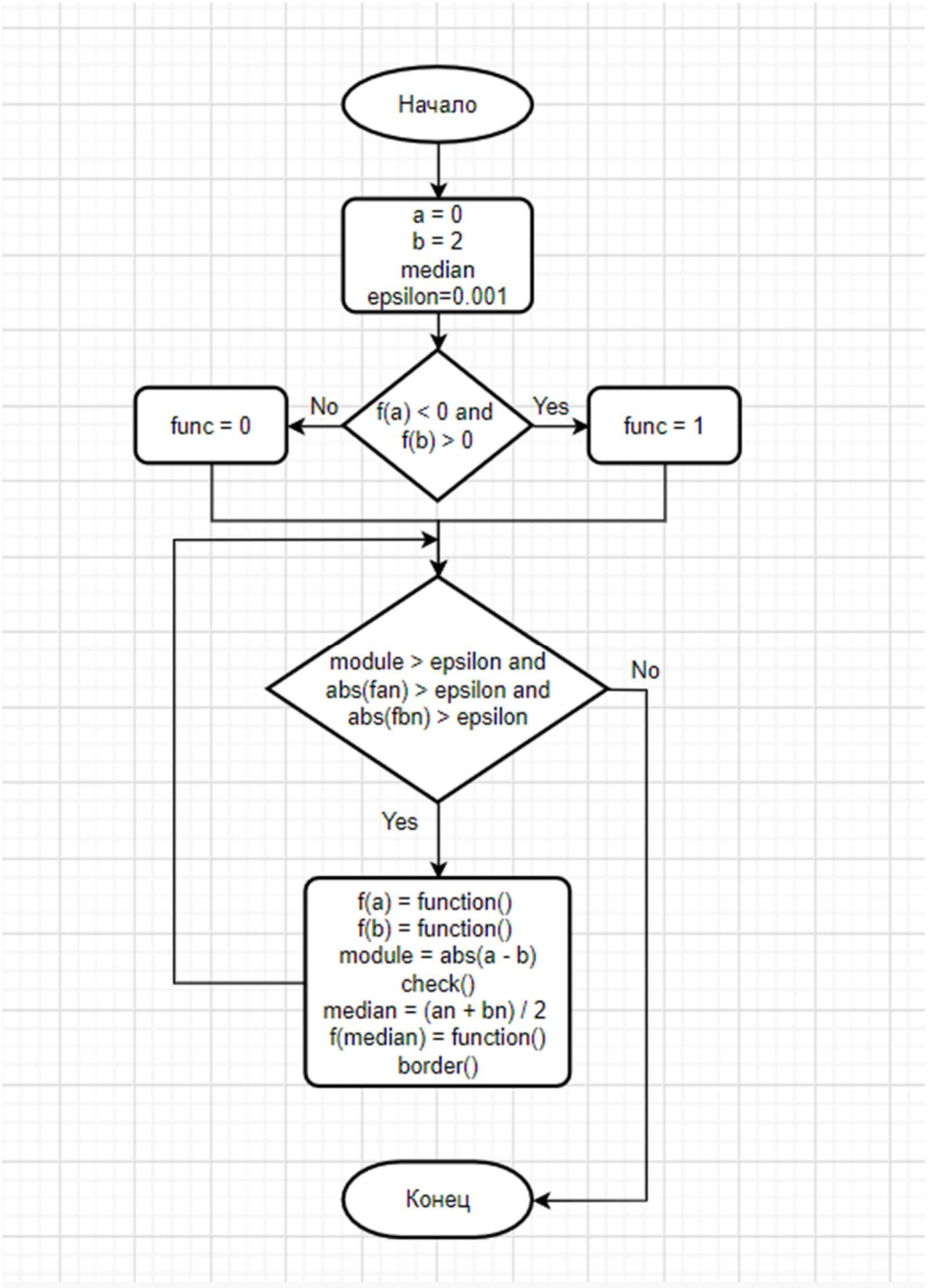
3. Проверим знак условия f(x)·f(a). Если f(x)·f(a) > 0, то корень находится на отрезке [x; b].

4. Половина отрезка [a; x] отбрасывается.

5. Левая граница интервала перемещается в точку x, a = x.

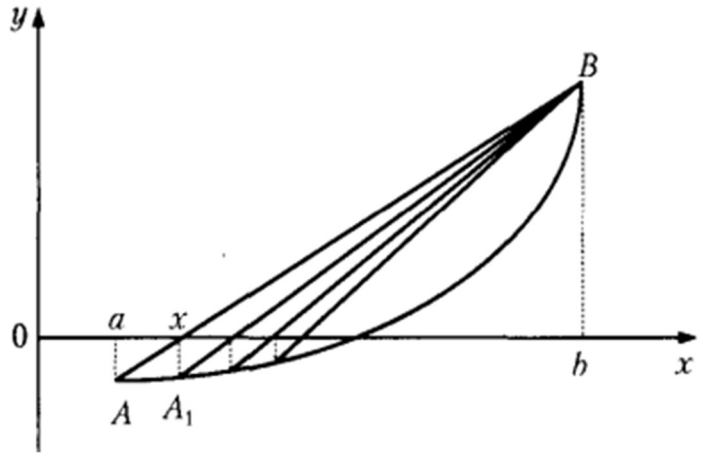
6. Если f(x)·f(a) < 0, то корень находится на отрезке [a; x] и отбрасывается [x; b].

При повторном делении интервала производятся те же самые операции: новый отрезок [a; b] делится пополам, вычисляется значение функции в точке деления f(x) и определяется отрезок, содержащий корень. Процесс деления продолжается до тех пор, пока длина отрезка [a; b], содержащего корень, не станет меньше некоторого малого заданного числа ε.


Начало
a = 0
b = 2
median
epsilon=0.001
f(a) < 0 and
f(b) > 0
No
Yes
func = 0
func = 1
module > epsilon and
abs(fan) > epsilon and
abs(fbn) > epsilon
No
Yes
f(a) = function()
f(b) = function()
module = abs(a - b)
check()
median = (an + bn) / 2
f(median) = function()
border()
Конец

- *Метод хорд*

Пусть уравнение f(x) = 0 имеет один корень на отрезке [a; b], а первая и вторая производные функции f(x) определены, непрерывны и сохраняют постоянные знаки на этом интервале.

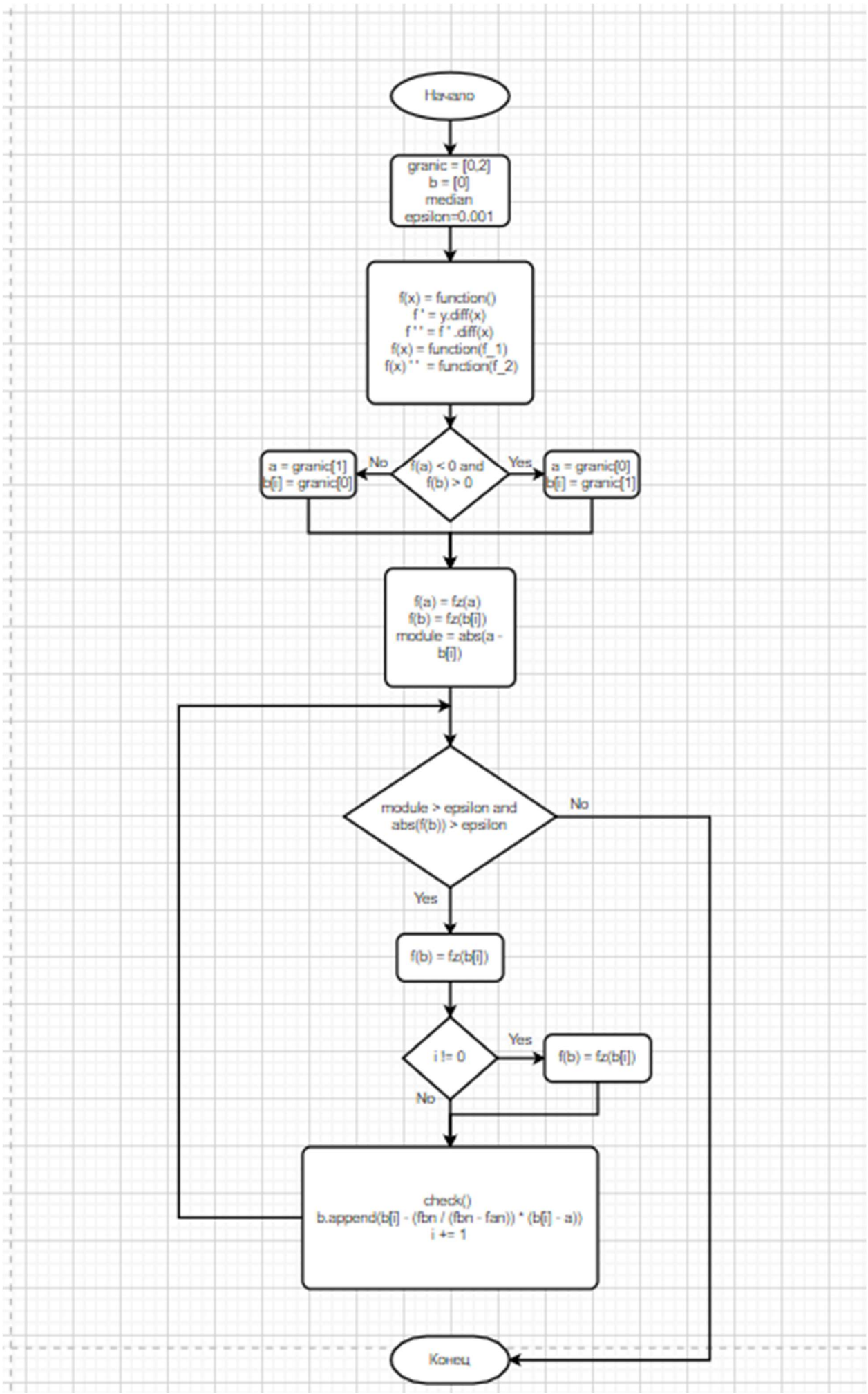


Проводим хорду через точки A и B. В точке пересечения хорды с осью x находим значение функции f(x), получаем точку A1, затем проводим новую хорду через точки A1 и B и т. д.

Уравнение прямой, проходящей через две точки (x1, y1) и (x2, y2), имеет вид:

$$(x - x1)/(x1 - x2) = (y - y1)/(y1 - y2) \qquad (1)$$

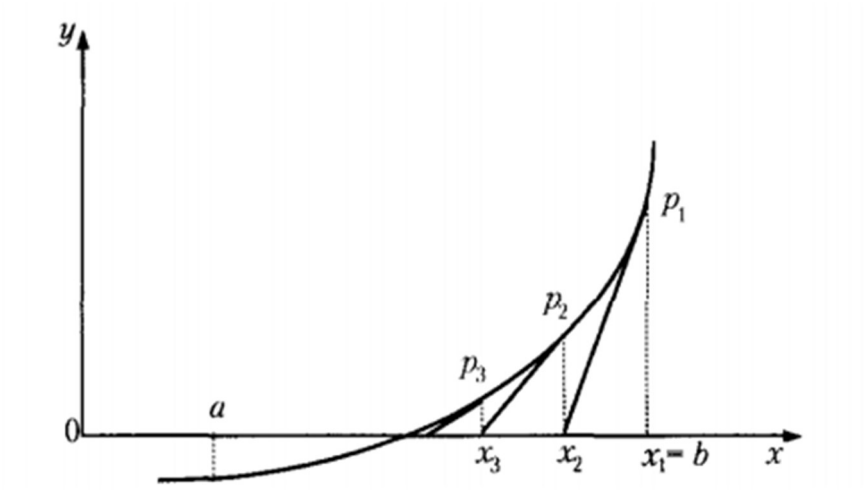
Из этого уравнения надо определить точку x – точку пересечения хорды с осью x. Значит, в уравнении (1) y = 0. Один из концов отрезка [a; b] является подвижным (в нашем случае a), другой конец – неподвижным. В качестве подвижного конца выбирается точка, для которой выполняется условие $f(x) \cdot f''(x) < 0$.


Начало
granic = [0,2]
b = [0]
median
epsilon=0.001
f(x) = function()
f ' = y.diff(x)
f ' ' = f ' .diff(x)
f(x) = function(f_1)
f(x) ' ' = function(f_2)
f(a) < 0 and f(b) > 0
No
Yes
a = granic[1]
b[i] = granic[0]
a = granic[0]
b[i] = granic[1]
f(a) = fz(a)
f(b) = fz(b[i])
module = abs(a - b[i])
module > epsilon and abs(f(b)) > epsilon
No
Yes
f(b) = fz(b[i])
i != 0
Yes
f(b) = fz(b[i])
No
check()
b.append(b[i] - (fbn / (fbn - fan)) * (b[i] - a))
i += 1
Конец

- *Метод Ньютона*

Пусть уравнение f(x) = 0 имеет один корень на отрезке [a; b], а первая и вторая производные функции f(x) определены, непрерывны и сохраняют постоянные знаки на этом интервале.

Выберем в качестве первого приближения точку x1 = b. Через точку p1 с координатами [x1; f(x1)] проведем касательную к кривой y = f(x). В качестве второго приближения x2 возьмем абсциссу точки пересечения этой касательной с осью x. Через точку p2 снова проведем касательную, пересечение которой с абсциссой дает следующее приближение x3 и т. д.



Уравнение касательной, проходящей через точку x1, имеет вид

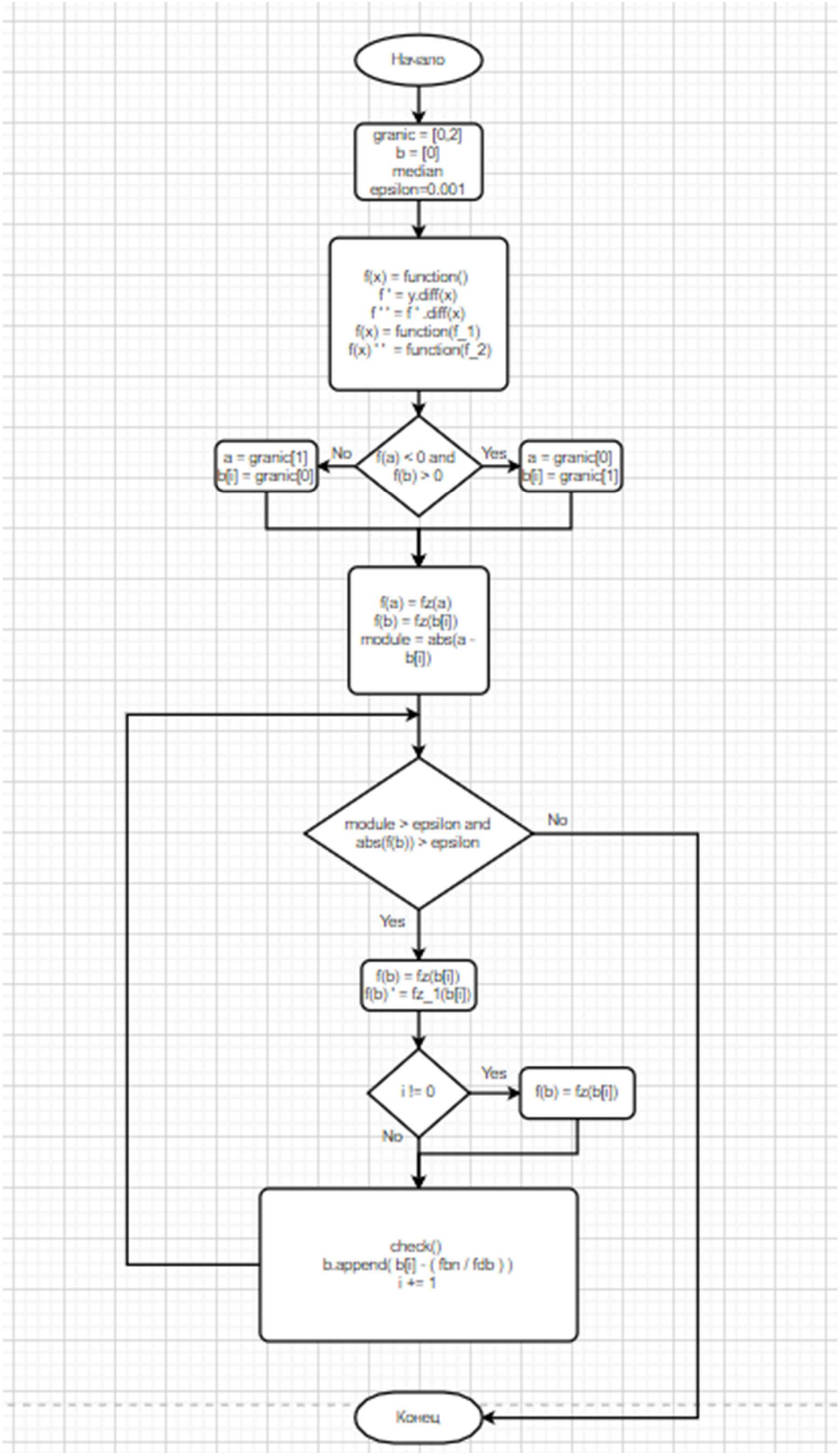
y – f(x1) = f '(x1)·(x – x1),

уравнение касательной, проходящей через точку x2, x2 = x1 – f(x1)/f '(x1).

В общем случае можно записать рекуррентную формулу:

xn+1 = xn – f(xn)/f '(xn).

Если в качестве первого приближения взять x1 = a и провести касательную к функции f(x) в точке pa, то она пересечет ось x вне отрезка [a; b]. В качестве первого приближения выбирается тот конец интервала [a; b], для которого выполняется условие f(x)·f "(x) > 0, где f "(x) — вторая производная. Это условие сходимости метода является достаточным, но не необходимым: если условие выполняется, то итерационный процесс обязательно сойдется, а если не выполняется, то может сойтись, а может и не сойтись


Начало
granic = [0,2]
b = [0]
median
epsilon=0.001
f(x) = function()
f ' = y.diff(x)
f '' = f ' .diff(x)
f(x) = function(f_1)
f(x) '' = function(f_2)
f(a) < 0 and f(b) > 0
No
Yes
a = granic[1]
b[i] = granic[0]
a = granic[0]
b[i] = granic[1]
f(a) = fz(a)
f(b) = fz(b[i])
module = abs(a - b[i])
module > epsilon and abs(f(b)) > epsilon
No
Yes
f(b) = fz(b[i])
f(b) ' = fz_1(b[i])
i != 0
Yes
No
f(b) = fz(b[i])
check()
b.append(b[i] - (fbn / fdb))
i += 1
Конец

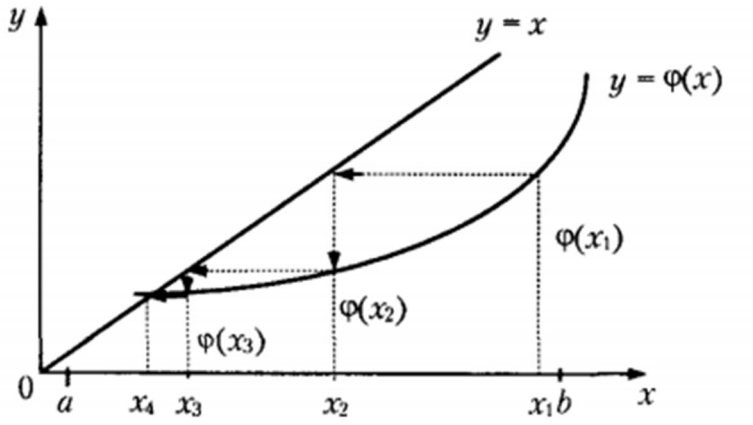
- *Метод простой итерации*

Использование метода предполагает представление уравнения f(x) = 0 в виде x = φ(x). Выберем на отрезке [a; b] первое приближение x1 и подставим его в правую часть уравнения, затем x2 = φ(x1).

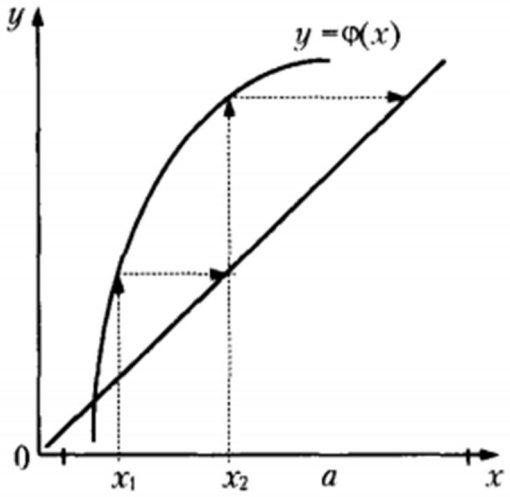
В общем виде можно записать:

xn+1 = φ(xn).

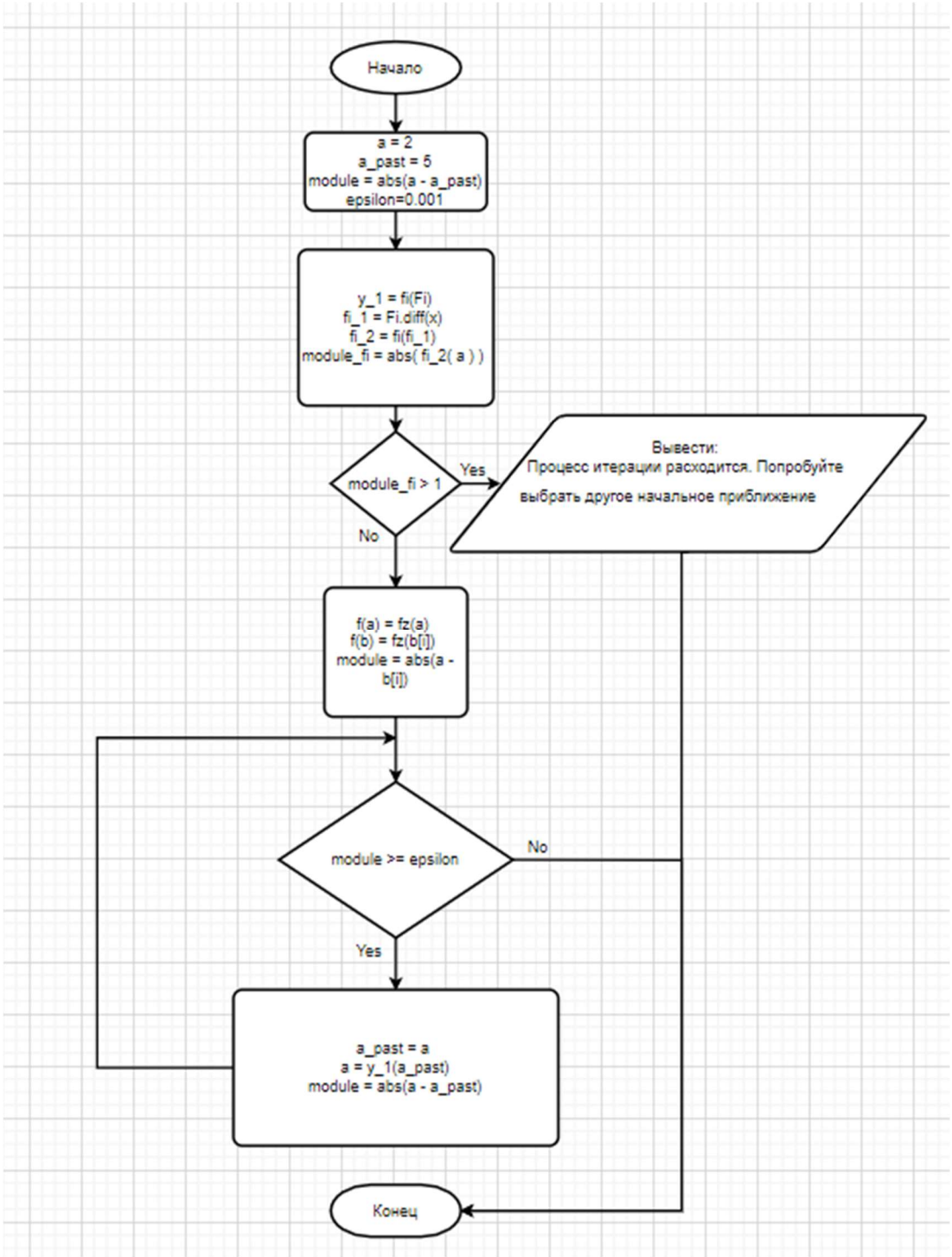
Геометрически способ простых итераций можно представить следующим образом. Построим графики двух функций: y = x и y = φ(x). Абсцисса точек пересечения этих двух графиков является корнем уравнения f(x) = 0.



Процесс итераций сходится при условии |φ'(x)| < 1. При |φ'(x)| > 1 процесс может расходиться Для выбора начального приближения вычисляют значения первых производных функции φ(x) в граничных точках интервала [a; b], содержащего корень, и за начальное приближение принимают тот конец интервала, для которого выполняется условие |φ'(x)| < 1.



Если обе производные по абсолютной величине меньше единицы, то за начальное приближение берут ту границу, для которой значение производной по абсолютной величине меньше, тогда процесс итераций сходится быстрее.


Начало
a = 2
a_past = 5
module = abs(a - a_past)
epsilon=0.001
y_1 = fi(Fi)
fi_1 = Fi.diff(x)
fi_2 = fi(fi_1)
module_fi = abs(fi_2(a))
module_fi > 1
Yes
No
Вывести:
Процесс итерации расходится. Попробуйте выбрать другое начальное приближение
f(a) = fz(a)
f(b) = fz(b[i])
module = abs(a - b[i])
module >= epsilon
No
Yes
a_past = a
a = y_1(a_past)
module = abs(a - a_past)
Конец

4. Расчетные данные

- *Метод бисекций*

A	B	mediana	f(a)	f(c)	f(a)*f(c)	|An - Bn|	
0	2	1	-12	-2	24	2	
1	2	1,5	-2	2,625	-5,25	1	
1	1,5	1,25	-2	0,265625	-0,53125	0,5	
1	1,25	1,125	-2	-0,87305	1,746094	0,25	
1,125	1,25	1,1875	-0,87305	-0,30591	0,267072	0,125	
1,1875	1,25	1,21875	-0,30591	-0,02078	0,006358	0,0625	
1,21875	1,25	1,234375	-0,02078	0,12225	-0,00254	0,03125	
1,21875	1,234375	1,226563	-0,02078	0,050692	-0,00105	0,015625	
1,21875	1,226563	1,222656	-0,02078	0,014945	-0,00031	0,007813	
1,21875	1,222656	1,220703	-0,02078	-0,00292	6,07E-05	0,003906	
1,220703	1,222656	1,22168	-0,00292	0,006011	-1,8E-05	0,001953	
1,220703	1,22168	1,221191	-0,00292	0,001545	-4,5E-06	0,000977	< 0.001

- *Метод хорд*

A	B	f(a)	f(b)	x	|xi+1 -xi|		
0	2	-12	8	1,2			
0	1,2	-12	-0,192	1,219512	0,8		
0	1,219512	-12	-0,01381	1,220918	0,019512		
0	1,220918	-12	-0,00096	1,221015	0,001405	-0,00096	< 0.001

- *Метод Ньютона*

B	f(b)	f(b)'	x	|xi -xi+1|		
2	8	12	1,333333			
1,333333	1,037037	9,333333	1,222222	0,666667		
1,222222	0,010974	9,148148	1,221023	0,111111		
1,221023	9,58E-07	9,146553	1,221023	0,0012	9,58E-07	< 0.001

- *Метод простой итерации*

B	Fi(b)	|B-Fi(x)|	
2	1,333333	0,666667	
1,333333	1,246914	0,08642	
1,246914	1,227141	0,019773	
1,227141	1,222475	0,004665	
1,222475	1,221368	0,001107	
1,221368	1,221105	0,000263	< 0.001

5. Листинг разработанной программы

- *Метод бисекций*

```
an = 0 #Левая граница (будем считать известной)
bn = 2 #Правая граница (будем считать известной)
epsilon = 0.001

def function(x):
    "========Функция расчета f(x) = x^3 - 3x^2 + 12x - 12 = 0 ====================="
    f = x ** 3 - 3 * x ** 2 + 12 * x - 12
    return f

def border(fab, func):
    "========Функция замены границы на посчитанную
медиану========================="
    global bn, an
    if fab > 0 and func == 1:
        bn = median
        return bn
    if fab < 0 and func == 1:
        an = median
        return an
    if fab > 0 and func == 0:
        an = median
        return an
    if fab < 0 and func == 0:
        bn = median
        return bn

def check(module, epsilon, fab, fbn):
    "========Функция проверки конца
алгоритма=================================="
    if module < epsilon :

    print("================================================================
=======")
```

```
                print("При границах: a = ", "%.5f" % an, "и b = ", "%.5f" % bn, ", |an - bn| -", "%.5f" % module, "< 0.001")

        print("================================================================================")

                exit()


        if abs(fan) < epsilon :

        print("================================================================================")

                print("При границах: a = ", "%.5f" % an, "и b = ", "%.5f" % bn, ", |f(a)| -", "%.5f" % abs(fan), "< 0.001")

        print("================================================================================")

                exit()


        if abs(fbn) < epsilon :

        print("================================================================================")

                print("При границах: a = ", "%.5f" % an, "и b = ", "%.5f" % bn, ", |f(b)| -", "%.5f" % abs(fbn), "< 0.001")

        print("================================================================================")

                exit()


fan = function(an)

fbn = function(bn)

module = abs(an - bn)


#========Проверка фунции f(x) на возрастанеи(убывание) ======================

if (fan < 0) and (fbn > 0):

        func = 1

else:

        func = 0


#========Шапка таблицы с полученными данными=================================
```

```
print("An          |Bn          |f(An)        |f(Bn)        ||an - bn|")

while (module > epsilon) and (abs(fan) > epsilon) and (abs(fbn) > epsilon):
        fan = function(an)
        fbn = function(bn)
        module = abs(an - bn)
#========Таблица с полученными данным=======================================
        print("%.5f" % an, "     ", "%.5f" % bn, "     ", "%.5f" % fan,"     ","%.5f" % fbn,"     ", "%.5f" % module)
        check(module,epsilon, fan, fbn)
        median = (an + bn) / 2 # Считаем медиану
        fab = function(median)
        border(fab, func)
```

- *Метод хорд*

```
from sympy import *
import numpy as np

granic = [0,2] #Левая и правая границы (будем считать известными)
epsilon = 0.001

i = 0 # Счетчик для значений подвижного конца отрезка
b = [0] # Значения подвижного конца отрезка

x = symbols('x')
y = x ** 3 - 3 * x ** 2 + 12 * x - 12

def function(z):
        "========Функция расчета f(x), f(x)' и f(x)" ================================="
        f = lambdify(x, z, 'numpy')
        return f

def check(module, epsilon, fbn):
        "========Функция проверки конца алгоритма=========================================="
        if (module < epsilon) :
```

```
                print("=====================================================")
                print("Искомое решение = ", "%.5f" % b[i], ", |bn i+1 - bn i| -", "%.5f" % module, "< 0.001")
                print("=====================================================")
                exit()


        if (abs(fbn) < epsilon) :
                print("=====================================================")
                print("Искомое решение = ", "%.5f" % b[i], ", |f(b)| ", "%.5f" % abs(fbn), "< 0.001")
                print("=====================================================")
                exit()


fz = function(y) # Воспользоваться функцией при f(x) = x^3 - 3*x^2 + 12*x - 12

f_1 = y.diff(x) # Первая производаня

f_2 = f_1.diff(x) #Вторая производная

fz_1 = function(f_1) # Воспользоваться функцией для f(x)'

fz_2 = function(f_2) # Воспользоваться функцией для f(x)"


# Определим неподвижный конец отрезка и обозначим его за "a", иной конец отрезка обозначим за "b"

# Для проверки воспользуемся формулой f(x) * f(x)" > 0 , тогда конец можно считать неподвижным

if ( fz(granic[0]) * fz_2(granic[0]) ) > 0 :

        print("Поскольку произведение f(" , granic[0] , ")", fz(granic[0]), "и f'(" , granic[0] , ") " , fz_2(granic[0]), " > 0 , то a неподвижный конец отрезка" )

        a = granic[0]

        b[i] = granic[1]

else:

        if (fz(granic[1]) * fz_2(granic[1])) > 0:

                print("Поскольку  произведение f(" , granic[1] , ")", fz(granic[1]), "и f'(" , granic[1] , ") " , fz_2(granic[1]), " > 0 , то b неподвижный конец отрезка" )

                a = granic[1]

                b[i] = granic[0]

        else:

                print("Не удалось найти неподвижный конец отрезка")

                exit()
```

```
fan = fz(a) # Рассчитаем значение функции f(x) для неподвижного конца отрезка
fbn = fz(b[i]) # Рассчитаем значение функции f(x) для конца отрезка, который будем сдвигать
module = abs(a - b[i])
print("An          |Bn          |f(An)        |f(Bn)        ||bn i+1 - bn i|")
while (module > epsilon) and (abs(fbn) > epsilon):
        fbn = fz(b[i]) # Рассчитаем значение функции f(x) для конца отрезка, который будем сдвигать
        if i != 0 :
                module = abs(b[i-1] - b[i])
        print("%.5f" % a, "     ", "%.5f" % b[i], "     ", "%.5f" % fan,"     ","%.5f" % fbn,"     ", "%.5f" % module)
        check(module, epsilon, fbn)
        # Рассчитаем точку пересечения хорды с осью X
        b.append(b[i] - (fbn / (fbn - fan)) * (b[i] - a))
        i += 1
```

- *Метод Ньютона*

```
from sympy import *
import numpy as np

granic = [0,2] #Левая и правая границы (будем считать известными)
epsilon = 0.001

i = 0 # Счетчик для значений подвижного конца отрезка
b = [0] # Значения подвижного конца отрезка

x = symbols('x')
y = x ** 3 - 3 * x ** 2 + 12 * x - 12

def function(z):
        "========Функция расчета f(x), f(x)' и f(x)" ================================"
        f = lambdify(x, z, 'numpy')
        return f

def check(module, epsilon, fbn):
```

```
        "========Функция проверки конца
алгоритма====================================="

        if module < epsilon:

        print("=================================================================
====================")

                print("Искомое решение = ", "%.5f" % b[i], ", точность -", "%.5f" % module, "<
0.001")

        print("=================================================================
====================")

                exit()


        if (abs(fbn) < epsilon) :

                print("=====================================================")

                print("Искомое решение = ", "%.5f" % b[i], ", |f(b)| ", "%.5f" % abs(fbn), "< 0.001")

                print("=====================================================")

                exit()


fz = function(y) # Воспользоваться функцией при f(x) = x^3 - 3*x^2 + 12*x - 12

f_1 = y.diff(x) # Первая производаня

f_2 = f_1.diff(x) # Вторая производная

fz_1 = function(f_1) # Воспользоваться функцией для f(x)'

fz_2 = function(f_2) # Воспользоваться функцией для f(x)"


# Определим неподвижный конец отрезка и обозначим его за "a", иной конец отрезка обозначим за
"b"

# Для проверки воспользуемся формулой f(x) * f(x)" > 0 , тогда конец можно считать
неподвижным

if ( fz(granic[0]) * fz_2(granic[0]) ) > 0 :

        print("Поскольку произведение f(" , granic[0] , ")", fz(granic[0]), "и f'(" , granic[0] , ") " ,
fz_2(granic[0]), " > 0 , то a неподвижный конец отрезка" )

        a = granic[0]

        b[i] = granic[1]

else:

        if (fz(granic[1]) * fz_2(granic[1])) > 0:

                print("Поскольку  произведение f(" , granic[1] , ")", fz(granic[1]), "и f'(" , granic[1] , ")
" , fz_2(granic[1]), " > 0 , то b неподвижный конец отрезка" )

                a = granic[1]
```

```
            b[i] = granic[0]
        else:
            print("Не удалось найти неподвижный конец отрезка")
            exit()

fan = fz(a) # Рассчитаем значение функции f(x) для неподвижного конца отрезка
fbn = fz(b[i]) # Рассчитаем значение функции f(x) для конца отрезка, который будем сдвигать
module = abs(a - b[i])
print("An          |Bn          |f(An)        |f(Bn)        ||bn i+1 - bn i|")
while (module > epsilon) and (abs(fbn) > epsilon):
        fbn = fz(b[i]) # Рассчитаем значение функции f(x) для конца отрезка, который будем сдвигать
        fdb = fz_1(b[i]) # Рассчитаем значение функции f(x)' для конца отрезка, который будем сдвигать
        if i != 0 :
            module = abs(b[i-1] - b[i])
        print("%.5f" % a, "     ", "%.5f" % b[i], "     ", "%.5f" % fan,"     ","%.5f" % fbn,"     ", "%.5f" % module)
        check(module, epsilon, fbn)
        b.append( b[i] - ( fbn / fdb ) )
        i += 1
```

- *Метод простой итерации*

```
from sympy import *
import numpy as np

# Задаем начальное приближение
a = 2 # [-1 ; 3]
a_past = 5 # Переменная запоминает предыдущее значение a
epsilon = 0.001
module = abs(a - a_past)  # Модуль разности a и a_past
x = symbols('x')

Fi = ( x ** 3 - 3 * x ** 2 - 12 ) / -12 # x = Fi(x)
# Преобразовали f(x) = x ** 3 - 3 * x ** 2 + 12 * x - 12 в Fi(x) = ( x ** 3 - 3 * x ** 2 - 12 ) / -12
```

```
def fi(z):
        "========Функция расчета f(x), f(x)' и f(x)" ================================="
        f = lambdify(x, z, 'numpy')
        return f


y_1 = fi(Fi)
fi_1 = Fi.diff(x) # Взяли производную от Fi(x)
fi_2 = fi(fi_1) # Воспользоваться функцией для f(x)'
module_fi = abs( fi_2( a ) ) # Считаем модуль производной |Fi'(a)|


# Проверяем на сходимость итераций, для этого необходимо чтобы |Fi'(a)| < 1
if module_fi > 1 :
        print("================================================================================")
        print("Процесс итерации расходится. Попробуйте выбрать другое начальное приближение")
        print("================================================================================")
        exit()


print("A          |Fi(a)       ||a(i) - a(i-1)|")
while module >= epsilon :
        a_past = a # Присваиваем старое значение "a"
        a = y_1(a_past) # Высчитываем новое значение Fi(a)
        module = abs(a - a_past) # Считаем модуль разности |a(i) - a(i-1)|
        print("%.5f" % a_past, "    ", "%.5f" % a, "     ", "%.5f" % module)


print("================================================================================")
print("Приближенное значение корня равно =", "%.5f" % a, ", |a(i) - a(i-1)|", "%.5f" % module, "<=", epsilon)
print("================================================================================")
```

6. Результаты работы программы

- *Метод бисекций*

```
An          |Bn          |f(An)         |f(Bn)         ||an - bn|
0.00000      2.00000      -12.00000       8.00000       2.00000
1.00000      2.00000      -2.00000       8.00000       1.00000
1.00000      1.50000      -2.00000       2.62500       0.50000
1.00000      1.25000      -2.00000       0.26562       0.25000
1.12500      1.25000      -0.87305       0.26562       0.12500
1.18750      1.25000      -0.30591       0.26562       0.06250
1.21875      1.25000      -0.02078       0.26562       0.03125
1.21875      1.23438      -0.02078       0.12225       0.01562
1.21875      1.22656      -0.02078       0.05069       0.00781
1.21875      1.22266      -0.02078       0.01494       0.00391
1.22070      1.22266      -0.00292       0.01494       0.00195
1.22070      1.22168      -0.00292       0.00601       0.00098
========================================================================
При границах: a =  1.22070 и b =  1.22168 , |an - bn| - 0.00098 < 0.001
========================================================================
[Finished in 0.1s]
```

- *Метод хорд*

```
Поскольку произведение f( 0 ) -12 и f''( 0 )  -6  > 0 , то a неподвижный конец отрезка
An          |Bn          |f(An)         |f(Bn)         ||bn i+1 - bn i|
0.00000      2.00000      -12.00000       8.00000       2.00000
0.00000      1.20000      -12.00000      -0.19200       0.80000
0.00000      1.21951      -12.00000      -0.01381       0.01951
0.00000      1.22092      -12.00000      -0.00096       0.00141
=====================================================
Искомое решение =  1.22092 , |f(b)|  0.00096 < 0.001
=====================================================
[Finished in 0.8s]
```

- *Метод Ньютона*

```
Поскольку произведение f( 0 ) -12 и f''( 0 )  -6  > 0 , то a неподвижный конец отрезка
An          |Bn          |f(An)         |f(Bn)         ||bn i+1 - bn i|
0.00000      2.00000      -12.00000       8.00000       2.00000
0.00000      1.33333      -12.00000       1.03704       0.66667
0.00000      1.22222      -12.00000       0.01097       0.11111
0.00000      1.22102      -12.00000       0.00000       0.00120
=====================================================
Искомое решение =  1.22102 , |f(b)|  0.00000 < 0.001
=====================================================
[Finished in 0.8s]
```

- *Метод простой итерации*

```
A           |Fi(a)        ||a(i) - a(i-1)|
2.00000      1.33333      0.66667
1.33333      1.24691      0.08642
1.24691      1.22714      0.01977
1.22714      1.22248      0.00467
1.22248      1.22137      0.00111
1.22137      1.22110      0.00026
================================================================================
Приближенное значение корня равно = 1.22110 , |a(i) - a(i-1)| 0.00026 <= 0.001
================================================================================
[Finished in 0.8s]
```

7. Вывод

В ходе лабораторной работы были изучены и применены методы: бисекций, хорд, Ньютона и простой итерации при решении нелинейных уравнений.