Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

**­­­­­Лабораторная работа №3**

**по дисциплине «Вычислительная математика»**

**«Численное решение нелинейных уравнений и систем»**

Вариант №3

Группа: P3212

Выполнил: Балин А. А.

Проверила: Наумова Н. А.

# Цель работы

Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений, выполнить программную реализацию методов.

# Вычислительная реализация

## Решение нелинейного уравнения

Уравнение: = 0.

* *Отделить корни и интервалы изоляции корней графически*

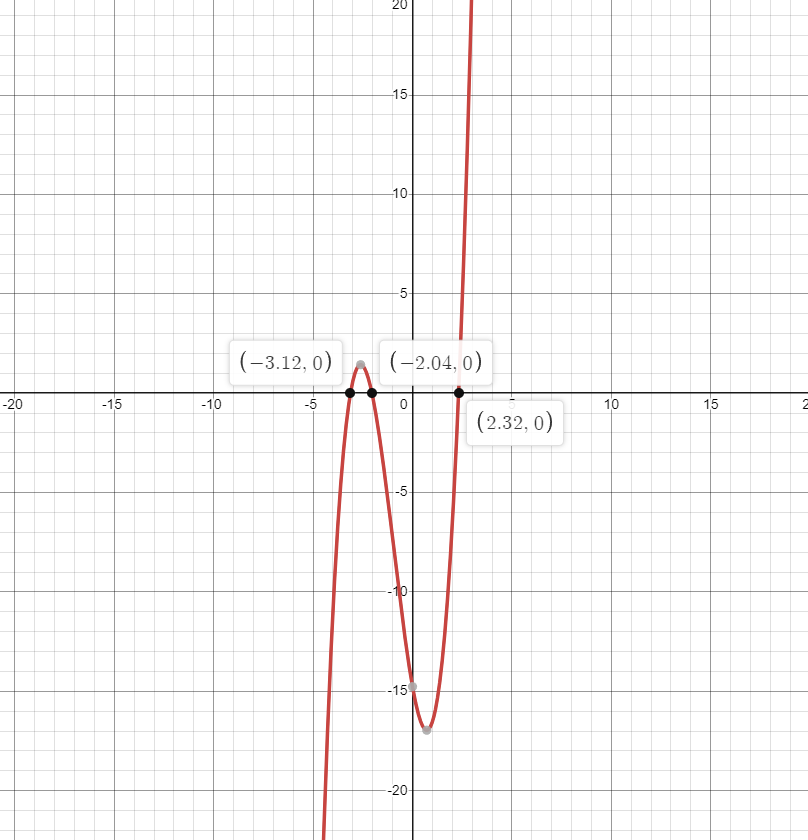


Рисунок 1. График исходной функции f(x).

Крайний левый корень уравнения находится в интервале .

Центральный в интервале .

Крайний правый в интервале

Точные значения корней указаны на рисунке.

* *Уточнение крайнего правого корня с помощью метода половинного деления*

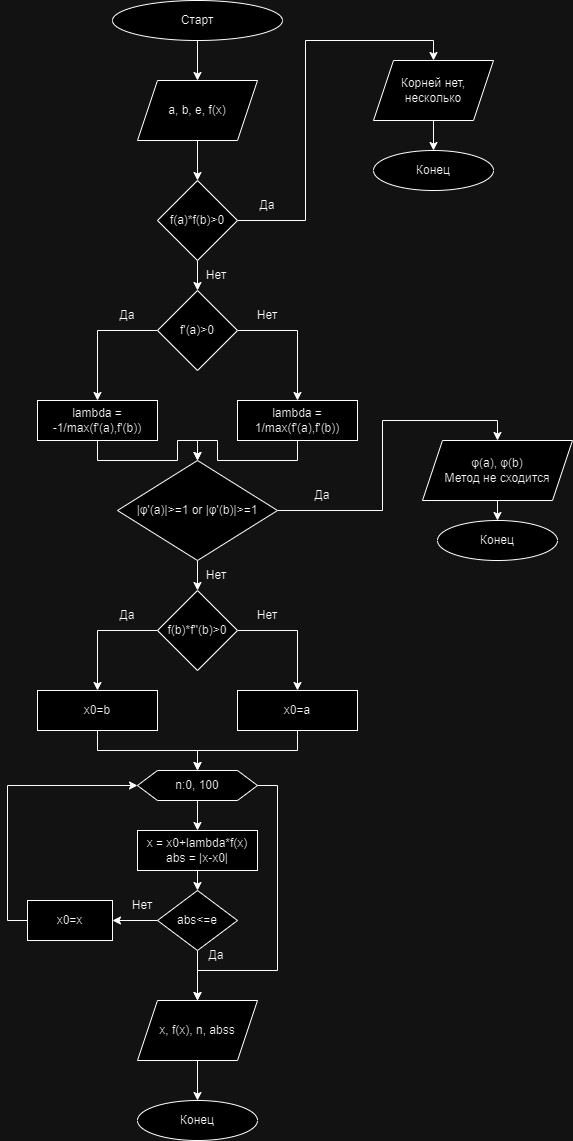


Схема 1. Блок-схема метода простых итераций.

Здесь и далее требуемая точность .

Вычислим теоретическое количество итераций для требуемой точности:

Вычисления:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** |  |  |  |  |  |  |  |  |
| **0** | **2,000** | **3,000** | **2,500** | **-6,618** | **20,976** | **4,594** | **1,000** | **НЕТ** |
| **1** | **2,000** | **2,500** | **2,250** | **-6,618** | **4,594** | **-1,611** | **0,500** | **НЕТ** |
| **2** | **2,250** | **2,500** | **2,375** | **-1,611** | **4,594** | **1,336** | **0,250** | **НЕТ** |
| **3** | **2,250** | **2,375** | **2,313** | **-1,611** | **1,336** | **-0,176** | **0,125** | **НЕТ** |
| **4** | **2,313** | **2,375** | **2,344** | **-0,176** | **1,336** | **0,570** | **0,063** | **НЕТ** |
| **5** | **2,313** | **2,344** | **2,328** | **-0,176** | **0,570** | **0,195** | **0,031** | **НЕТ** |
| **6** | **2,313** | **2,328** | **2,320** | **-0,176** | **0,195** | **0,009** | **0,016** | **НЕТ** |
| **7** | **2,313** | **2,320** | **2,316** | **-0,176** | **0,009** | **-0,084** | **0,008** | **ДА** |

Таблица 1. Уточнение правого корня.

* *Уточнение крайнего левого корня с помощью простого метода итерации*

Выразим из исходного уравнения и проверим достаточное условие сходимости:

На интервале функция , следовательно, достаточный признак не выполнен, поэтому не при всех начальных приближениях на итерационная последовательность будет сходиться к искомому . При подстановке начального приближения последовательность не сходилась при числе итераций , а при сошлась:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** |  |  |  |  |  |  |
| **0** | **-3,000** | **-3,136** | **-3,115** | **0,030** | **0,136** | **НЕТ** |
| **1** | **-3,136** | **-3,115** | **-3,121** | **-0,009** | **0,021** | **НЕТ** |
| **2** | **-3,115** | **-3,121** | **-3,119** | **0,003** | **0,007** | **ДА** |

Таблица 2. Уточнение левого корня.

Немного о способе преобразования уравнений:

* *Уточнение центрального корня методом Ньютона*

# C:\Users\artem\Downloads\Telegram Desktop\newton.png

Схема 2. Блок-схема метода Ньютона.

Нужно выбрать начальное приближение :

;

;

Следовательно, .

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** |  |  |  |  |  |  |
| **0** | **-2,000** | **-0,194** | **-4,966** | **-2,039** | **0,039** | **НЕТ** |
| **1** | **-2,039** | **-0,005** | **-4,715** | **-2,040** | **0,001** | **ДА** |

Таблица 3. Уточнение среднего корня.

## Решение системы нелинейных уравнений

Исходная система:

Решим её методом простой итерации.

* *Отделение корней графически*

Изображение выглядит как диаграмма, График, линия, текст

Автоматически созданное описание

Рисунок 2. График исходной системы уравнений.

Как видно на рисунке 2, решение уравнения ограничено:

*,*

*,*

поэтому выберем начальное приближение и .

* *Решение системы и проверка достаточного условия сходимости*

Выразим переменные:

Проверим достаточное условие сходимости:

;

;

На заданном множестве для и :

Следовательно, достаточное условие сходимости не выполняется и итерационный ряд может расходиться.

Вычисления:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** |  |  |  |  |  |  |  |
| **0** | **3,000** | **1,000** | **3,540** | **0,916** | **0,540** | **0,084** | **НЕТ** |
| **1** | **3,540** | **0,916** | **3,609** | **1,325** | **0,069** | **0,408** | **НЕТ** |
| **2** | **3,609** | **1,325** | **3,244** | **1,361** | **0,365** | **0,037** | **НЕТ** |
| **3** | **3,244** | **1,361** | **3,208** | **1,123** | **0,036** | **0,238** | **НЕТ** |
| **4** | **3,208** | **1,123** | **3,433** | **1,095** | **0,225** | **0,028** | **НЕТ** |
| **5** | **3,433** | **1,095** | **3,458** | **1,259** | **0,025** | **0,164** | **НЕТ** |
| **6** | **3,458** | **1,259** | **3,307** | **1,275** | **0,152** | **0,016** | **НЕТ** |
| **7** | **3,307** | **1,275** | **3,291** | **1,171** | **0,016** | **0,104** | **НЕТ** |
| **8** | **3,291** | **1,171** | **3,389** | **1,160** | **0,098** | **0,012** | **НЕТ** |
| **9** | **3,389** | **1,160** | **3,400** | **1,230** | **0,011** | **0,070** | **НЕТ** |
| **10** | **3,400** | **1,230** | **3,334** | **1,237** | **0,065** | **0,007** | **НЕТ** |
| **11** | **3,334** | **1,237** | **3,327** | **1,191** | **0,007** | **0,046** | **НЕТ** |
| **12** | **3,327** | **1,191** | **3,370** | **1,186** | **0,043** | **0,005** | **НЕТ** |
| **13** | **3,370** | **1,186** | **3,375** | **1,217** | **0,005** | **0,031** | **НЕТ** |
| **14** | **3,375** | **1,217** | **3,346** | **1,220** | **0,028** | **0,003** | **НЕТ** |
| **15** | **3,346** | **1,220** | **3,343** | **1,200** | **0,003** | **0,020** | **НЕТ** |
| **16** | **3,343** | **1,200** | **3,362** | **1,198** | **0,019** | **0,002** | **НЕТ** |
| **17** | **3,362** | **1,198** | **3,364** | **1,211** | **0,002** | **0,013** | **НЕТ** |
| **18** | **3,364** | **1,211** | **3,352** | **1,213** | **0,012** | **0,001** | **НЕТ** |
| **19** | **3,352** | **1,213** | **3,350** | **1,204** | **0,001** | **0,009** | **ДА** |

Таблица 4. Поиск решения системы методом простых итераций.

Нам повезло, и итерационная последовательность сошлась к решению с заданной точностью.

# Программная часть

## Реализация нелинейного уравнения

from math import exp, sin, cos,sqrt

from typing import List, Callable

**class** Equations:

**def** \_\_init\_\_(self):

        self.eq = [self.first, self.second, self.third, self.fourth]

        self.d = [self.f1, self.f2, self.f3, self.f4]

    @staticmethod

**def** first(x):

        return x\*\*2-3\*x+1

    @staticmethod

**def** second(x):

        return exp(x)-sin(x)+x

    @staticmethod

**def** third(x):

        return 1/sin(x)+x\*\*2

    @staticmethod

**def** fourth(x):

        return exp(x)-x\*\*2+10\*cos(x)

    @staticmethod

**def** f1(x):

        return (2\*x+1)/3

    @staticmethod

**def** f2(x):

        return cos(x)-exp(x)

    @staticmethod

**def** f3(x):

        return -2/(sqrt(1-1/(x\*\*4))\*x\*\*3)

    @staticmethod

**def** f4(x):

        return (exp(x)-10\*sin(x))/2/sqrt(abs(exp(x)+10\*cos(x)))

**def** binary\_search(self, a, b, eps, i):

        k = 0

        x = (a+b)/2

        while abs(a-b) > eps:

            x = (a+b)/2

            if self.eq[i](a)\*self.eq[i](x) < 0:

                b = x

            else:

                a = x

            k+=1

        return {"x": x, "f": self.eq[i](x), "k":k}

**def** check\_for\_roots(self,a,b,i,s):

        if s>100:

            return True

        if(self.eq[i](a)\*self.eq[i](b)>0):

            raise ValueError("The function has no roots in the given interval")

**def** secant\_method(self, a, b, eps, i):

        k = 0

        x = [a]

        x.append((a+b)/2)

        while abs(self.eq[i](x[-1]))>eps:

            x.append(x[-1]-(x[-1]-x[-2])/(self.eq[i](x[-1])-self.eq[i](x[-2]))\*self.eq[i](x[-1]))

            k+=1

        return {"x": x[-1], "f": self.eq[i](x[-1]), "k":k}

**def** simple\_iteration(self,a,b,eps,i):

        k = 0

        s = True

        m = 0

        for x in range(int(a/eps),int(b/eps),int(eps/eps)):

            if abs(self.d[i](x))>1:

                s = False

                if abs(self.d[i](x))>m:

                    m = abs(self.d[i](x))

        x = (a+b)/2

        if s:

            x = x - 1/m\*self.d[i](x)

            while abs(self.eq[i](x))>eps:

                x = x - 1/m\*self.d[i](x)

                k+=1

            if (x == x - 1/m\*self.d[i](x)):

                raise IndexError("The method is not converging")

            return {"x": x, "f": self.eq[i](x), "k":k}

        else:

            k = 0

            x = (a+b)/2

            x = x - 1/m\*self.d[i](x)

            while abs(self.eq[i](x))>eps:

                x = x - 1/m\*self.d[i](x)

                k+=1

                if k>200:

                    raise IndexError("The method is not converging")

            if (x == x - 1/m\*self.d[i](x)):

                raise IndexError("The method is not converging")

            return {"x": x, "f": self.eq[i](x), "k":k}

from math import tan,sin,cos,exp,log

**class** System:

    @staticmethod

**def** firstdx(x,y):

        return (-(x+tan(y)-1)/(sin(x))\*\*2-1/(tan(x))-2\*y-3)/(1/(cos(y)\*\*2\*sin(x)\*\*2)+2)

    @staticmethod

**def** firstdy(x,y):

        return -x-tan(y)+1-System.firstdx(x,y)/cos(y)\*\*2

    @staticmethod

**def** secondx(x,y):

        return ((y-exp(x)+5)/y-log(y)+3)/((exp(x))/y-1)

**def** secondy(x,y):

        return -exp(x)\*System.secondx(x,y)-y-exp(x)+5

    @staticmethod

    @staticmethod

**def** Newton(a0,b0,a,b,e,eq):

        x = (a0+a)/2

        y = (b0+b)/2

        if(eq=="1"):

            while True:

                x1 = x + System.firstdx(x,y)

                y1 = y + System.firstdy(x,y)

                if abs(x1-x)<e and abs(y1-y)<e:

                    return [x1,y1]

                x = x1

                y = y1

        elif(eq=="2"):

            while True:

                x1 = x + System.seconddx(x,y)

                y1 = y + System.seconddy(x,y)

                if abs(x1-x)<e and abs(y1-y)<e:

                    return [x1,y1]

                x = x1

                y = y1

## Пример работы программы

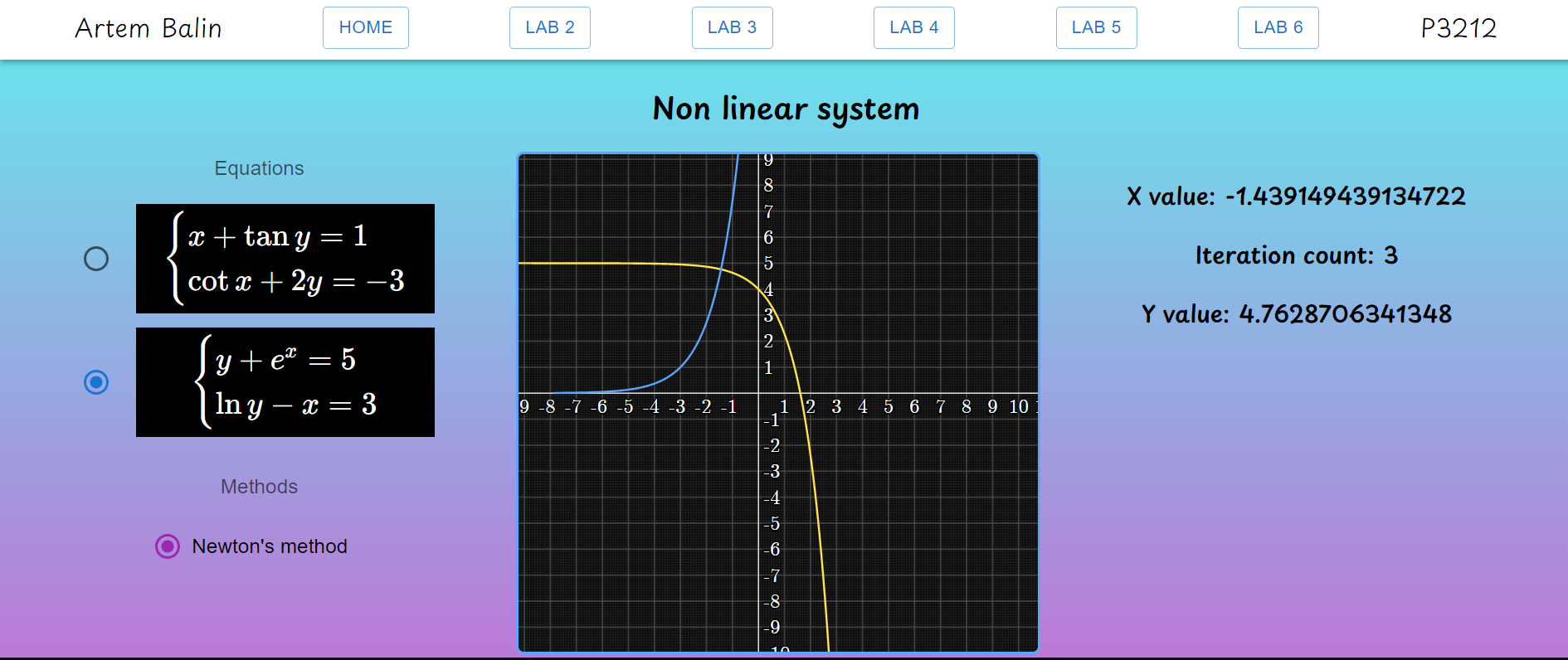


Рисунок 5. Пример работы программы.

# Вывод

В ходе реализации данной лабораторной работы я ознакомился с численными методами решения нелинейных уравнений и их систем.