***Санкт-Петербургский национальный исследовательский университет ИТМО***

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Вычислительная математика»

**Отчет**

По лабораторной работе №2

Вариант 9

Выполнила:

Миличенко А.Е.

P3210

Преподаватель:

Малышева Т.А.

Санкт-Петербург

2024 г.

**Цель работы**

Изучить численные методы решения нелинейных уравнений и их

систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений,

выполнить программную реализацию методов.

**Листинг программы**

Файл system\_main.py:

import math

import matplotlib.pyplot as plt

def simple\_iteration\_method(func1, func2, x01, x02, a1, b1, a2, b2, eps):

x1 = func1(x01)

x2 = func2(x02)

while abs(x1 - x01) > eps or abs(x2 - x02) > eps:

x2, x02 = func1(x1), x2

x1, x01 = func2(x2), x1

return x1, x2

def draw\_plots(func1, func2, a, b, root, eps):

xs1, xs2 = [], []

ys1, ys2 = [], []

x = a

while x < b:

xs1.append(x)

ys1.append(func1(x))

xs2.append(func2(x))

ys2.append(x)

x += eps

plt.xlabel("X")

plt.ylabel("Y")

plt.scatter([root], [func1(root)], color='red')

plt.plot(xs1, ys1, 'r')

plt.plot(xs2, ys2, 'g')

plt.plot([a, b], [0, 0], 'b')

plt.show()

def choice\_system(choice):

if choice == 1:

return lambda x: math.sin(x + 1), lambda y: 1 - math.cos(y) / 2, 0.6, 0.8, 0.8, 1

elif choice == 2:

return lambda x: -math.cos(x - 1) + 0.5, lambda y: math.cos(y) + 3, 3.2, 3.4, 1.1, 1.3

print('Выберите понравившуюся систему(введите ниже ее номер):')

print('1. sin(x + 1) - y == 0 и 2x + cosy = 2')

print('2. cos (x – 1) + y == 0.5 и x – cos (y) == 3')

choice = input()

while choice not in {'1', '2'}:

print('Введите номер понравившейся системы')

choice = input()

choice = int(choice)

f1, f2, a1, b1, a2, b2 = choice\_system(choice)

x01 = (a1 + b1) / 2

x02 = (a2 + b2) / 2

eps = 0.000001

print(simple\_iteration\_method(f1, f2, x01, x02, a1, b1, a2, b2, eps))

root = simple\_iteration\_method(f1, f2, x01, x02, a1, b1, a2, b2, eps)[0]

draw\_plots(f1, f2, -2, 4, root, 0.1)

Файл eq\_methods.py:

def proiz\_at\_point(func, x, eps=1e-6):

return (func(x + eps) - func(x - eps)) / (2 \* eps)

#Метод хорд

def horde\_method(f, a, b, eps=10e-3):

x0 = a

for i in range(1, 1000):

x1 = a - ((b-a)/(f(b) - f(a))) \* f(a)

if abs(x1 - x0) <= eps or abs(f(x1)) <= eps:

return x1

break

if f(x1) \* f(x0) < 0:

b = x1

else:

a = x1

x0 = x1

def newton\_method(f, df, x0, eps=10e-3):

for i in range(1, 1000):

x1 = x0 - f(x0) / df(x0)

if (

abs(x1 - x0) <= eps or

abs(f(x1) / df(x1)) <= eps or

abs(f(x1)) <= eps

):

return x1

x0 = x1

def simple\_iteration\_method(f, x0, a, b, eps):

m\_f = 0

x = a

while x < b:

m\_f = max(m\_f, abs(proiz\_at\_point(f, x)))

x = x + eps

if proiz\_at\_point(f, a) > 0:

h = -1/m\_f

else:

h = 1/m\_f

phi = lambda x: x + h \* f(x)

phi\_ = lambda x: 1 + р \* proiz\_at\_point(f, x)

x = phi(x0)

while abs(x-x0) > eps:

x, x0 = phi(x), x

return x

Файл eq\_main.py:

import eq\_methods

import math

import matplotlib.pyplot as plt

def choice\_interval(choice):

if choice == 1:

f = lambda x: x \*\* 3 + 2.84 \* x \*\* 2 - 5.606 \* x - 14.766

print("Выберите понравившийся интервал(напишите одну из цифр 1, 2 или 3)")

print("(-3.2, -3) - 1, (-2.2, -2) - 2, (2.2, 2.4) - 3")

interval = input()

while interval not in {'1','2','3'}:

print("Выберите понравившийся интервал(напишите одну из цифр 1, 2 или 3)")

interval = input()

if interval == '1':

a, b = -3.2, -3

return f, a, b

elif interval == '2':

a, b = -2.2, -2

return f, a, b

elif interval == '3':

a, b = 2.2, 2.4

return f, a, b

elif choice == 2:

f = lambda x: x \*\* 3 - x + 4

a, b = -2, -1

return f, a, b

elif choice == 3:

f = lambda x: math.sin(x \*\* 2) + x + 2

a, b = -2, -1.6

return f, a, b

def get\_derivative\_at\_point(func, x0, dx=0.000001):

return (func(x0 + dx) - func(x0)) / dx

def draw\_plot(func, a, b, root, eps):

xs = []

ys = []

x = a

while x < b:

xs.append(x)

ys.append(func(x))

x += eps

plt.xlabel("X")

plt.ylabel("Y")

plt.plot(xs, ys, 'g')

plt.scatter([root], [func(root)], color='red')

plt.plot([a, b], [0, 0], 'b')

plt.show()

def verify(f, a, b):

return f(a) \* f(b) < 0

def proiz\_f(x):

if choice == 1:

return 3\*(x\*\*2) + 5.68\*x - 5.606

elif choice == 2:

return 3\*(x\*\*2) - 1

elif choice == 3:

return 2\*x\*(math.cos(x\*\*2)) + 1

print('Выберите понравившуюся функцию(в ответ напишите ее номер):')

print('1. x \*\* 3 + 2.84 \* x \*\* 2 - 5.606 \* x - 14.766')

print('2. x \*\* 3 - x + 4')

print('3. sin(x \*\* 2) + x + 2')

choice = input()

while choice not in ['1', '2', '3']:

print('Введите номер понравившейся функции:')

choice = input()

choice = int(choice)

function, a, b = choice\_interval(choice)

x0 = (a+b)/2

eps =0.0001

print("Метод хорд:", eq\_methods.horde\_method(function, a, b))

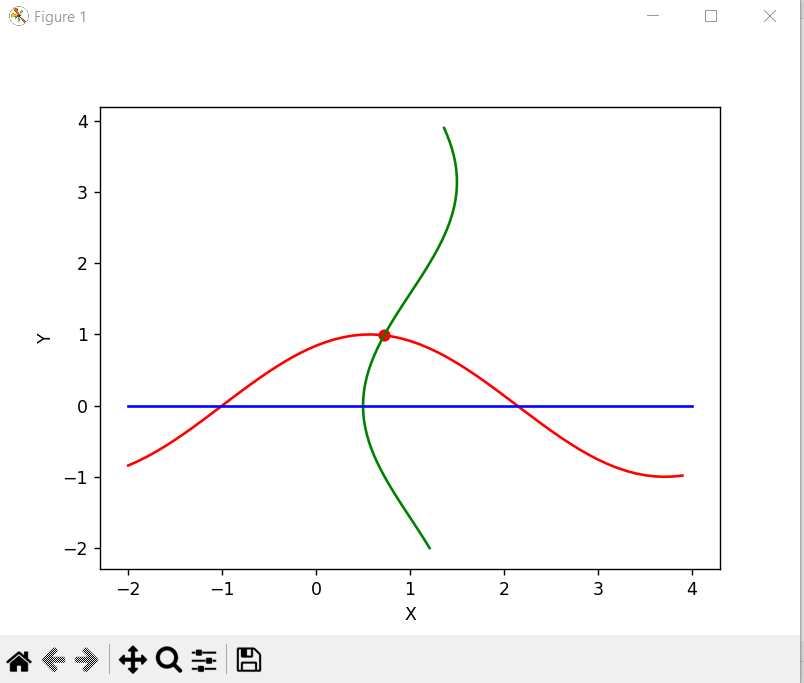
root = eq\_methods.horde\_method(function, a, b)

draw\_plot(function, a, b, root, eps)

print("Метод Ньютона:", eq\_methods.newton\_method(function, proiz\_f, x0))

print("Метод простой итерации:", eq\_methods.simple\_iteration\_method(function, x0, a, b , eps))

**Результат работы программы**



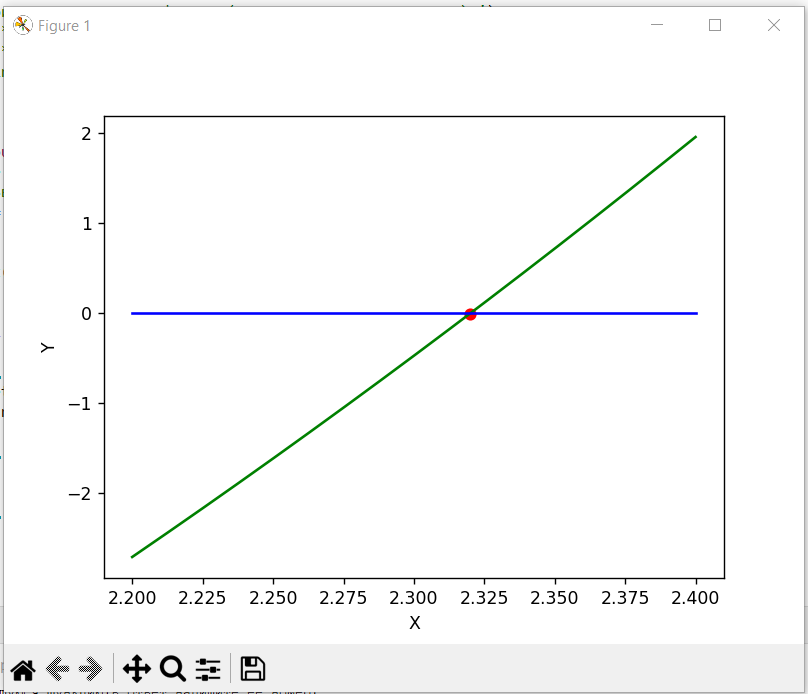
Выберите понравившуюся систему(введите ниже ее номер):

1. sin(x + 1) - y == 0 и 2x + cosy = 2

2. cos (x – 1) + y == 0.5 и x – cos (y) == 3

1

(0.7248830919883416, 0.9881520991258634)



Выберите понравившуюся функцию(в ответ напишите ее номер):

1. x \*\* 3 + 2.84 \* x \*\* 2 - 5.606 \* x - 14.766

2. x \*\* 3 - x + 4

3. sin(x \*\* 2) + x + 2

1

Выберите понравившийся интервал(напишите одну из цифр 1, 2 или 3)

(-3.2, -3) - 1, (-2.2, -2) - 2, (2.2, 2.4) - 3

3

Метод хорд: 2.319816963031261

**Вывод**

В процессе изучения численных методов решения нелинейных уравнений и их систем мы рассмотрели различные подходы, такие как метод половинного деления, метод Ньютона, метод секущих и метод простых итераций. Каждый из этих методов имеет свои особенности, преимущества и ограничения.