Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Методы численного анализа»

ОТЧЕТ

к лабораторной работе №4

на тему:

«РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ»

БГУИР 6-05 0612 02 86

Выполнил студент группы 353505
МАРТЫНКЕВИЧ Евгений
Дмитриевич

(дата, подпись студента)

Проверил доцент кафедры
информатики
АНИСИМОВ Владимир Яковлевич

(дата, подпись преподавателя)

Минск 2024

Содержание

Цель работы

- изучить метод простых итераций и метод Ньютона решения нелинейных уравнений;
- составить программу решения нелинейных уравнений указанными методами, применимую для организации вычислений на ЭВМ;
- выполнить тестовые примеры и проверить правильность работы программы

ЗАДАНИЕ. Решить систему нелинейных уравнений:

$$tg(xy+m)=x \qquad \text{где } x>0,\ y>0,$$
$$ax^2+2y^2=1,$$

с точностью до 0,0001 методами простых итераций и Ньютона,

принимая для номера варианта k значения параметров a и m из таблицы:

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14
m	0,0	0,1	0,1	0,2	0,2	0,3	0,3	0,4	0,4	0,1	0,2	0,3	0,2	0,2
a	0,5	0,6	0,7	0,8	0,9	1,0	0,5	0,6	0,7	0,8	0,9	1,0	0,7	0,5

Начальные приближения найти графически. Сравнить скорость сходимости методов.

Вариант 3

Программная реализация

Для проверки решения подставим найденный корень в функцию и найдем ее значение.

Исходные данные

Система, полученная в результате подстановки в ***m*** и ***a***:

```
-x + tan(x*y + 0.1) = 0
0.7*x**2 + 2*y**2 - 1 = 0
```

Примечание: x**n – возведение x в степень n.

Код метода простых итераций:

```
def simple_method(phi_x, phi_y, X):
    X_solve = Matrix(X)
    X_next = Matrix([[0.0], [0.0]])
    count = 0
    while count < 2000:
        X_next[0] = phi_x.subs([(x, X_solve[0]),(y, X_solve[1])])
        X_next[1] = phi_y.subs([(x, X_solve[0]),(y, X_solve[1])])
        if max(abs(X_next - X_solve)) < 10 ** (-4):
            print(f"Кол-во итераций {count}")
            return X_next
        X_solve = X_next.copy()
        count += 1
    return X_solve
```

Код метода Ньютона:

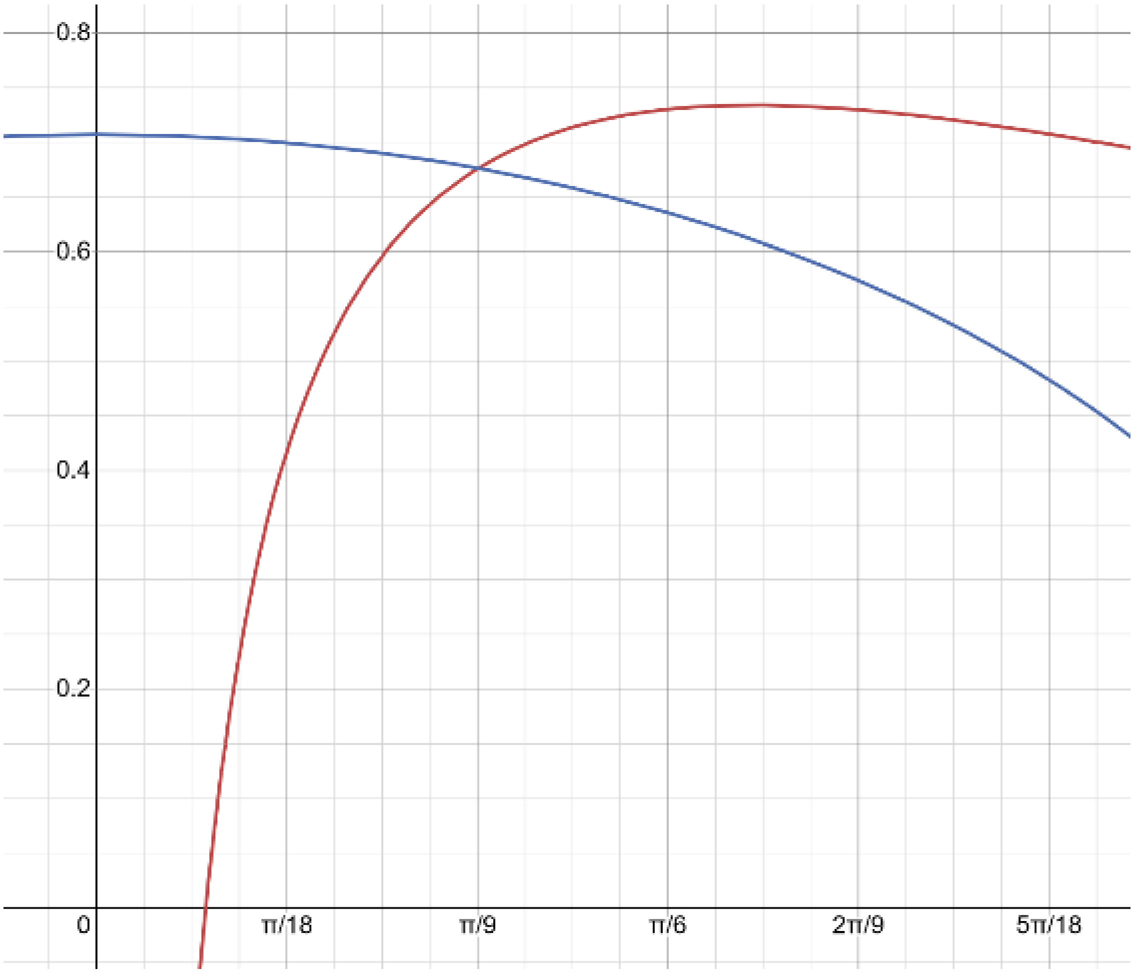
```
def newton_method(f1, f2, X):
    X_solve = Matrix(X)
    J = Matrix([[f1.diff(x), f1.diff(y)], [f2.diff(x), f2.diff(y)]])
    F = Matrix([[f1], [f2]])
    count = 0
    while count < 2000:
        NumJ = get_numeric_matrix(J, X_solve)
        NumF = get_numeric_matrix(F, X_solve)
        if(NumJ.det() == 0):
            print("Определитель матрицы Якоби равен 0 :(")
            exit()
        X_next = X_solve - (NumJ ** (-1) * NumF)
        if(max(abs(X_next - X_solve)) < 10 ** (-4)):
            print(f"Кол-во итераций: {count}")
            return X_next
        X_solve = X_next.copy()
        count += 1
    return X_solve


def get_numeric_matrix(Matr, X_solve):
    Numeric = Matr.copy()
    tup = Matr.shape
    for i in range(0, tup[0] * tup[1]):
        Numeric[i] = Matr[i].subs([(x, X_solve[0]), (y, X_solve[1])])
    return Numeric
```

Полученные результаты

```
Исходная система уравнений:
-x + tan(x*y + 0.1) = 0
0.7*x**2 + 2*y**2 - 1 = 0
Приближение:
[0.3, 0.6]
Метод простых итераций:
Кол-во итераций 14
Вектор решений:
Matrix([[0.349690667221588], [0.676183697707782]])
Значение f1 в точке: 6.27486198832017e-5
Значение f2 в точке: 4.72800108549665e-5
Метод Ньютона:
Кол-во итераций: 3
Вектор решений:
Matrix([[0.349869684918810], [0.676133804251189]])
Значение f1 в точке: -6.10622663543836e-16
Значение f2 в точке: 1.55431223447522e-15
```

График функций:



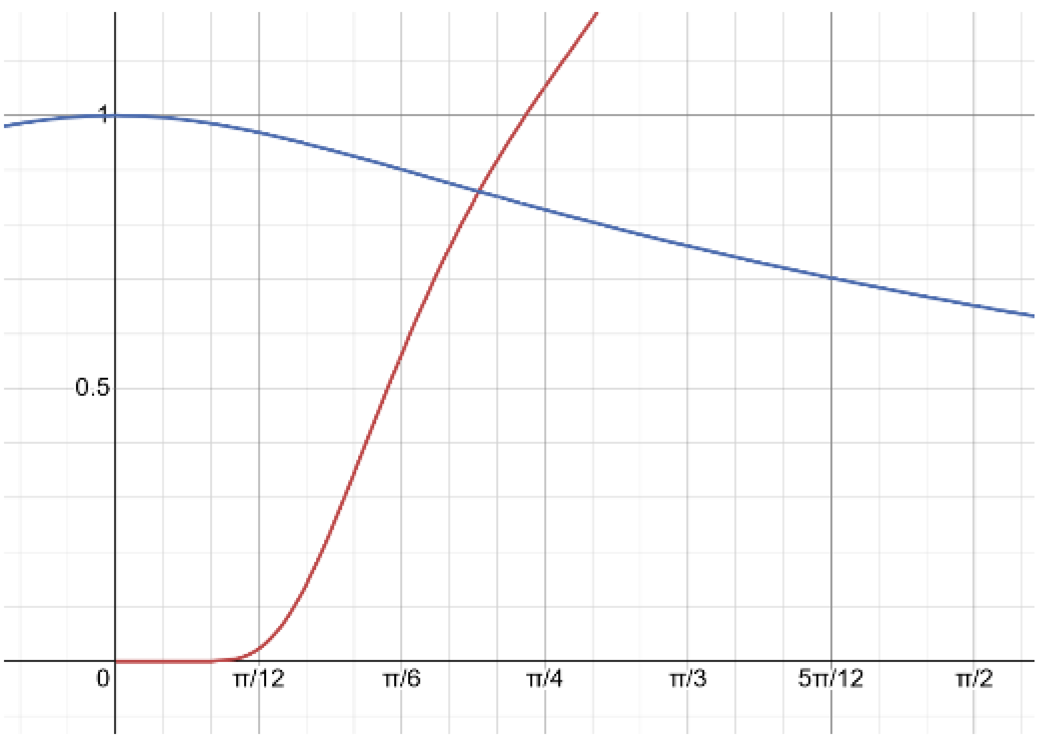
Результаты вычислений:

	Метод простых итераций	*Метод Ньютона*
Решение найденное с точностью 10^-4	x = 0.34969066 y = 0.67618370	x = 0.34986968 y = 0.67613380
Функция-1, Функция-2	0.00006274 0.00004728	-6.106e-16 1.554e-15
Кол–во итераций	14	3

Тестовый пример 1.

```
Исходная система уравнений:
-x + sin(x*y)**x = 0
-y + cos(x*y)**y = 0
Приближение:
[0.6, 0.8]
Метод простых итераций:
Кол-во итераций 7
Вектор решений:
Matrix([[0.665316078894085], [0.860954006618983]])
Значение f1 в точке: -9.85763470140455e-6
Значение f2 в точке: -2.94765634878402e-6
Метод Ньютона:
Кол-во итераций: 2
Вектор решений:
Matrix([[0.665304672854928], [0.860955196779156]])
Значение f1 в точке: 6.29988283762373e-11
Значение f2 в точке: -6.07131012131390e-11
```

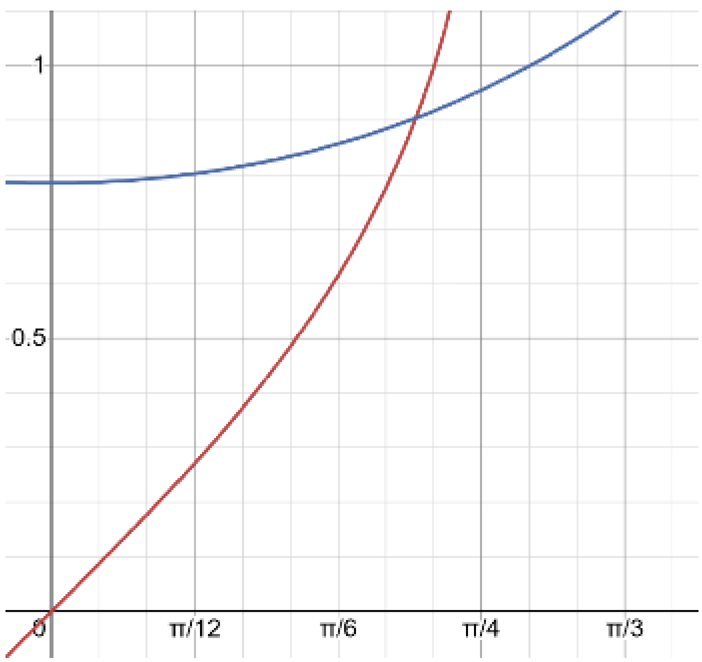
График функций:



Тестовый пример 2.

```
Исходная система уравнений:
-sin(y) + tan(x) = 0
-cos(x) + cot(y) = 0
Приближение:
[0.7, 0.9]
Метод простых итераций:
Кол-во итераций 7
Вектор решений:
Matrix([[0.666255074360238], [0.904554823195875]])
Значение f1 в точке: 2.65894007531742e-5
Значение f2 в точке: 1.30184282509660e-5
Метод Ньютона:
Кол-во итераций: 2
Вектор решений:
Matrix([[0.666239432492970], [0.904556894302416]])
Значение f1 в точке: 7.13540337926588e-13
Значение f2 в точке: 2.24043006369357e-13
```

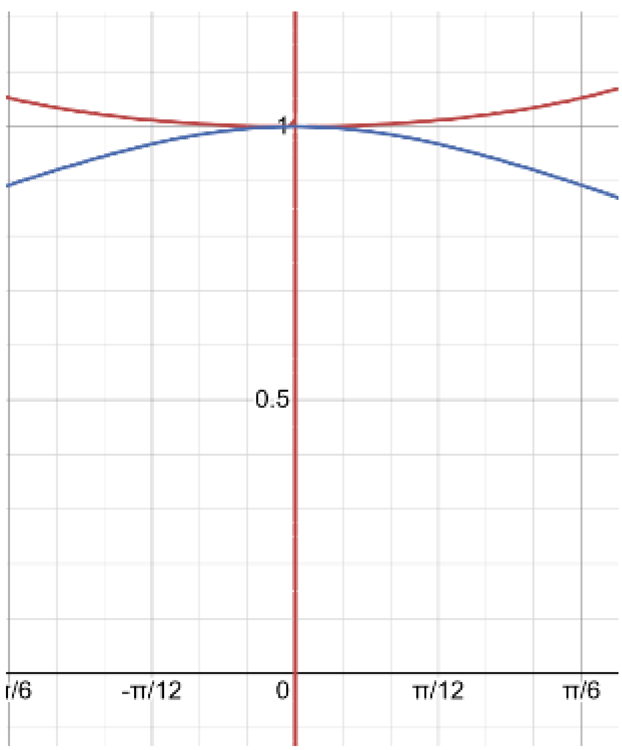
График функций:



Тестовый пример 3.

```
Исходная система уравнений:
-x + sin(x*y) = 0
-y + cos(x*y) = 0
Приближение:
[0.1, 0.9]
Метод простых итераций:
Кол-во итераций 174
Вектор решений:
Matrix([[0.0529522903669825], [0.998597043328735]])
Значение f1 в точке: -9.89282035142106e-5
Значение f2 в точке: 5.24092061415793e-6
Метод Ньютона:
Кол-во итераций: 15
Вектор решений:
Matrix([[0.000115903951080631], [0.999999995458787]])
Значение f1 в точке: -7.85848125396844e-13
Значение f2 в точке: -2.17565021554122e-9
```

График функций:



Вывод

В ходе выполнения лабораторной работы я изучил метод простых итераций и метод Ньютона решения нелинейных уравнений, написал программу их реализации на языке Python, правильность работы программы проверил на тестовых примерах.

На основании тестов можно сделать следующие выводы:

- Программа позволяет получить решения системы с заданной точностью (заданная точность в условиях лабораторной работы 10^-4);
- Метод Ньютона эффективнее по сравнению с методом простых итераций, так как затрачивает меньшее число итераций;
- Оптимальным способом решения нелинейных уравнений является применение метода Ньютона, так скорость сходимости в этом методе почти всегда квадратичная.
- Основной недостаток методов – малая область сходимости ($\bar{x}_0$ должен быть достаточно близок к решению уравнения).