Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы численного анализа

**ОТЧЁТ**

к лабораторной работе №4

на тему

Решение систем нелинейных уравнений

Выполнил: студент группы 253505

Снежко Максим Андреевич

Проверил: Анисимов Владимир Яковлевич 

Минск 2023

**Содержание**

[1. ЦЕЛЬ РАБОТЫ 3](#_Toc116828981)

[2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 4](#_Toc116828982)

[3. АЛГОРИТМ РЕШЕНИЯ 9](#_Toc116828983)

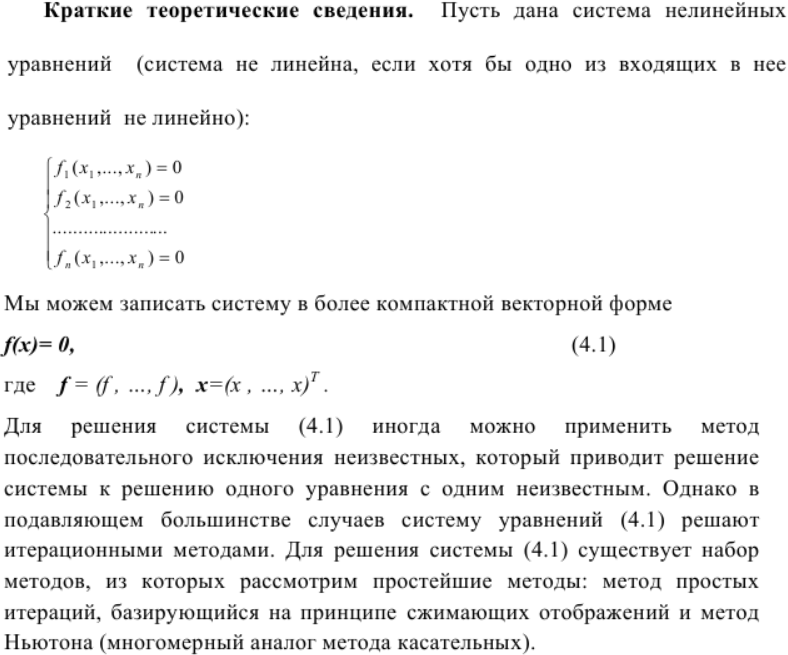
[4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 10](#_Toc116828984)

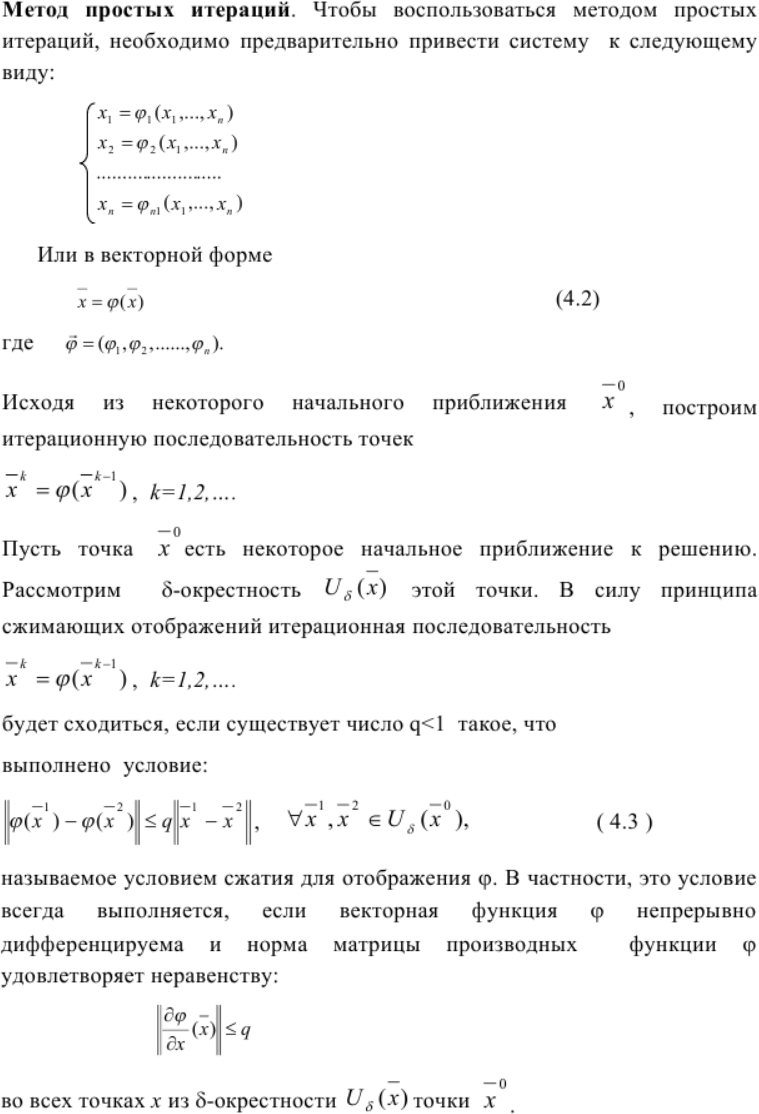
[5. ТЕСТОВЫЕ ПРИМЕРЫ 12](#_Toc116828985)

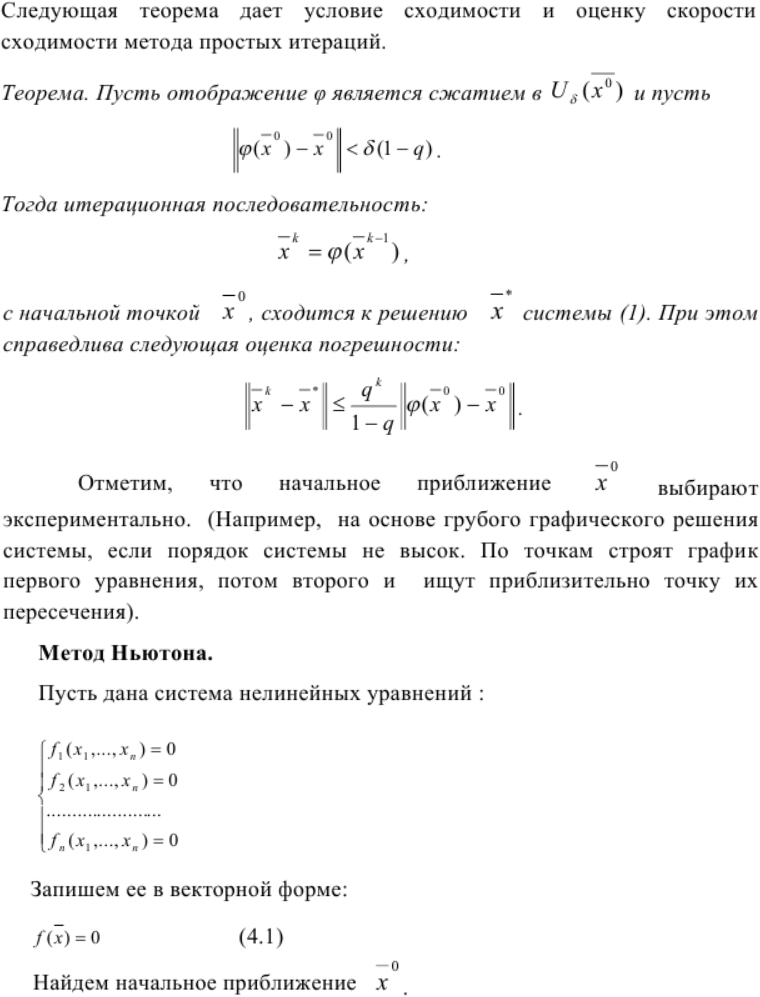
[6. ЗАДАНИЕ 17](#_Toc116828986)

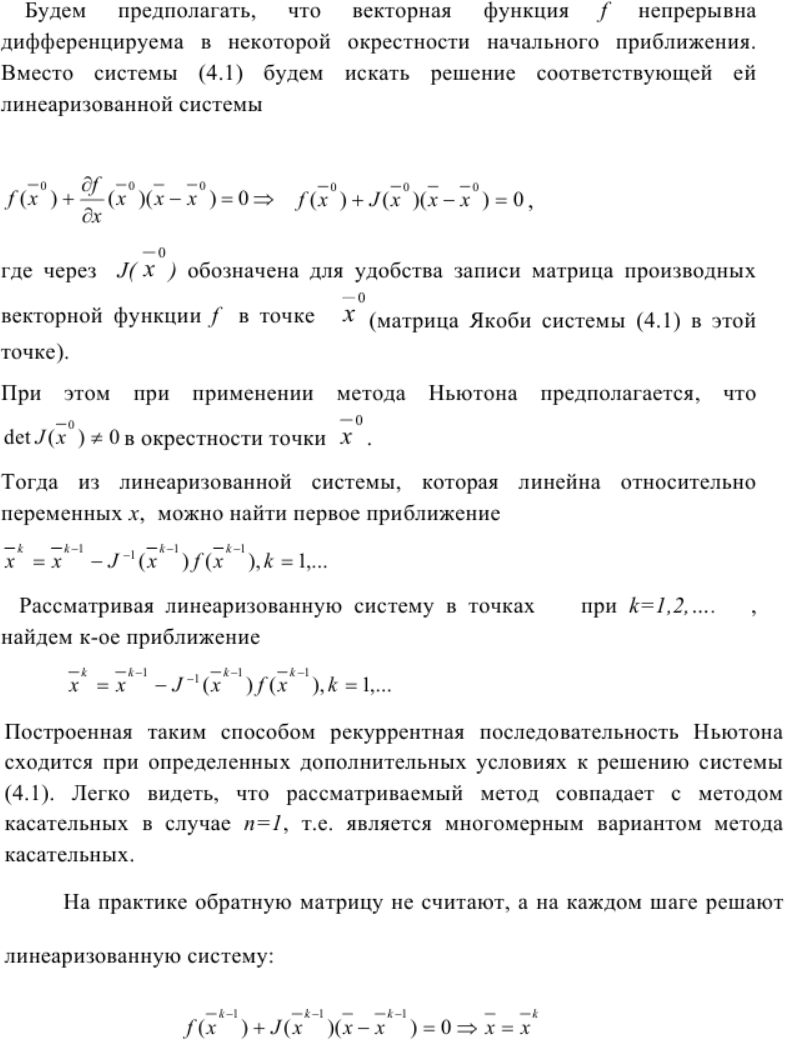
[7. ВЫВОД 18](#_Toc116828987)

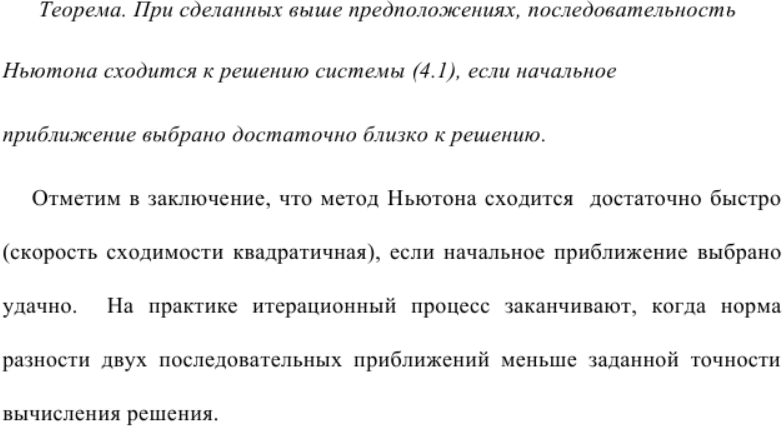
1. **ЦЕЛЬ РАБОТЫ**
2. Изучить методы численного решения систем нелинейных уравнений (метод простой итерации, метод Ньютона).
3. Составить алгоритм и программу численного решения нелинейных уравнений методами простой итерации и Ньютона.
4. Проверить правильность работы программы на тестовых примерах.
5. Численно решить систему нелинейных уравнений заданного варианта.
6. Сравнить число итераций (трудоемкость), необходимых для достижения заданной точности вычисления разными методами.
7. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**











# **АЛГОРИТМ РЕШЕНИЯ**



# **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

import numpy

import sympy

m = 0.2

a = 0.9

iters = 0

EPS = 10.0 \*\* -4

# Исходные уравнения в формате f(x,y) = 0

(x, y) = sympy.symbols("x y")

eq1 = sympy.tan(x \* y + m) - x

eq2 = a \* (x \*\* 2) + 2 \* (y \*\* 2) - 1

print("Система нелинейных уравнений:")

print(eq1, "= 0")

print(eq2, "= 0")

print()

# Вычисление исходных уравнений: f1(x,y), f2(x,y)

def val1(x, y):

return numpy.tan(x \* y + m) - x

def val2(x, y):

return a \* (x \*\* 2) + 2 \* (y \*\* 2) - 1

# Вычисление из исходных уравнений: x = phi(x,y), y = phi(x,y)

def eqx(x, y):

return numpy.tan(x \* y + m)

def eqy(x, y):

return numpy.sqrt((1 - a \* (x \*\* 2)) / 2)

# Вычисление матрицы Якоби

def J(x, y):

return numpy.array([

[(1 + numpy.tan(x \* y + m) \*\* 2) \* y - 1, (1 + numpy.tan(x \* y + m) \*\* 2) \* x],

[2 \* a \* x, 4 \* y]

])

# Графики исходных уравнений

plots = sympy.plot\_implicit(sympy.Eq(eq1, 0), (x, -2, 2), (y, -2, 2), line\_color = "blue", show = False)

plots.extend(sympy.plot\_implicit(sympy.Eq(eq2, 0), (x, -2, 2), (y, -2, 2), line\_color = "red", show = False))

# Метод простых итераций

def SimpleIterations(x0, y0):

global iters

iters = 0

(x, y) = (x0, y0)

while True:

iters += 1

oldx = x

oldy = y

x = eqx(x, y)

y = eqy(x, y)

if (not (numpy.isfinite(x) and numpy.isfinite(y))):

raise RuntimeError("Последовательность {x} расходящаяся")

if (max(abs(x - oldx), abs(y - oldy)) < EPS):

return (x, y)

# Метод Ньютона

def NewtonMethod(x0, y0):

global iters

iters = 0

(x, y) = (x0, y0)

while True:

iters += 1

j = J(x, y)

f = numpy.array([[val1(x, y)], [val2(x, y)]])

deltas = numpy.linalg.solve(j, -f)

x += deltas[0][0]

y += deltas[1][0]

if (not (numpy.isfinite(x) and numpy.isfinite(y))):

raise RuntimeError("Последовательность {x} расходящаяся")

if (max(abs(deltas)) < EPS):

return (x, y)

x0 = 0.9

y0 = 0.5

print("Начальное приближение: \n(x0, y0) = ", (x0, y0))

print()

def solute(method):

global iters

try:

(x, y) = method(x0, y0)

print(f"{method.\_\_name\_\_} (итераций: {iters})")

print(f"(x, y) = ({x:.4f}, {y:.4f})")

except Exception as ex:

print(f"Ошибка: {ex} - в {method.\_\_name\_\_} (итераций: {iters})")

print()

SimpleIterations.\_\_name\_\_ = "Метод простых итераций"

NewtonMethod.\_\_name\_\_ = "Метод Ньютона"

solute(SimpleIterations)

solute(NewtonMethod)

plots.show()

1. **ТЕСТОВЫЕ ПРИМЕРЫ**

Тестовый пример 1

Решить систему нелинейных уравнений с точностью до 0,0001 методами простых итераций и Ньютона:

Ответ:

|  |  |
| --- | --- |
|  | |
| Начальное приближение: | |
| Метод простых итераций | Метод Ньютона |
|  |  |
| Количество итераций | |
| 19 | 4 |

Тестовый пример 2

Решить систему нелинейных уравнений с точностью до 0,0001 методами простых итераций и Ньютона:

Ответ:

|  |  |
| --- | --- |
|  | |
| Начальное приближение: | |
| Метод простых итераций | Метод Ньютона |
|  |  |
| Количество итераций | |
| 19 | 5 |

Тестовый пример 3

Решить систему нелинейных уравнений с точностью до 0,0001 методами простых итераций и Ньютона:

Ответ:

|  |  |
| --- | --- |
|  | |
| Начальное приближение: | |
| Метод простых итераций | Метод Ньютона |
|  |  |
| Количество итераций | |
| 423561 | 4 |

Тестовый пример 4

Решить систему нелинейных уравнений с точностью до 0,0001 методами простых итераций и Ньютона:

Ответ:

|  |  |
| --- | --- |
|  | |
| Начальное приближение: | |
| Метод простых итераций | Метод Ньютона |
|  |  |
| Количество итераций | |
| 6 | 4 |

Тестовый пример 5

Решить систему нелинейных уравнений с точностью до 0,0001 методами простых итераций и Ньютона:

Ответ:

|  |  |
| --- | --- |
|  | |
| Начальное приближение: | |
| Метод простых итераций | Метод Ньютона |
|  |  |
| Количество итераций | |
| 6 | 1337 |

1. ЗАДАНИЕ

Вариант 11

Решить систему нелинейных уравнений с точностью до 0,0001 методами простых итераций и Ньютона:

Ответ:

|  |  |
| --- | --- |
|  | |
| Начальное приближение: | |
| Метод простых итераций | Метод Ньютона |
|  |  |
| Количество итераций | |
| 10 | 4 |

# **ВЫВОД**

Таким образом, в ходе выполнения лабораторной работы были изучены методы численного решения систем нелинейных уравнений (метод простой итерации, метод Ньютона), составлен алгоритм и программа численного решения систем нелинейных уравнений методами простой итерации и Ньютона, проверена правильность работы программы на тестовых примерах, численно решена система уравнений заданного варианта.

Исходя из полученных результатов, можно сделать вывод о большей трудоемкости метода простых итераций (тестовые примеры 1,2,3,4). Метод простых итераций обладает линейной скоростью сходимости. Метод Ньютона сходится достаточно быстро (квадратичная скорость сходимости), если начальное приближение выбрано удачно (в примере 5 выбрано неудачно).