**Методы численного анализа**

**Отчёт по лабораторной работе №2**

**Вариант 5**

**Студент 2 курса 8 группы**

Агинский Антон Викторович

**Преподаватель**

Будник Анатолий Михайлович

ФПМИ БГУ  
2022

**Постановка задачи**

Найти решение системы нелинейных уравнений

применяя следующие методы:

1. Метод Гаусса-Зейделя с реализацией по методу простой итерации.

2. Метод Ньютона.

Необходимо:

А) Отделить один корень системы.

Б) Выбрать начальное приближение, найти двумя вышеуказанными методами решение данного нелинейного уравнения с точностью . Критерий останова итерационного процесса: .

В) Сравнить методы по скорости сходимости и точности.

**Отделение корней**

Отделим корень исходной системы следующим образом: выразим из второго уравнения и подставим в первое. Далее, найдя найдём начальное приближения таким же образом, как мы находили его при отделении корня одного нелинейного уравнения.

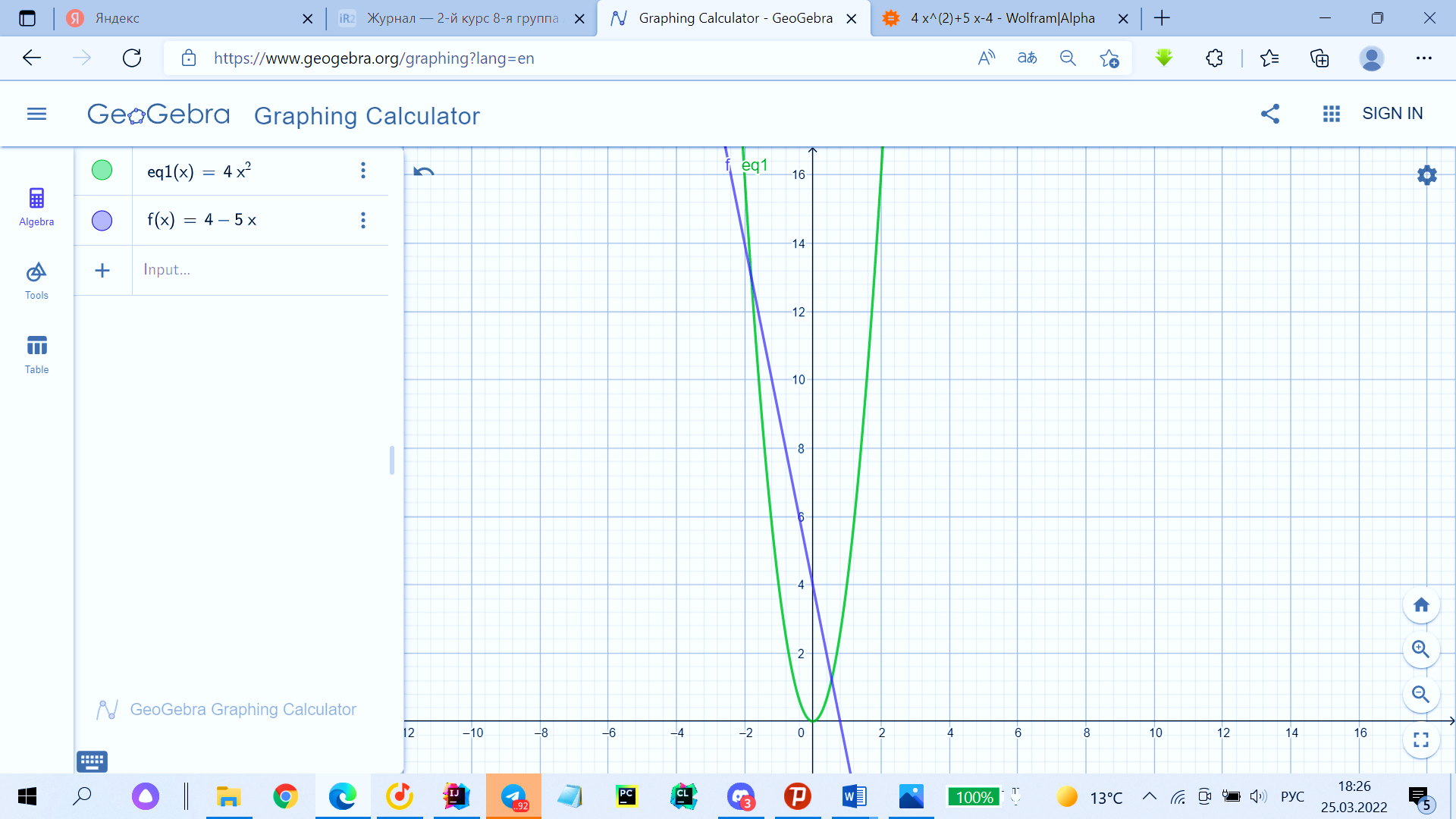
Из второго уравнения , значит . Представим в виде , где . Построим графики этих функций и найдём их пересечение.

Из графика на следующей странице видим, что одна из точек пересечения графиков . Подтвердим графические соображения. Имеем :

, ,

,

Из этого следует, что на найденном отрезке уравнение имеет единственный корень и для нахождения корня с заданной точностью за начальное приближение можно взять .



**Нахождение решения**

***1. Метод Гаусса-Зейделя с реализацией по методу простой итерации.***

*Алгоритм*

Используя, построим последовательность приближений по следующему правилу:

1) Решаем уравнение

(1)

относительно методом простой итерации для решения нелинейного уравнения с одной переменной.

2) Решаем уравнение

(2)

относительно методом простой итерации для решения нелинейного уравнения с одной переменной.

3) Если , то останавливаем итерационный процесс и принимаем за наше решение точку , в противном случае переходим к пункту 1).

Для проверки выполнения критерия останова в пункте 3) используем следующую норму:

Для решения уравнения (1) методом просто итерации приведем его к виду, удобному для итерации относительно :

Дальнейший алгоритм МПИ описан в лабораторной работе №1 (в качестве критерия останова выбираем ).

Аналогично решаем уравнение (2), удобный для итерации вид которого следующий:

Невязка найденного решения равна .

Для последующего анализа скорости сходимости храним количество итераций в переменной *iterationsQuantity* и, соответственно, увеличиваем переменную после каждой итераций внешнего цикла.

*Листинг*

Класс *Point* для хранения полученного решения:

class Point {

double x;

double y;

}

Функция для нахождения :

double norm(double x\_1, double y\_1, double x\_2, double y\_2) {

double x = x\_1 - x\_2;

double y = y\_1 - y\_2;

return max(abs(x), abs(y));

}

Функция для нахождения :

double f(double x, double y) { return (y \* x + y - 5); }

Функция для нахождения :

double phiForF(double x, double y) { return ((5 - y) / y); }

Функция для нахождения решения уравнения (1):

double fixedPointIterationForF(double x\_K, double eps, double y\_K) {

double x\_KPlusOne = phiForF(x\_K, y\_K);

while(Math.abs(x\_K - x\_KPlusOne) >= eps) {

x\_K = x\_KPlusOne;

x\_KPlusOne = phiForF(x\_K, y\_K);

}

return x\_KPlusOne;

}

Функция для нахождения :

double phiForG(double y, double x) { return (4 \* x + 1); }

Функция для нахождения решения уравнения (2):

double fixedPointIterationForG(double y\_K, double eps, double x\_KPlusOne) {

double y\_KPlusOne = phiForG(y\_K, x\_KPlusOne);

while(Math.abs(y\_K - y\_KPlusOne) >= eps) {

y\_K = y\_KPlusOne;

y\_KPlusOne = phiForG(y\_K, x\_KPlusOne);

}

return y\_KPlusOne;

}

Функция для нахождения решения СНУ методом Гаусса-Зейделя:

Point GaussSeidelMethod(double x\_null, double y\_null, double eps) {

double x\_K = x\_null;

double y\_K = y\_null;

double x\_KPlusOne = fixedPointIterationForF(x\_K, eps, y\_K);

double y\_KPlusOne = fixedPointIterationForG(y\_K, eps, x\_KPlusOne);

int iterationsQuantity = 1;

while(norm(x\_KPlusOne, y\_KPlusOne, x\_K, y\_K) >= eps) {

x\_K = x\_KPlusOne;

y\_K = y\_KPlusOne;

x\_KPlusOne = fixedPointIterationForF(x\_K, eps, y\_K);

y\_KPlusOne = fixedPointIterationForG(y\_K, eps, x\_KPlusOne);

iterationsQuantity++;

}

Point solution = new Point();

solution.x = x\_KPlusOne;

solution.y = y\_KPlusOne;

return solution;

}

*Результаты*

Найденное решение:

Невязка:

Для достижения заданной точности потребовалось итераций.

***2.Метод Ньютона***

*Алгоритм*

Используя , построим последовательность приближений по следующему правилу:

,

Заканчиваем итерационный процесс, если:

Для проверки выполнения критерия останова используем следующую норму:

Частные производные функций и :

*.*

Формула для вычисления Якобиана:

Невязка найденного решения равна .

Для последующего анализа скорости сходимости храним количество итераций в переменной *iterationsQuantity* и, соответственно, увеличиваем переменную после каждой итераций.

*Листинг*

Функция для нахождения :

double g(double x, double y) { return (y - 4 \* x - 1); }

Функции для нахождения :

double partialDerivativeDfDx(double x, double y) { return y; }

double partialDerivativeDfDy(double x, double y) { return x + 1; }

double partialDerivativeDgDx(double x, double y) { return -4; }

double partialDerivativeDgDy(double x, double y) { return 1; }

Функция для нахождения :

double jacobian(double x, double y) {

return (partialDerivativeDfDx(x, y) \* partialDerivativeDgDy(x, y) -

partialDerivativeDgDx(x, y) \* partialDerivativeDfDy(x, y));

}

Функция для нахождения решения СНУ методом Ньютона:

Point NewtonMethod(double x\_null, double y\_null, double eps) {

double x\_K = x\_null;

double y\_K = y\_null;

double x\_KPlusOne = x\_K - ((1 / jacobian(x\_K, y\_K)) \*

(partialDerivativeDgDy(x\_K, y\_K) \* f(x\_K, y\_K) -

partialDerivativeDfDy(x\_K, y\_K) \* g(x\_K, y\_K)));

double y\_KPlusOne = y\_K - ((1 / jacobian(x\_K, y\_K)) \*

(- partialDerivativeDgDx(x\_K, y\_K) \* f(x\_K, y\_K) +

partialDerivativeDfDx(x\_K, y\_K) \* g(x\_K, y\_K)));

int iterationsQuantity = 1;

while (norm(x\_KPlusOne, y\_KPlusOne, x\_K, y\_K) >= eps) {

x\_K = x\_KPlusOne;

y\_K = y\_KPlusOne;

x\_KPlusOne = x\_K - (1 / jacobian(x\_K, y\_K)) \*

(partialDerivativeDgDy(x\_K, y\_K) \* f(x\_K, y\_K) -

partialDerivativeDfDy(x\_K, y\_K) \* g(x\_K, y\_K));

y\_KPlusOne = y\_K - (1 / jacobian(x\_K, y\_K)) \*

(- partialDerivativeDgDx(x\_K, y\_K) \* f(x\_K, y\_K) +

partialDerivativeDfDx(x\_K, y\_K) \* g(x\_K, y\_K));

iterationsQuantity++;

}

Point solution = new Point();

solution.x = x\_KPlusOne;

solution.y = y\_KPlusOne;

return solution;

}

*Результаты*

Найденное решение:

Невязка:

Для достижения заданной точности потребовалось итераций.

**Сравнение методов**

Примененные методы имеют разные теоретические скорости сходимости: метод Гаусса-Зейделя с реализацией по МПИ – линейную, метод Ньютона – квадратичную. На практике метод Ньютона также оказался быстрее (на нахождение решения данным методом понадобилось всего 5 итераций, тогда как количество итераций при решении методом Гаусса-Зейделя – 27). Такое большое количество итераций при использовании метода

Гаусса-Зейделя можно объяснить тем, что канонические виды не содержат в себе тех переменных относительно которых решаются уравнения (1) и (2) и, вообще говоря, данные уравнения можно сразу решить, подставив переменные и соответственно, не применяя метод простой итерации.

По точности более удачным оказался метод Ньютона (невязка при использовании метода Ньютона, при использовании метода Гаусса-Зейделя ), объяснить это можно скоростью сходимости (аналогично рассуждениям в лабораторной работе №1).