**Методы численного анализа**

**Отчёт по лабораторной работе №1**

**Вариант 13**

**Студент**

Агинский Антон Викторович

**Преподаватель**

Будник Анатолий Михайлович

ФПМИ БГУ  
2022

**Постановка задачи**

Найти корень нелинейного уравнения применяя следующие методы:

1. Метод простой итерации

2. Метод Ньютона

3. Метод секущих

Необходимо:

А) Отделить один корень уравнения графическим методом.

Б) Выбрать начальное приближение , исходя из выполнения условий теоремы о сходимости метода простой итерации.

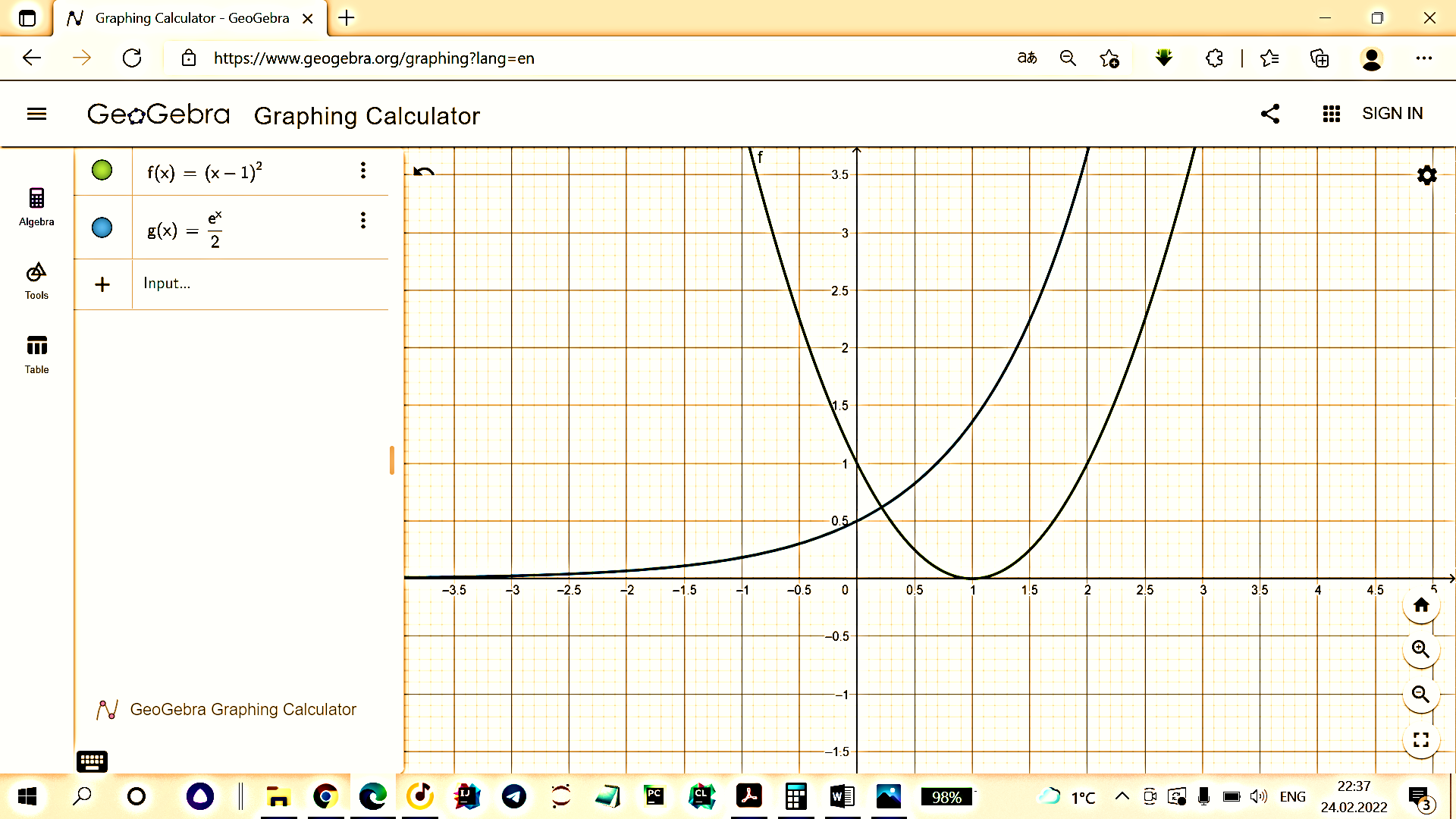
В) Используя начальное приближение, найти тремя вышеуказанными методами решение данного нелинейного уравнения с точностью . Критерий останова итерационного процесса: .

Г) Сравнить методы по скорости сходимости и точности.

**Отделение корней**

Графически отделим корень исходного уравнения, представив

в виде и найдя точку пересечения графиков и .



Отсюда видим, что точка пересечения графиков . Подтвердим графические соображения. Имеем :

, ,

,

Из этого следует, что на найденном отрезке уравнение имеет единственный корень.

**Выбор начального приближения**

Приведем уравнение к виду , удобному для итерации:

*Теорема (о сходимости метода простой итерации).*

Если:

1) определена и непрерывна в области ,

2) в этой области удовлетворяет условию Липшица

с константой ,

на практике условие Липшица часто заменяется на равносильное условие:

,

3) Числа связаны соотношением , где ,

то:

1) последовательность приближений по методу простой итерации с исходным приближением может быть построена,

2) в указанной области уравнение имеет единственное решение,

3) построенная последовательность приближений с ростом сходится к этому решению,

4) скорость сходимости характеризуется неравенством:

.

Проверим условия о сходимости метода простой итерации, выбрав в качестве начального приближения :

1) Возьмем . Функция определена и непрерывна в данной области,

2) возьмём ,

3) , возьмём .

Значит, при метод простой итерации сходится.

**Нахождение решения**

***1. Метод простой итерации***

*Алгоритм*

Используя, построим последовательность приближений по следующему правилу:

Заканчиваем итерационный процесс, если:

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 f(double x) {  
 return (Math.*pow*((x - 1), 2) - (0.5 \* Math.*exp*(x)));  
}

Функция для нахождения :

double phi(double x) {

return (Math.pow(x, 2) / 2) + 0.5 - (Math.exp(x) / 4);

}

Функция для нахождения решения НУ методом простой итерации:

double fixedPointIteration(double x\_null, double eps) {

double x\_K = x\_null;

double x\_KPlusOne = phi(x\_K);

int iterationsQuantity = 1;

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

x\_K = x\_KPlusOne;

x\_KPlusOne = phi(x\_K);

iterationsQuantity++;

}

return x\_KPlusOne;

}

*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:

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 derivativeOfF(double x) {

return ((2 \* x) - 2 - 0.5 \* Math.exp(x));

}

Функция для нахождения решения НУ методом Ньютона:

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

double x\_K = x\_null;

double x\_KPlusOne = x\_K - (f(x\_K)/derivativeOfF(x\_K));

int iterationsQuantity = 1;

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

x\_K = x\_KPlusOne;

x\_KPlusOne = x\_K - (f(x\_K)/derivativeOfF(x\_K));

iterationsQuantity++;

}

return x\_KPlusOne;

}

*Результаты*

Найденное решение:

Невязка:

Для достижения заданной точности потребовалось итерации.

***3.Метод секущих***

*Алгоритм*

Используя, а также найденный методом простой итерации , построим последовательность приближений по следующему правилу:

Заканчиваем итерационный процесс, если:

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 решения НУ методом секущих:

double secantMethod(double x\_null, double x\_one, double eps) {

double x\_KMinusOne = x\_null;

double x\_K = x\_one;

double x\_KPlusOne = x\_K - f(x\_K) \* ((x\_K - x\_KMinusOne) / (f(x\_K) - f(x\_KMinusOne)));

int iterationsQuantity = 1;

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

x\_KMinusOne = x\_K;

x\_K = x\_KPlusOne;

x\_KPlusOne = x\_K - f(x\_K) \* ((x\_K - x\_KMinusOne) / (f(x\_K) - f(x\_KMinusOne)));

iterationsQuantity++;

}

return x\_KPlusOne;

}

*Результаты*

Найденное решение:

Невязка:

Для достижения заданной точности потребовалось итерации.

**Сравнение методов**

Все три разобранных метода имеют разные скорости сходимости: метод простой итерации – линейную, метод Ньютона – квадратичную, порядок сходимости метода секущих равен золотому сечению. Таким образом, мы можем однозначно сказать, какой метод теоретически сходится быстрее. Однако на практике МПИ сошёлся с такой же скоростью, с какой и должен был теоретически, а методы Ньютона и секущих показали одинаковую скорость сходимости. Это можно объяснить выбором . Эмпирическим путём проверено, что при другом приближении метод секущих можем сходится медленнее, чем метод Ньютона.

Самым точным из трёх оказался метод Ньютона. Разница между и величиной невязки объясняется тем, что при квадратичной скорости сходимости с каждой итерацией количество значащих цифр увеличивается примерно в два раза. То есть перед последней итерацией порядок невязки был равен , однако критерий останова ещё не выполнялся. И таким образом после следующей(последней) итерации порядок невязки уже стал . При анализе величин невязок других методов можно рассуждать аналогично, основываясь на скоростях сходимости каждого метода из предыдущего абзаца.

Методом Ньютона было получено самое точное решение, причём за наименьшее число операций. Однако данный метод не лишён недостатков. Например, если вычисление производной затруднено, мы можем использовать метод секущих, который является «приближенным» методом Ньютона, но это всё равно не избавляет нас от всех проблем, связанных с условиями применения метода: 1) метод может не сойтись, если начальное приближение недостаточно близко к решению, 2) если производная не непрерывна в точке корня, то метод может расходиться в любой окрестности корня и 3) другие проблемы, из-за которых метод может расходиться, либо сходиться со скоростью, которая ниже, чем квадратичная. Если уравнение не может быть решено методом Ньютона, то следует использовать МПИ, для которого существует универсальный способ построения . Скорость сходимости метода простой итерации может быть увеличена использованием, например, метода Стеффенсена, который является модификацией МПИ.