**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

**Кафедра информационных систем управления**

**Отчет**

**По предмету «Методы Вычислений»**

Выполнил студент группы № 15

Кражевский Алексей Игоревич

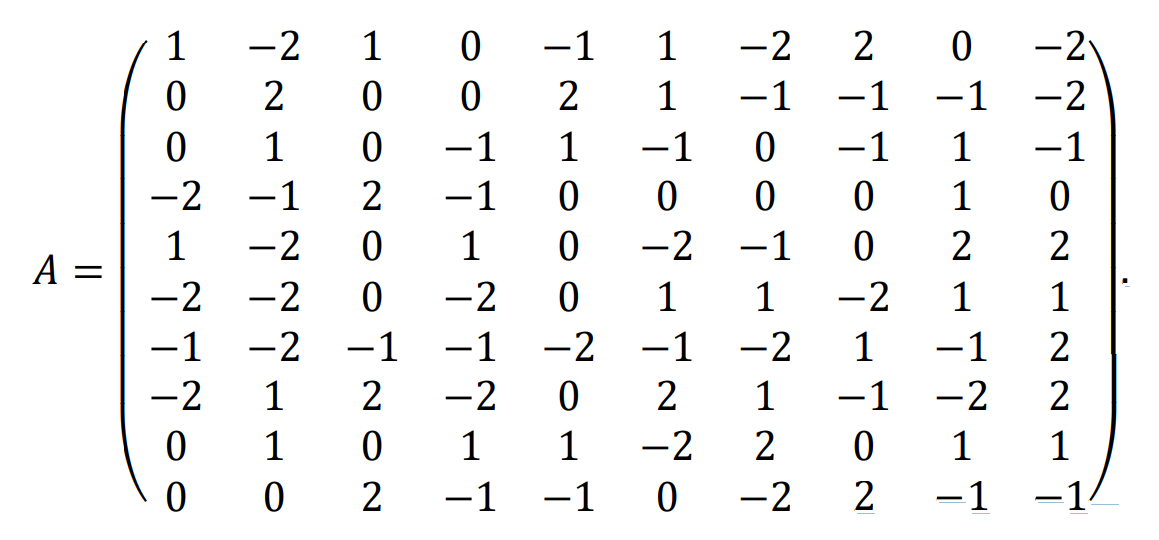
# Минск 2022

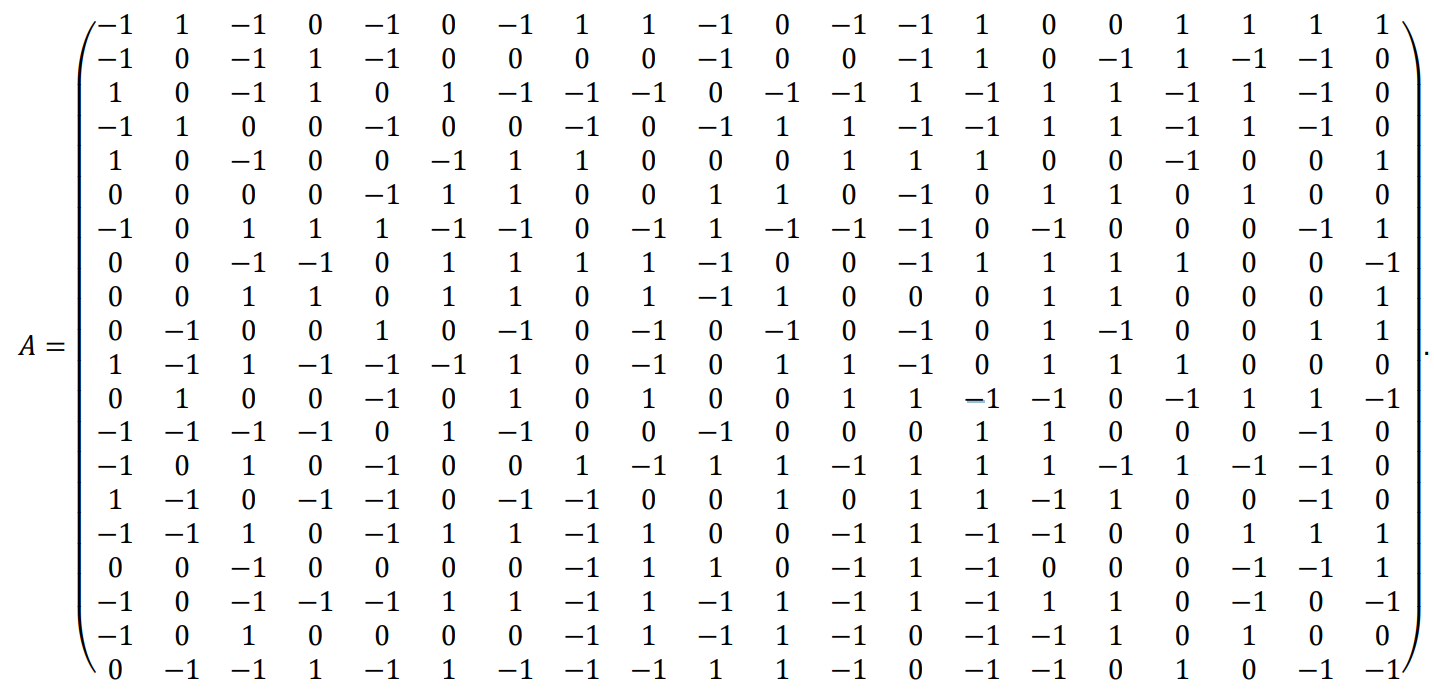
**Лабораторная работа №2**

**Решение проблемы собственных значений. Численное решение нелинейных уравнений.**

**Условие лабораторной:**

*Матрицы для заданий 1-2:*





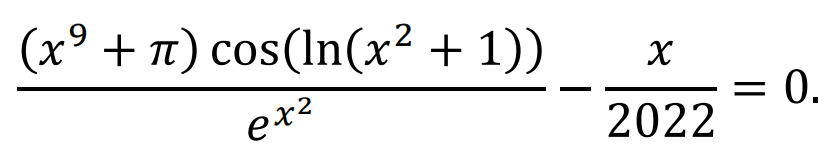
*Пунты выполнения лабораторной работы:*

1. Написать программу, которая находит максимальное по модулю собственное значение (пару максимальных по модулю собственных значений) и соответствующий ему (им) собственный вектор (собственные векторы) матрицы 𝐴 с максимально возможной точностью (в пределах обычных double чисел). Применить программу к следующим ниже входным данным. В отчете подробно изложить способ определения случая и критерия остановки итераций. Проведите экспериментальное исследование скорости работы вашей программы в зависимости от размерности матрицы, используя для тестов матрицу со случайными числами. Постройте график зависимости времени работы от размерности. Вычислите асимптотическую сложность реализованного вами алгоритма.

2. Написать программу, реализующую 𝑄𝑅-алгоритм нахождения всех собственных значений матрицы 𝐴 с максимально возможной точностью. Применить программу к матрицам из первого задания. Проведите экспериментальное исследование скорости работы вашей программы в зависимости от размерности матрицы, используя для тестов матрицу со случайными числами. Постройте график зависимости времени работы от размерности. Вычислите асимптотическую сложность реализованного вами

алгоритма.

3. Дано нелинейное уравнение:



Отделить все его корни. Обосновать (не обязательно доказать строго) единственность каждого корня на отрезке, отсутствие других корней. Методом бисекции сузить отрезки отделенности корней до размера не более 10−4. Методом Ньютона найти все корни с максимально возможной точностью. В отчет включить количество итераций обоих методов.

**Результаты выполнения лабораторной работы**

Программа реализована на языке c++. Реализованы специальные методы для пунктов задания (реализации алгоритмов) и методы, нужные для реализации алгоритмов.

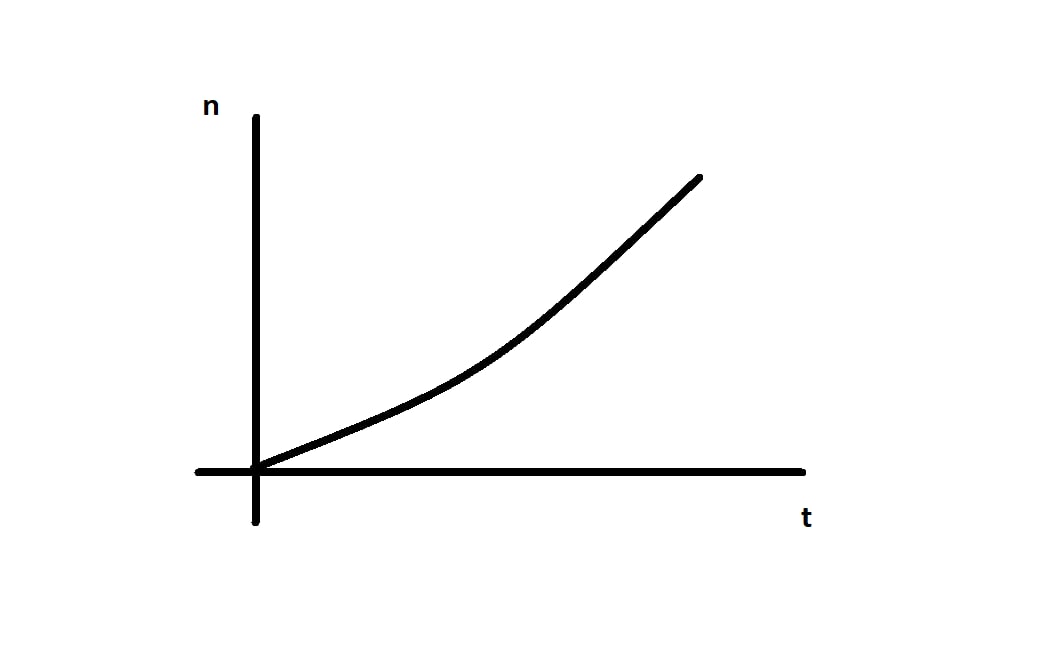
*Пункт 1*

Критерий остановки степенного метода:



В качестве эпсилон я передавал значение 1e-6. На каждой итерации идет проверка всех случаев, так как нельзя сразу определить нужный. Для каждого случая проверялся критерий остановки и выводилась соответственно высчитанная по нужным формулам пара собственных значений и собственных векторов. Для реализации алгоритма я использовал структуру, которая хранит матрицу u, y и несколько этих же матриц с предыдущих итераций для использования в формулах.

График зависимости времени от размерности матрицы должен быть примерно таким:



В силу своих кривых рук и глаз у меня не получилось довести данный метод до ума, при его использовании ничего не сходится в случае парных комплексных собственных значений. В остальных случаях все работает хорошо. Однако это не особо помогло нормально протестировать метод.

*Пункт 2*

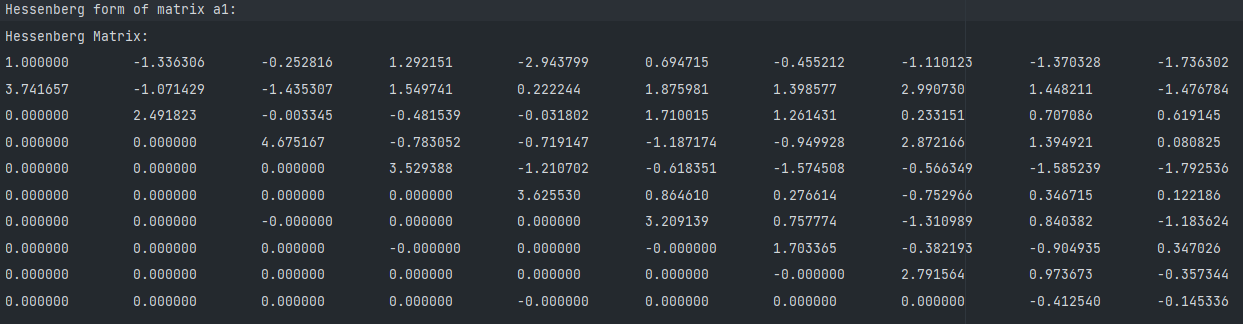
Для получения оптимального времени выполнения QR-алгоритма, сначала приведем матрицы к форме Хессенберга. Для этого реализован специальный метод. После приведения к форме Хессенберга передаем полученную матрицу в метод, реализовывающий QR-алгоритм. Для выполнения QR-алгоритма используется структура QR\_Matrix, которая хранит матрицы R, T, RQ для следующей итерации алгоритма. После всех итераций запускается метод изъятия собственных значений из полученной RQ матрицы.

При работе с первой матрицей я получил корректные собственные значения. Время выполнения по неизвестной мне причине довольно большое. Для первой матрицы я установил количество итераций равное 1000. Для данной матрицы этого вполне хватит. Время выполнения:



Также в отдельный файл выводится представление этой матрицы в форме Хессенберга и также выводится сама матрица после завершения всех итераций.

Например, форма Хессенберга для А1:

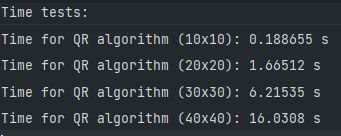
****

Для второй матрицы все действия остаются такими же, за исключением количества итераций. Я поставил 7000, так как при использовании, например, 1000 итераций появляются ненужные комплексные корни вместо действительных (вероятно, из-за маленьких чисел, которые остаются под диагональю). Время на выполнение также немаленькое:



Например, при использовании QR-алгоритма на 1000 итераций на второй матрице время выполнения было около 20 секунд. Собственные значения все также находятся корректно. Скриншоты формы Хессенберга или матрицы после алгоритма не прикреплял из-за большого размера.

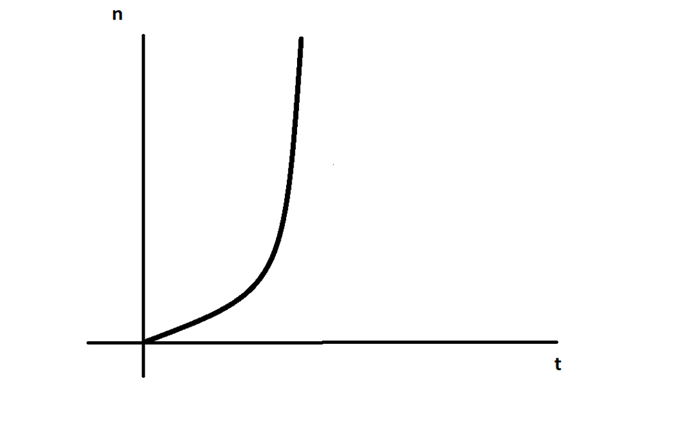
Тесты скорости при количестве итераций = 100 показали следующие результаты:



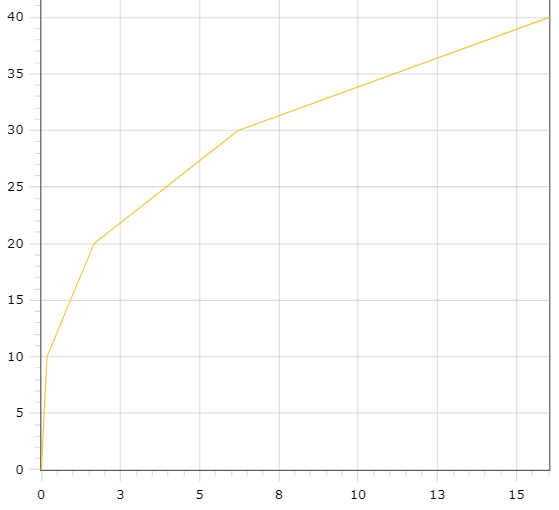
Как видно, скорость работы при 100 итерациях не высокая, но и не маленькая. Однако в таком случае максимальная точность не гарантирована, особенно на больших матрицах.

Матрицы для тестов заполнял случайными double числами.

Если построить график зависимости времени работы от размерности матрицы, то он должен быть примерно таким



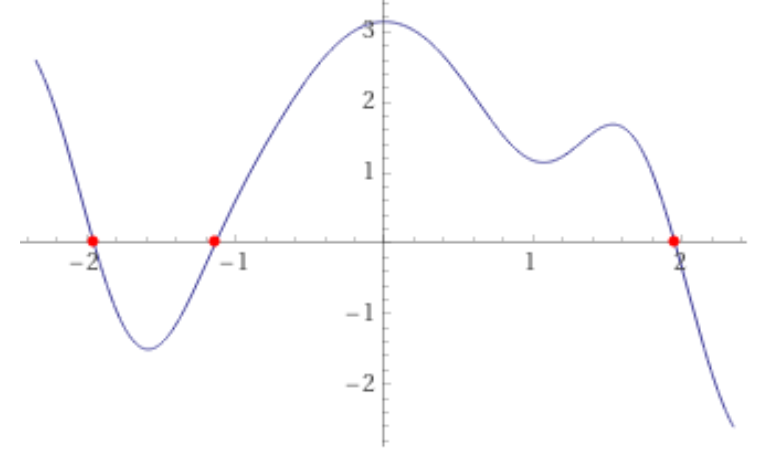
Однако, если построить график по точкам, полученным при выполнении работы получается нечто такое



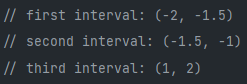
Т.е., скорость сначала не сильно увеличивается, но с увеличением размерности матрицы начинает существенно возрастать. Асимптотика алгоритма при использовании формы Хессенберга должна составлять около О(n^2), однако что-то мне подсказывает, что у меня асимптотика на уровне О(n^3) (при большом количестве итераций это становится сильнее заметно).

*Пункт 3*

Для начала отделим корни уравнения. Попробуем графически:



У нас получилось 3 корня. Из графика мы примерно видим в каких границах находятся эти корни. Также мы можем подставить значения справа и слева от полученных точек в фунцию и убедится, что у них разные знаки, а значит они находятся по разные стороны от оси ОХ. Единственность каждого корня можно видеть на графике, но также можно посчитать производные в разных точках отрезков и убедится в этом. Отделим промежутки:



Теперь сузим эти отрезки методом бисекций до точности 10−4. Используем алгоритм из Фалейчика и получаем следующие отрезки:







Критерий сходимости был следующий:



Метод бисекций сработал за 12-13 итераций для разных корней, что неплохо и незатратно по времени:







Для метода Ньютона я взял за х левую границу полученных промежутков. Точность я вводил 1e-16, что довольно много, но метод Ньютона при критерии остановки



сработал за 3-4 итерации и довольно быстро нашел максимально точное решение нелинейного уравнения.

