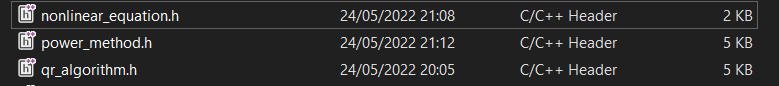
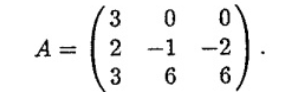
Основной код лабораторной находиться в заголовочных файлах. 

1. Я искал сразу два доминантных значения, чтобы определить случай. Для этого хранил три вектора приближенных значений x1, x2, x3. И дальше как объяснял Иван Бондарь решалась задача наименьших квадратов СЛАУ: [x1, x2]\*[c] = [-x3] методом нормальных уравнений (умножением на транспонированную матрицу). После чего по корням уравнения с коэффициентами [c] находились собственные значения. Собственный вектор это x3 (или [х3 – λx2, x3- λx1] в случае если λ две штуки). При этом в случае если λ1 = λ2 находиться только один собственный вектор (при тестах в этом случае находило 2 комплексно сопряженных значения с маленькой мнимой частью (λ ~1е-6i)). Пример: 

Итерации останавливались когда (или кол-во итераций превышало 400). Ошибка это 1e-7, т.к. с большой точностью плохо решается СЛАУ и везде выдавало –бесконечность. Я выводил все итерации в файл, т.к. в конце опять же могло появляется деление на 0 или неточности, так что лучше глазами перепроверять последние пару итераций.

В цикле решается СЛАУ за и умножаются матрицы за столько же. Т.к. итераций в цикле всего 400 максимум на асимптотику это не сильно повлияет, но у достаточного большая константа из-за того что операции за выполняются по многу раз.

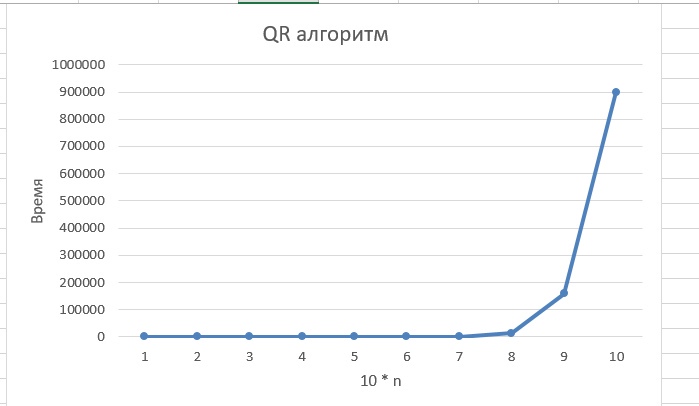
График примерно соответствует кубу. Скачки вызваны скорее всего разным кол-вом итераций у матриц. 

2. К форме Хессенберга приводил “лениво” методом отражений. Лениво, т.к. матрицы отражений просто умножаются с двух сторон за куб, без оптимизации. Из-за чего приведение к форме Хессенберга происходит за . Что значительно замедляет алгоритм.

Но QR разложение на итерациях сделано быстро с помощью метода вращений. Вращение происходит не умножением на матрицу, а умножением двух строк или столбцов на косинусы и синусы за O(n). Матрицы вращения хранятся как вектор углов вращения и умножаются слева а потом справа. Таким образом я сначала вращаю ряды, а потом на те же углы колонны. Если не брать в расчет кол-во итераций, то QR (формально QR декомпозиция, с учетом итераций) должен выполняться за O(n\*n). НО общая сложность с Хессенбергом  **,** что достаточно плохо.

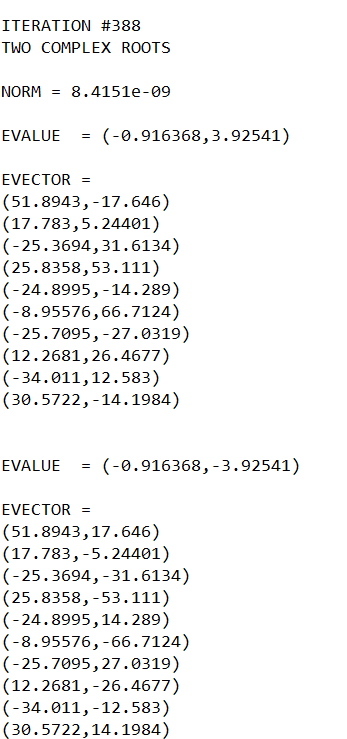
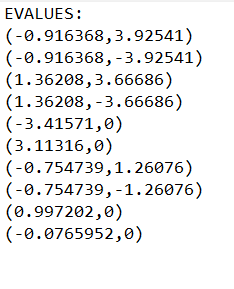
Итерации останавливались, когда разница между соответствующими собственными значениями становилась меньше погрешности. Тут нормально проходит точность 1e-15.

График примерно соответствует четвертой степени (гиперкубу?) (матрицы 100 на 100 считаются уже настолько долго, что надоедает ждать)



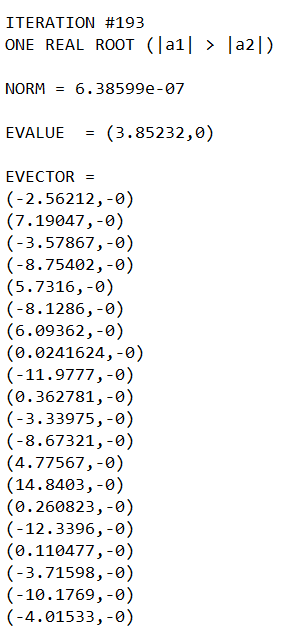
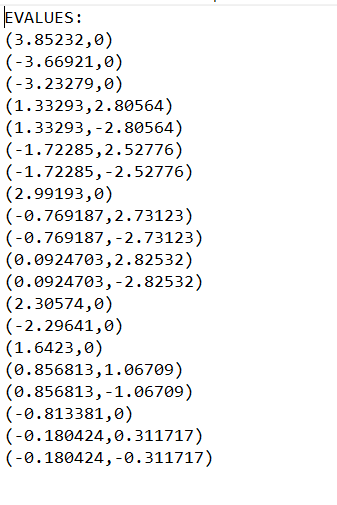
*Результаты применения программы к матрицам из условия*

*Матрица 10х10*

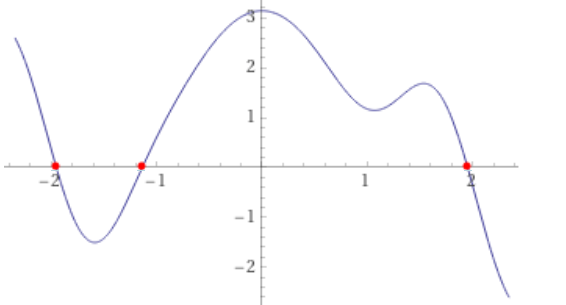
 **

*---------------------------------*

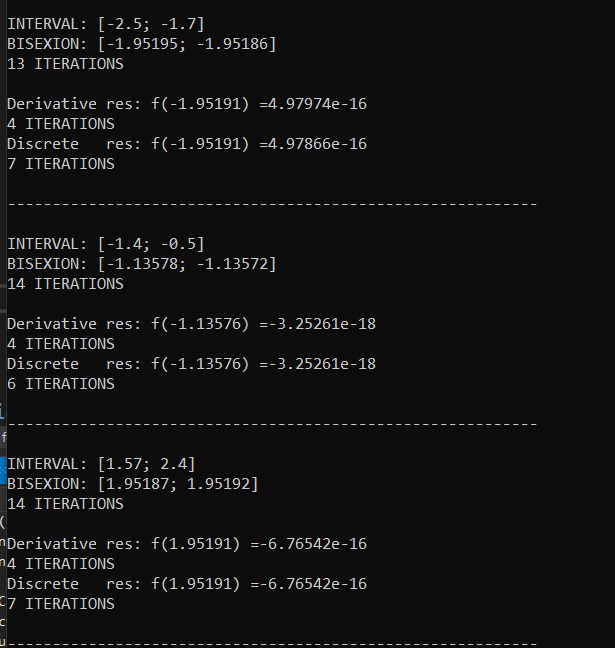
*Матрица 20х20*

** **

*3. Алгоритм простой. Сделал две вариации ньютона (дискретного и с точной производной). Зависимость точности дискретного от дельты не тестировал.*

*Основная проблема — это найти промежутки. По графику их можно очевидно найти.* **

То что на бесконечности не будет корней можно доказать строго, найти промежутки знакопостоянсква, знаки производной. Но в задании этого не требуется и основная вычислительная часть сделана.

*Результат:*