Вычислительные методы алгебры. Лабараторная работа 2

Константин Мехович

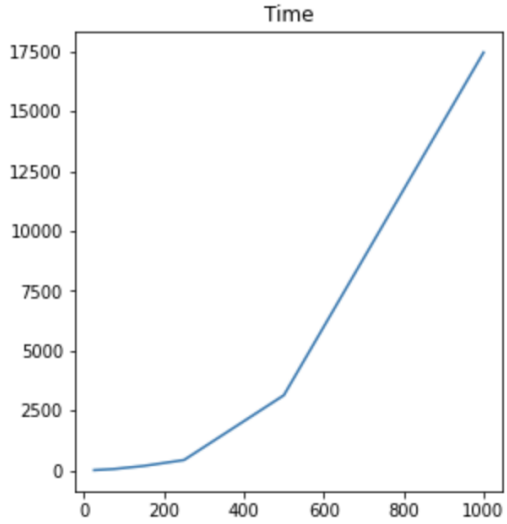
17 декабря 2021 г.

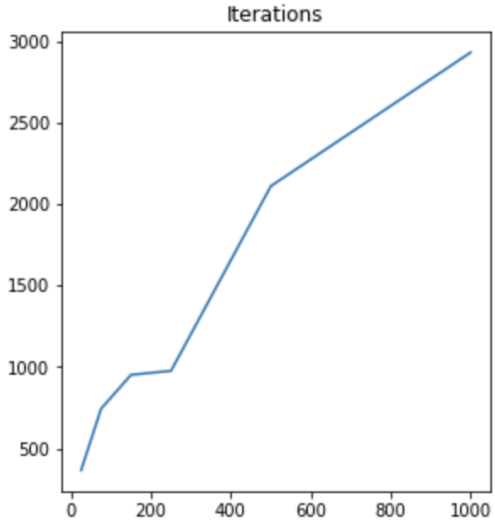
1 Степенной метод

Определение случая было реализовано в моем методе так: давайте считать на каждой итерации как будто собственное значение подходит под тривиальный случай, а также будет решать задачу минимальных квадратов для нахождения собственного значения если оно комплексное или имеет вид $\pm\lambda$. Перед каждой итерацией мы считаем норму невязки с вектором из тривиального случая. Если она достаточна мала - максимальное по модулю собственное значение единственно и мы его нашли плюс нашли вектор к нему. Как решаем для комплексных: мы просто закончим итерации, если оба наших собственных значения мало отличаются от подсчитанных на предыдущей итерации. Чтобы найти собственный вектор к комплексным собственным значениям требуется решить простую систему:

$$\begin{cases} u^k = \vec{v_1} + \vec{v_2}, \\ Au^k = \lambda_1\vec{v_1} + \lambda_2\vec{v_2}, \end{cases}$$

В этой системе две неизвестных - два вектора, которые легко выражаются.
Асимптотическая сложность: вычислим сложность одной итерации. Для тривиального случая оказывается, что сложность линейная. Для второго случая выходит так, что требуется решить задачу минимальных квадратов с матрицей, у которой одна из размерностей всегда 2. Если считать с учетом этого, то сложность - $4n$. Но асимптотическая сложность $O(n)$. В итоге у нас получится сложность алгоритма $O(c \cdot n)$, где c - количество осуществленных итераций.
Для примеров решил. Дабы не загрязнять отчет, ответы лежат в $task1/ans1.txt$ & $task1/ans2.txt$.
Пример 1: 383 итерации, 12 миллисекунд.
Пример 2: 80 итераций, 5 миллисекунд.
В рандомных матрицах числа от -10 до 10. Для более точного результата я запускал на каждом размере 3 раза разные случайные матрицы, а потом посчитал среднее и составил графики.





2 Метод Данилевского

Моя реализация метода Данилевского - с выбором главного элемента. Чтобы найти корни я взял отрезок [-10000; 10000], разделил его на $2 \cdot 10^6$ и получил шаг. Проитерировался с этим шагом по всему отрезку запуская из каждой точки поиск корня методом Ньютона. Из этого взял лишь уникальные значения - это был ответ.
На примерах такое решение сработало и нашло все действительные корни и вектора. Дабы не загрязнять отчет, ответы лежат в $task2/ans1.txt$ & $task2/ans2.txt$.

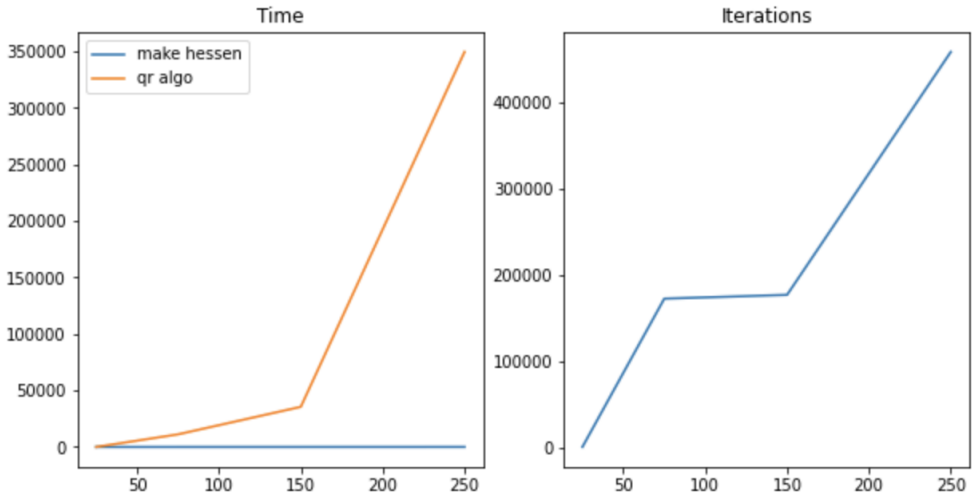
Время работы зависит от того, на каком по размеру отрезке вам надо найти корни, с каким шагом будете идти, а также не стоит забывать, что элементарные преобразования для приведения к форме Фробениуса работают за n^3.
Вектора вообще можно просто посчитать, если запоминать какие преобразования мы делали над столбцами в переводе к форме Фробениуса: нужно просто применить их же, но в обратном порядке к собственному вектору матрицы Фробениуса. Доказательство этого следует из:

$$A'x' = \lambda x' \Leftrightarrow S^{-}1ASx' = \lambda x' \Leftrightarrow ASx \ = \lambda Sx'$$

Где A' - матрица формы Фробениуса, S - невырожденное матрица / преобразование подобия. x' - собственный вектор A', то $x = Sx'$ - собственный вектор A. Преобразования бывают - свап столбцов, деление на число, отнимание от столбца другого столбца умноженного на число. Преобразований будет не больше $n + n + n^2$. Короче это все быстро, очень круто, даже без перемножения матриц перехода.

3 QR-алгоритм

Нечего сказать про реализацию, так как мы просто строим хессенберговую, с учетом того, чтобы спектр сохранился. Потом выполняем итерации, вращаем, перемножаем, пытаемся выделить корни. Проблема возникает когда корни не выделяются из матрицы долго. Это может происходить из-за того что их модули близки. Причем зафейлить могут всего лишь две пары комплексных корней, то есть быть может, из 200-порядка матрицы если бы убрали эти корни, то мы бы посчитали весьма быстро, а так мы все пытаемся уменьшить одно число под диагональю, чтобы разделить эти корни. Конечно, проигнорить одно число под диагональю нельзя, потому что оно повлияет на другие корни. Короче, можем сгенерить какую-нить рандомную матрицу, а у нее случайно окажутся рядом СЗ и будем считать этим алгосом очень долго...
Асимптотика - $O(max(n^3, c \cdot n^2))$, где c - кол-во итераций алгоса. n^2 из того что делаем преобразования вращения. n^3 из построения хессенберговой матрицы (отражения).
Для примеров решил. Дабы не загрязнять отчет, ответы лежат в $task3/ans1.txt$ & $task3/ans2.txt$.
В матрицах числа от -100 до 100.


Time
make hessen
qr algo
350000
300000
250000
200000
150000
100000
50000
0
50
100
150
200
250
Iterations
400000
300000
200000
100000
0
50
100
150
200
250