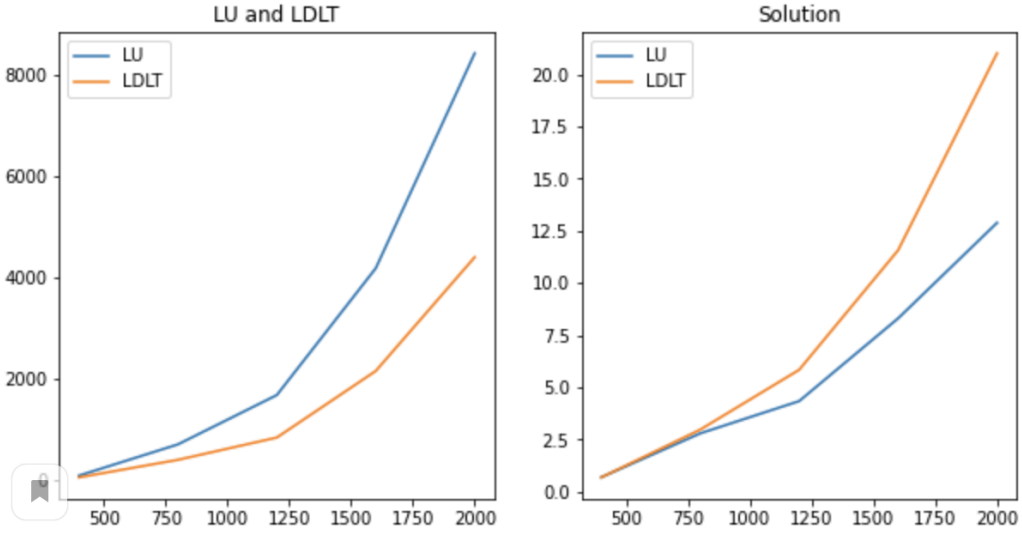
Вычислительные методы алгебры. Лабараторная работа 1

Константин Мехович

7 ноября 2021 г.

1 LU & LDLT

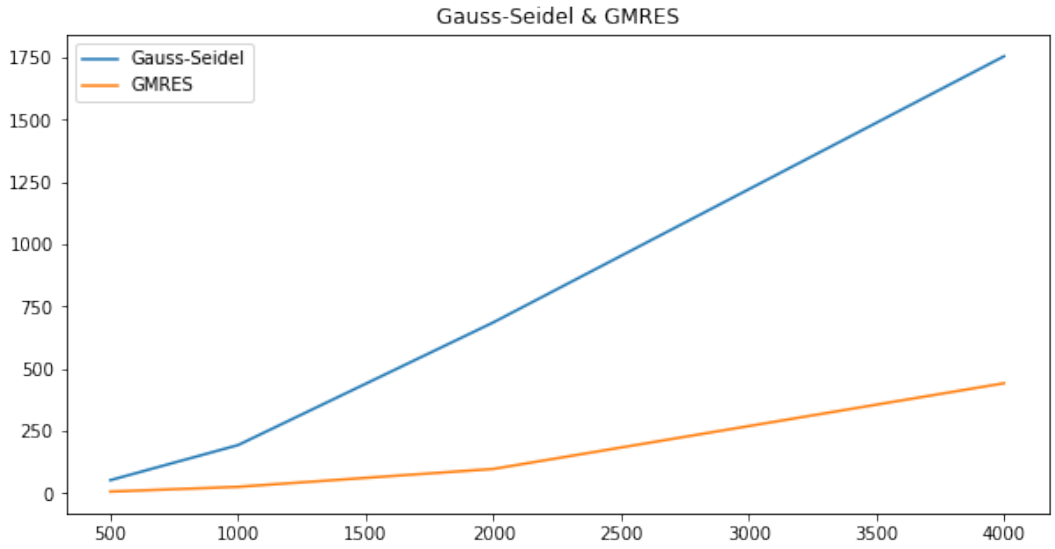
Построение LU и LDLT отличается ровно в 2 раза: LDLT быстрее. Что касается решения системы, то тут есть нюанс: все асимптотически сложности одинаковые, но LDLT работает с половиной матрицы, тогда получаем то, что у нее в одном из ходов (в зависимости от реализации) мы идем по столбцу и кэш это тормозит в отличие от LU. Мы можем это решить, добавив преобразование половинно-валидной матрицы LDLT в фулл-валидную (т. к. она симметричная), сказав что это условие матрицы LDLT и запихнуть этот процесс в построения LDLT, и решать таким образом, чтобы скорость решений LU и LDLT не отличалась. Но, на мой взгляд, интересней рассматривать случай, когда мы не можем таким образом скопировать, потому выходит так:



То есть в действительности решение LDLT будет медленней, причем эта зависимость исходит от размерности матрицы.
Сгенерированные числа в матрице: от -1 до 1 дробные через Mersenne Twister Generator (aka mt19937). Также, с учетом моих реализаций, я смог запустить код на 5000. LU = 131 sec, LDLT = 71 sec. В построении сохраняется константа 2. solve lu = 87 msec, solve ldlt = 307 (!) msec. Из этого видим, что время решения LDLT уже весьма больше времени решения LU. Из этого можно сделать вывод, что если нас просят решить СЛАУ и дают на выбор готовые матрицы LU и LDLT, в случае если LDLT задана так, что только ее половина валидна, т. е. L располагается в нижнем треугольнике, и L^T там же располагается, только ее нужно читать "транспонированно" , то выбрать нужно LU, оно будет быстрее из-за кэша.

2 Гаусс-Зейдель & GMRES

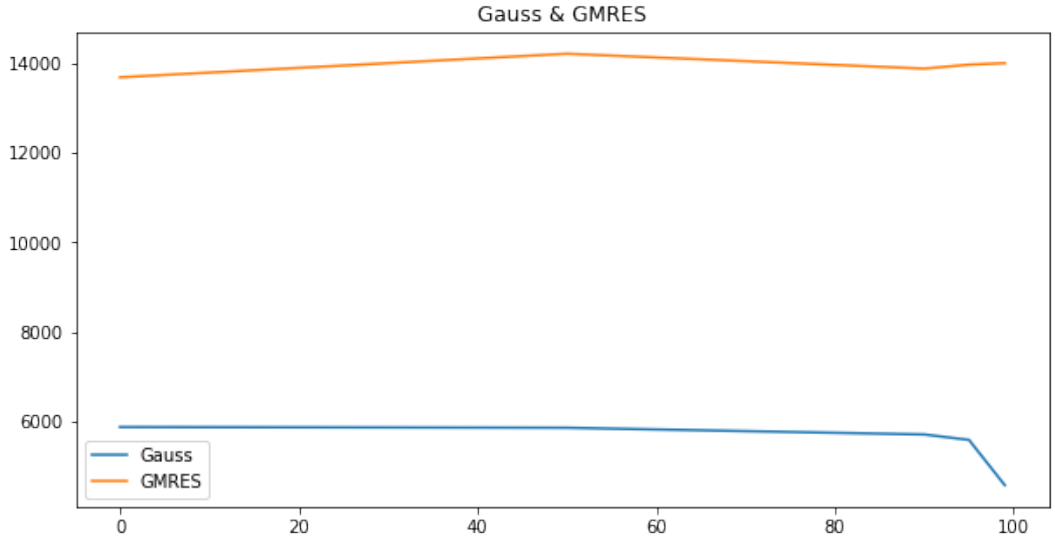
Сходимость в методе Гаусса-Зейделя обуславливается тем, что исходная матрица имеет строгое диагональное преобладание, следовательно итерационный процесс будет сходиться (Лемма 2.3 конспекта ПИ).
Количество итераций у Гаусса-Зейделя: 5-6-6-5. У GMRES: 2-3-3-4.



Получилось так, что Гаусс-Зейдель уже на 4000 медленней GMRES в 3,5 раза. Конечно, тут решает насколько хорошо соптимизирован GMRES. Например, важнейшая оптимизация, это не считать при каждой итерации исходный вектор x и невязку. Критерия остановки в алгоритме Арнольди достаточно, чтобы увидеть, что ответ лучше в разумное количество раз мы уже не улучшим.
Еще оптимизация - храним матрицу поворотов как два вектора, потому что значений в матрице суперпозиций матриц поворота у нас не будет больше чем $2 \cdot n$.
А еще надо заметить, что в реализации проще хранить матрицу Q как массив векторов, нежели как свою структурку матрицы, потому что тогда мы можем легко вытаскивать вектор, когда мы считаем вектор z в алгоритме Арнольди, скалярное произведение, приравниваем новый вектор. Единственный раз, когда мы работаем с ней как с матрицей - восстанавливаем ответ x.
Взял начальный вектор $Q[0]$ как в обобщенном методе минимальных невязок - нормированный b.
P.S. Я там еще нормально так прокомментил код в *gmres.cpp* если что, но вроде ничего не упустил.

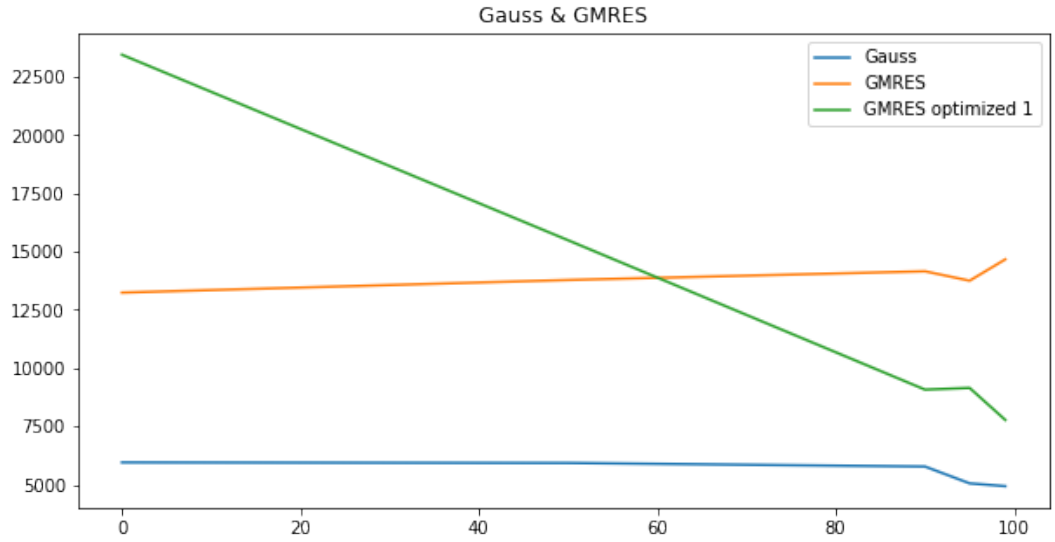
3 Разряженная матрица

Следует начать с того, что нам придется генерировать невырожденную матрицу и может показаться, что если генерировать в лоб, то могут быть проблемы. Однако, как мы помним в матрице $1000 \cdot 1000$ если взять 99% заменить нулями, то останется 10000 ненулевых. По определению определителя, это сумма 1000 умноженных элементов с разными строками и столбцами. Короче, мне показалось, что проблем не будет. Так и оказалось. Я генерил число в каждую ячейку и с вероятностью x ставил туда 0. Погрешность количества нулей получилась около 0.1%. Генерил числа дробные от -2 до 2, в столбец b от -10 до 10. Вышло так, что Гаусс был не так точен как GMRES (в 10 раз точнее), но GMRES сделал все 1000 итераций и в итоге не вышло выйти в евклидову норму невязки 10^{-10}, доходило до 10^{-9}. По времени вышло так:



Еще я проверил что будет, если от GMRES требовать такой же точности, как только что посчитал Гаусс. Типо может быть, что итераций окажется меньше и он будет быстрее, но это оказалось ложным утверждением.

Оптимизация 1. У нас много времени выйдет на перемножение матрицы на столбец в алгоритме Арнольди в разряженной матрице когда мы будем умножать огромное количество ноликов. Можно предпросчитать в матрице A ненулевые позиции и исходя из них уже считать произведение матрицы на вектор. Выйдет вот так:



Получится, что на плотных матрицах с "оптимизацией" работает медленней, что логично, но это нас не интересует, потому что мы можем считать по разному для матриц разной разряженности. Поэтому ответом будет график из $min\{GMRES, GMRES\ optimzed\}$.