# Лабораторная работа №1 по предмету Методы Вычислений

Автор: Томашевич Константин, Вариант 8.

## Часть 1

С помощью обычного rand() было бы довольно сложно и костыльно генерировать десятичные дроби з 13 знаками после запятой, RAND\_MAX же ~32000, да и rand() не такой уж и рандомный, как хотелось бы. Поэтому я решил использовать генератор случайных чисел Mersenne Twister, который используется в GMP. Код его я, конечно же, стырил, ибо конкретно он не часть лабы. Добивался наличия хотя бы 13 значащих цифр после запятой я таким образом:

1. gen <- random number in [-4, 4].
2. If fractional of gen \* 10^13 lower than 10^-4 then gen <- gen + 10^-13.
3. return gen.

## Часть 2

Пожалуй, наименее интересная часть лабораторной работы. Метод Гаусса-Жордана пишется довольно просто и топорно, но работает при этом чрезвычайно быстро :sarcasm:. Вот зачем нужно было выдумывать всякие гмресы с арнольди, если можно просто потратить всего в раз в 70 (в случае 256х256 матриц) больше времени и просто решить ? Впрочем, недавно учёные нашли Илью, который умеет считать за . Возможно, это и есть будущее алгебры, только Илья пока что отказывается делиться этим секретным алгоритмом.

Теперь по делу: число обусловленности для всех матриц колеблется в интервале [4.5; 5], что не так уж и плохо выглядит в общем случае, но вот если вспомнить, что , всё выглядит уже не так радужно. Но, к счастью, непреодолимое желание приблизиться к числу обусловленности испытывала только евклидова норма разницы реального решения и решения через МНК, с остальными как-то обошлось.

Заметим, что средний элемент матрицы таки совсем не ноль. Но мы же генерировали её рандомно! Нас обманули и мерсен твистер не работает? Таки не обманули, просто диагонали всё-таки заполнялись не рандомно (диагональное доминирование делали же), вот и не ноль.

## Часть 3

Наконец-то мы перешли к реальному решению СЛАУ. И вот у нас метод Гауса с выбором максимального элемента по всей матрице. По правде говоря, с написанием этого метода я запарился больше всего, ибо тут и порядок решений нужно в конце восстановить, и постоянно перекидывать строки и столбцы в процессе. Запарно и дебажить пришлось.

Благодаря тому, что матрицы мы, всё-таки, генерируем довольно хорошие для этого алгоритма, евклидова норма разности решений колеблется в промежутке , что довольно таки точно. Для эксперемента я попробовал отключить выбор максимального элемента по матрице, в результате чего максимальная норма поднялась до , то есть в 1000 раз! Тем самым я убедился, что такая противная операция, как выбор максимального элемента по матрице, не просто имеет право на существование, а ещё и даёт очень даже хороший результат. Но всё-таки насколько попаболь была её дебажить...

Теперь рассмотрим время выполнения, оно составляет в среднем 39мс на матрицах 256х256, что почти в 4.5 раза быстрее поиска обратной матрицы по Гаусу-Жордану. Но стоп, мы же выполняем почти те же операции! Да ещё и играем со строками и колонками (и шрифтами), строки-то можно поменять местами за О(1), а вот столбцы только за О(n). Чего тогда так быстро? А штука просто в том, что в Гаусе у нас справа один столбец b, а не целая матрица. То есть разница в скорости очевидна.

## Часть 4

После написания и дебага обычного Гауса, написать было уже не настолько сложно. Да, я думал, что LU надо тоже делать в выбором максимального элемента по матрице. Но после адского Гауса с той же болью уже не было так сложно.

Сначала поговорим про евклидову норму разности решений. Впрочем, про неё говорить, возможно, и не стоит, так как она до последней цифры совпадает с той же нормой у обычного Гауса. И удивляться тут нечему – так как мы делаем фактически то же самой, только «запоминаем» ходы Гауса в объединённую LU-матрицу (так-как свойства матриц L и U позволяют совместить их в одну для хранения) и перестановки строк и столбцов в матрицы и (которые в памяти можно хранить просто как вектора перестановки), потом просто повторяем все нужные действия с данным вектором b.

А вот что насчёт времени? строится за 38мс, на 1мс быстрее, чем решается СЛАУ па Гаусу. Почему? Но ведь при построении мы не делаем обратного хода. На решение СЛАУ с построенным разложением уходит всего 1мс, что довольно логично, ведь мы просто мучаем один столбец b. Никакой магии, конечно, не происходит, ведь 38+1=39мс – тот же Гаусс. Зато можно решать потом много СЛАУ на очень большой скорости, благодаря уже полученному разложению.

## Часть 5

## Часть 6

## Часть 7

## Часть 8

## Часть 9

## Часть 10