Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Дисциплина: «Вычислительная математика»

Лабораторная работа №1
Вариант: Метод Гаусса с выбором главного элемента

Выполнил: Кизилов Степан Александрович,
группа P32312
Преподаватель: Перл Ольга Вячеславовна

Санкт-Петербург, 2023 год

1 Описание метода

У нас есть матрица СЛАУ. На первой итерации мы ищем наибольший элемент в 1 столбце и начинаем искать с 1 строки. Пусть он находится в строке матрицы main_row и его значение равно main_elem. Теперь к каждой строке матрицы (кроме main_row) мы должны добавить строку main_row умноженную на коэффициент

$$c_j = -\frac{matrix_{j,1}}{main\_elem}$$

В таком случае прозойдёт чудо:

$$new\_matrix_{j,1} = matrix_{j,1} + c_j \cdot main\_elem = matrix_{j,1} + (-\frac{matrix_{j,1}}{main\_elem} \cdot main\_elem) =$$

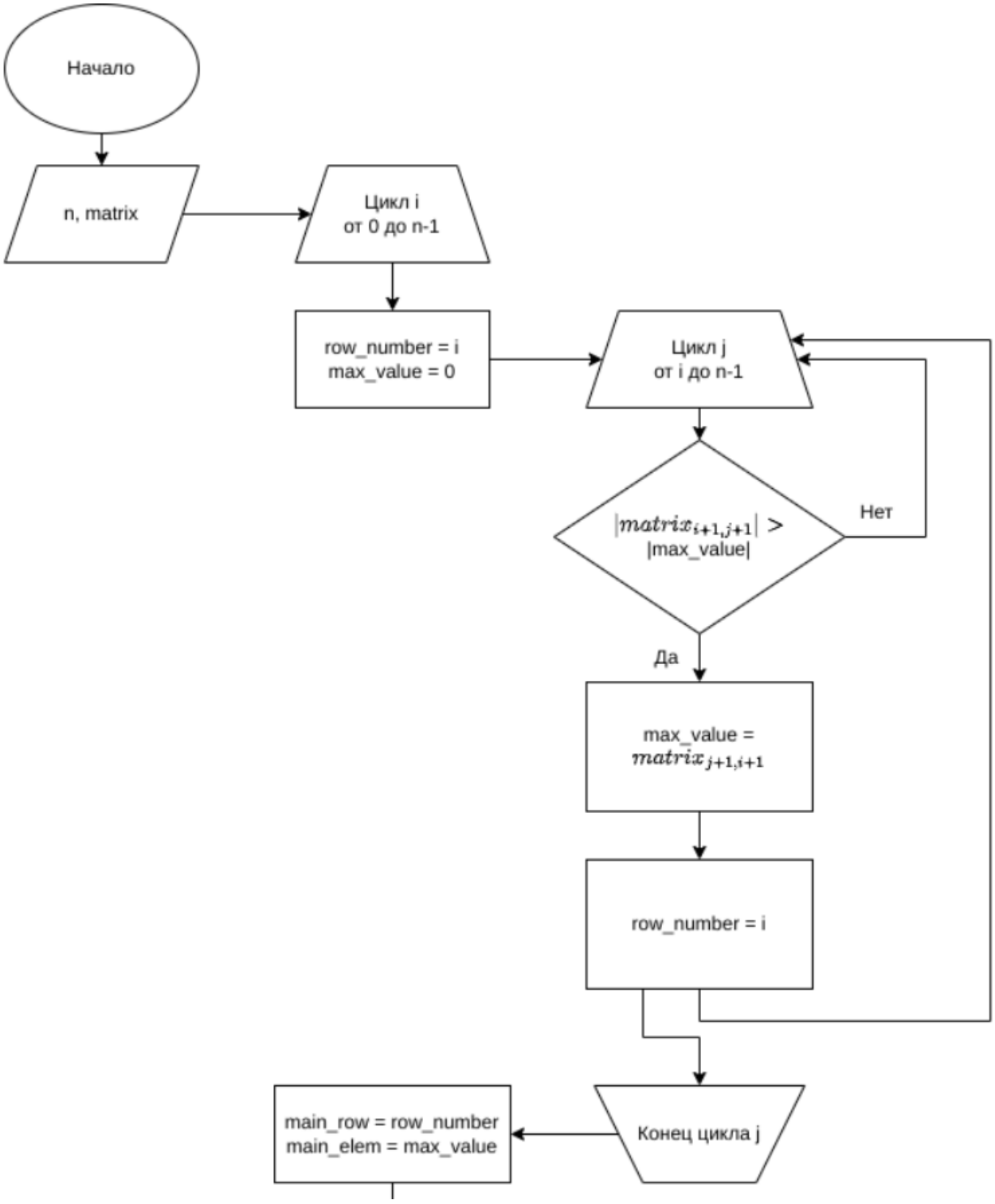
$$= matrix_{j,1} - matrix_{j,1} = 0$$

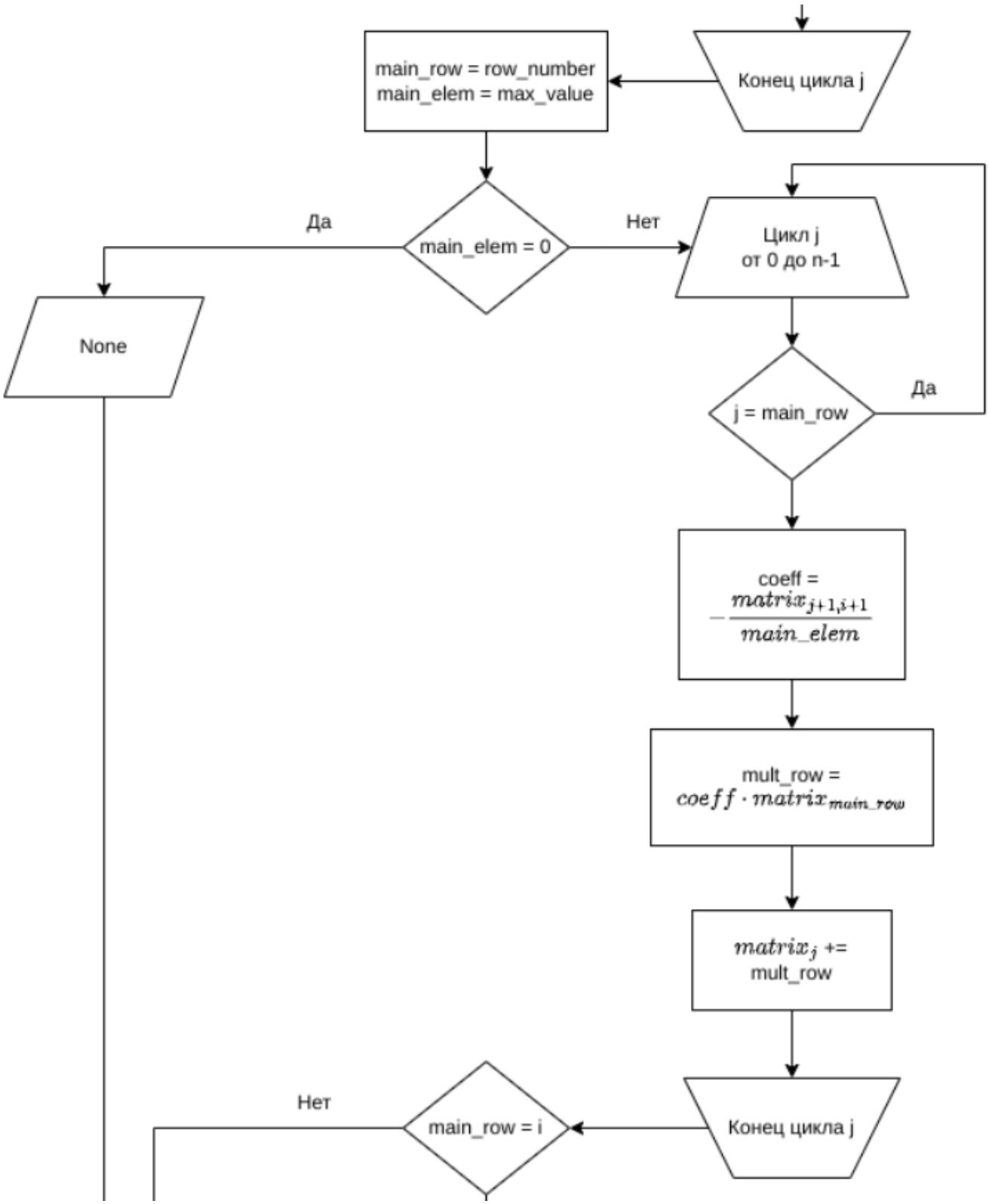
У нас зануляется весь первый стобец, кроме строки main_row. Затем мы поменяем местами эту строку и $matrix_1$

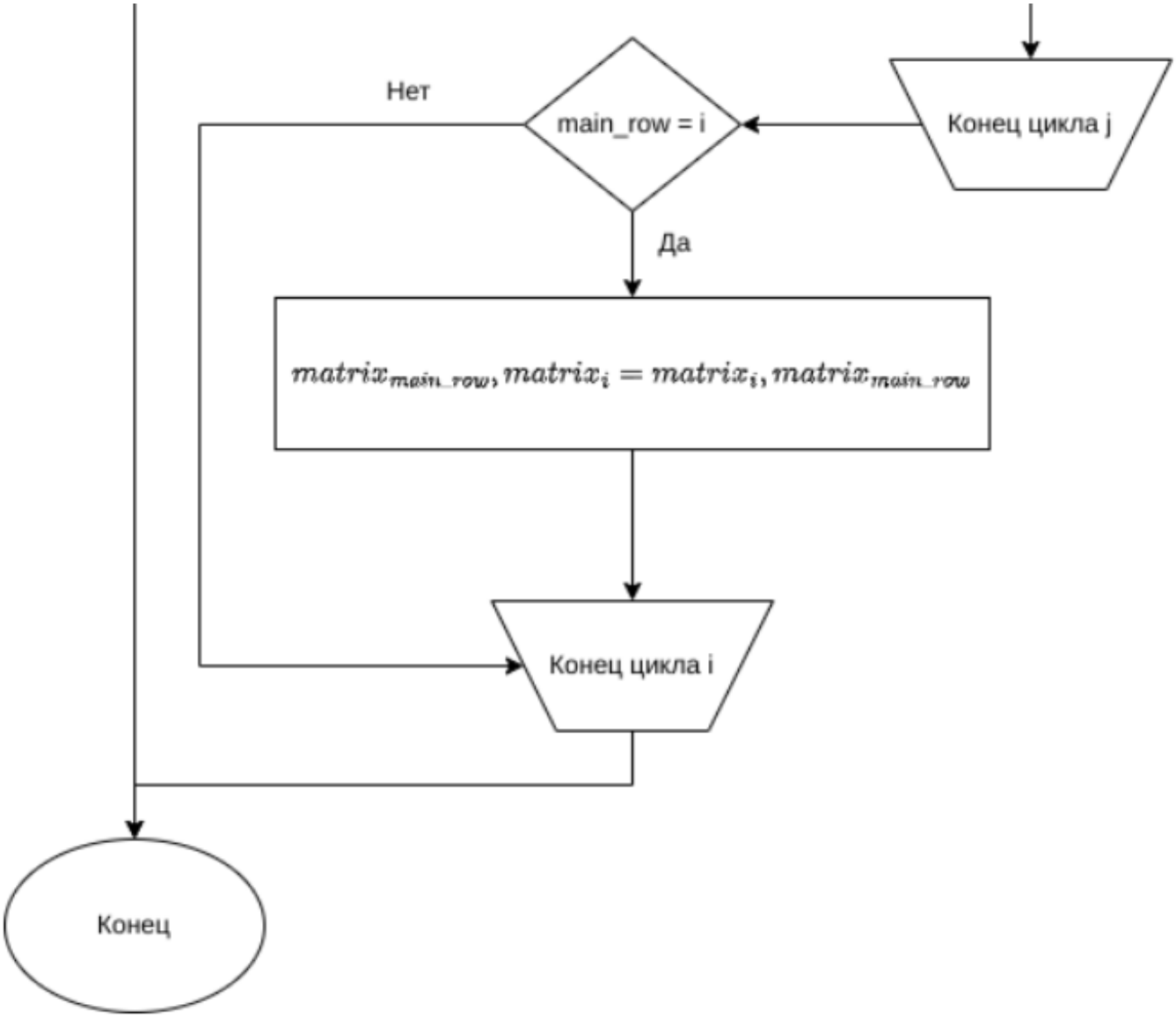
Дальнейшие итерации продолжаем со строки 2 и для столбца 2. В конечном счёте у нас останется матрица, у которой есть ненулевые числа только на главной диагонали и значения столбца B. В таком случае легко найти ответ:

$$x_i = \frac{final\_B_i}{final\_matrix_{i,i}}$$

2 Блок-схема метода




Конец цикла j
main_row = row_number
main_elem = max_value
main_elem = 0
Да
Нет
None
Цикл j
от 0 до n-1
j = main_row
Да
coeff =
$-\frac{matrix_{j+1,i+1}}{main\_elem}$
mult_row =
$coeff \cdot matrix_{main\_row}$
$matrix_j$ +=
mult_row
Конец цикла j
main_row = i
Нет


Нет
main_row = i
Конец цикла j
Да
$matrix_{main\_row}, matrix_i = matrix_i, matrix_{main\_row}$
Конец цикла i
Конец

3 Исходный код

```
def solve_matrix(matrix: list[list[float]]) -> Union[tuple, None]:
    n = len(matrix)
    wm = working_matrix = copy.deepcopy(matrix)

    for i in range(n):
        main_row, main_elem = MO.find_max_value_in_column(wm, i, i)
        if main_elem == 0:
            return None

        for j in range(n):
            if j == main_row:
                continue
            coeff = - wm[j][i] / main_elem
            MO.add_multiplied_row(wm, MO.get_multiplied_row(wm, main_row, coeff), j)

        if main_row != i:
            MO.change_rows(wm, main_row, i)

    det = reduce(lambda x, y: x * y, [row[i] for i, row in enumerate(matrix)])

    return [row[-1] / row[i] for i, row in enumerate(working_matrix)], working_matrix, det
```

4 Примеры и результаты работы

4.1 Матрица 1

```
[59, -94, -52, 66, -5, -47]
[56, 62, 64, -33, 84, -6]
[43, -46, -66, -56, -42, -67]
[-63, -68, 42, -70, -65, 13]
[6, -19, -9, 31, -65, -38]

Diagonal:
[-63.0, 0.0, 0.0, 0.0, 0.0, 55.25]
[0.0, -157.68, 0.0, 0.0, 0.0, 1.86]
[0.0, -0.0, 101.21, 0.0, 0.0, 0.97]
[0.0, -0.0, 0.0, -132.18, 0.0, -13.84]
[0.0, -0.0, 0.0, 0.0, -66.34, -36.86]

det = -1098497400

Answer:
x1 = -0.88
x2 = -0.01
x3 = 0.01
x4 = 0.1
x5 = 0.56

Error:
r1 = 0.0
r2 = 7.105427357601002e-15
r3 = -1.4210854715202004e-14
r4 = -7.105427357601002e-15
r5 = 0.0

Working time: 0.000
```

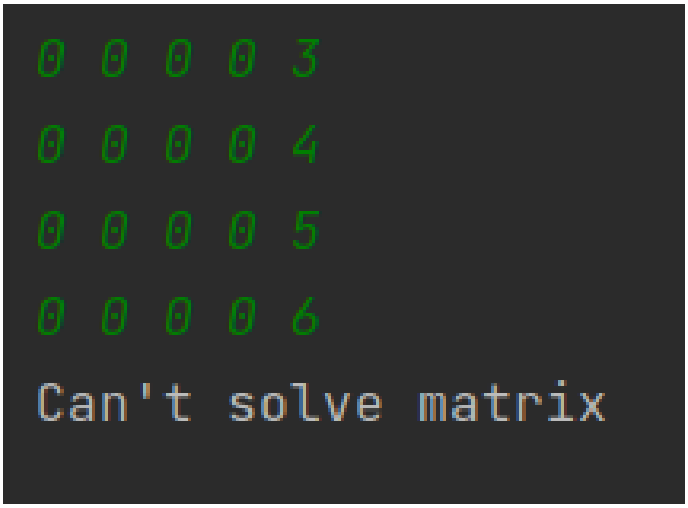
4.2 Матрица 2

```
[2, 22, -70, -5, 15, -77, 15, 18]
[44, -43, 73, -99, -43, -47, -52, 45]
[6, -2, -8, -94, -16, -73, 36, -35]
[-57, -99, 51, 94, 25, 7, 51, -87]
[-49, 3, 19, -87, -89, -20, -92, 16]
[67, -65, -3, 89, -67, -82, 72, -59]
[-91, 0, -96, -89, 100, -15, -56, 100]

Diagonal:
[-91.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -61.25]
[0.0, -99.0, 0.0, 0.0, 0.0, 0.0, 0.0, 22.66]
[0.0, 0.0, -146.65, 0.0, 0.0, 0.0, 0.0, 67.08]
[0.0, 0.0, 0.0, -196.01, 0.0, 0.0, 0.0, 12.6]
[0.0, 0.0, 0.0, 0.0, -134.46, 0.0, 0.0, -69.4]
[0.0, 0.0, 0.0, 0.0, 0.0, -53.07, -0.0, -1.52]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 85.21, -91.95]

det = -26430670336

Answer:
x1 = 0.67
x2 = -0.23
x3 = -0.46
x4 = -0.06
x5 = 0.52
x6 = 0.03
x7 = -1.08

Error:
r1 = -3.552713678800501e-15
r2 = 1.4210854715202004e-14
r3 = -1.4210854715202004e-14
r4 = 0.0
r5 = -1.4210854715202004e-14
r6 = -7.105427357601002e-15
r7 = 0.0

Working time: 0.000
```

4.3 Матрица 3

```
1 1 1 1 5
1 1 1 1 6
1 1 1 1 -2
1 1 1 1 3
Can't solve matrix
```

4.4 Матрица 4



5 Вывод

В ходе выполнения работы реализовали Метод Гаусса с выбором главного элемента и запустили его на 4 входных матрицах.
В первых двух случаях алгоритм нашёл решения с погрешностью порядка 10^{-14}, что, по моему мнению, является довольно неплохо.
В остальных двух случаях, алгоритм не смог найти решение, о чём выдал соответствующее сообщение.
Сложность алгоритма составляет $O(n^3)$, так как у нас есть 3 вложенных цикла, которые проходятся по матрице размерности n.
Кроме того, недостатком метода является хранение всей матрицы в памяти, что может быть затратно, если необходимо найти решение для матрицы довольно больших размеров.