|  |  |
| --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ | |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  «Пермский государственный национальный исследовательский университет» | |
| **ЧИСЛЕННЫЕ МЕТОДЫ** *Лабораторная работа №\_2\_*  **«**Прямые методы решения систем линейных алгебраических уравнений**»** | |
| Варианты: №3, №6 | |
|  | Работу выполнили студенты группы ПМИ-2-16  Мироненко Анастасия Олеговна,  Зимин Илья Владимирович |
| Оценка отчета   |  |  |  | | --- | --- | --- | |  | Баллы |  | | Конспект | 1 |  | | Опоздание с отчетом | -0.05 день |  | | Попытки | -0.5 попыт. |  | | Замечания к отчету | -0.25 зам. |  | | Программа | 1 |  | | Выводы  (Заключение) | 1 |  | | Защита | 1 |  | | ИТОГО: |  |  | | Проверил:  профессор, доктор физико-математических наук  С. В. Русаков  “\_\_\_\_” 2018 г. |
| Замечания:  Пермь 2018 | |

**Содержание**

[1. Задание. 3](#_Toc525161389)

[2. Исходные данные. 4](#_Toc525161390)

[3. Решение 5](#_Toc525161391)

[4. Краткие выводы. 6](#_Toc525161392)

[5. Текст программы. 7](#_Toc525161393)

1. **Задание**
2. Решить заданную систему линейных алгебраических уравнений 

методом *LU* разложения с выбором главного элемента по столбцу.

1. Используя полученное ранее *LU* разложение вычислить обратную матрицу к исходной.
2. Вычислить определитель и найти число обусловленности исходной матрицы в кубической, октаэдрической и евклидовой нормах.

*Замечание:* Для определения собственных значений матрицы (евклидова норма) можно воспользоваться любым математическим пакетом.

1. **Исходные данные**

|  |  |
| --- | --- |
| Вариант №3: | Вариант №6 |
|  |  |

1. **Решение**

В данном разделе будут приведены результаты, а также подробные шаги выполнения программы, на примере двух наборов: №3 и №6.

Программа считывает из файла «data.txt» матрицу , а так же вектор , затем автоматически генерирует файл «output.txt» и выводит в него все необходимые данные.

Набор №3. а)

Файл «data.txt»:

-9.7 5.2 -3.2 -0.9

1.1 -8.3 -3.1 -8.4

8.0 8.2 -1.5 0.3

8.0 -2.0 2.2 -9.4

1 2 3 4

Файл «output.txt»:

Вектор x:

1.0000000 2.0000000 3.0000000 4.0000000

Исходная матрица А:

-9.7000000 5.2000000 -3.2000000 -0.9000000

1.1000000 -8.3000000 -3.1000000 -8.4000000

8.0000000 8.2000000 -1.5000000 0.3000000

8.0000000 -2.0000000 2.2000000 -9.4000000

Вектор b:

-12.5000000 -58.4000000 21.1000000 -27.0000000

Построение матрицы U:

ШАГ 1

Максимальный элемент в столбце 1: 9.7000000

Перестановка строк не требуется

1-ю строку делим на -9.7000000

1.0000000 -0.5360825 0.3298969 0.0927835

1.1000000 -8.3000000 -3.1000000 -8.4000000

8.0000000 8.2000000 -1.5000000 0.3000000

8.0000000 -2.0000000 2.2000000 -9.4000000

Из 2-й строки вычитаем 1-ю, умноженную на 1.1000000

1.0000000 -0.5360825 0.3298969 0.0927835

0.0000000 -7.7103093 -3.4628866 -8.5020619

8.0000000 8.2000000 -1.5000000 0.3000000

8.0000000 -2.0000000 2.2000000 -9.4000000

Из 3-й строки вычитаем 1-ю, умноженную на 8.0000000

1.0000000 -0.5360825 0.3298969 0.0927835

0.0000000 -7.7103093 -3.4628866 -8.5020619

0.0000000 12.4886598 -4.1391753 -0.4422680

8.0000000 -2.0000000 2.2000000 -9.4000000

Из 4-й строки вычитаем 1-ю, умноженную на 8.0000000

1.0000000 -0.5360825 0.3298969 0.0927835

0.0000000 -7.7103093 -3.4628866 -8.5020619

0.0000000 12.4886598 -4.1391753 -0.4422680

0.0000000 2.2886598 -0.4391753 -10.1422680

ШАГ 2

Максимальный элемент в столбце 2: 12.4886598

Меняем местами 2-ю и 3-ю строки

1.0000000 -0.5360825 0.3298969 0.0927835

0.0000000 12.4886598 -4.1391753 -0.4422680

0.0000000 -7.7103093 -3.4628866 -8.5020619

0.0000000 2.2886598 -0.4391753 -10.1422680

2-ю строку делим на 12.4886598

1.0000000 -0.5360825 0.3298969 0.0927835

0.0000000 1.0000000 -0.3314347 -0.0354136

0.0000000 -7.7103093 -3.4628866 -8.5020619

0.0000000 2.2886598 -0.4391753 -10.1422680

Из 3-й строки вычитаем 2-ю, умноженную на -7.7103093

1.0000000 -0.5360825 0.3298969 0.0927835

0.0000000 1.0000000 -0.3314347 -0.0354136

0.0000000 0.0000000 -6.0183507 -8.7751114

0.0000000 2.2886598 -0.4391753 -10.1422680

Из 4-й строки вычитаем 2-ю, умноженную на 2.2886598

1.0000000 -0.5360825 0.3298969 0.0927835

0.0000000 1.0000000 -0.3314347 -0.0354136

0.0000000 0.0000000 -6.0183507 -8.7751114

0.0000000 0.0000000 0.3193660 -10.0612184

ШАГ 3

Максимальный элемент в столбце 3: 6.0183507

Перестановка строк не требуется

3-ю строку делим на -6.0183507

1.0000000 -0.5360825 0.3298969 0.0927835

0.0000000 1.0000000 -0.3314347 -0.0354136

-0.0000000 -0.0000000 1.0000000 1.4580592

0.0000000 0.0000000 0.3193660 -10.0612184

Из 4-й строки вычитаем 3-ю, умноженную на 0.3193660

1.0000000 -0.5360825 0.3298969 0.0927835

0.0000000 1.0000000 -0.3314347 -0.0354136

-0.0000000 -0.0000000 1.0000000 1.4580592

0.0000000 0.0000000 0.0000000 -10.5268730

ШАГ 4

Максимальный элемент в столбце 4: 10.5268730

Перестановка строк не требуется

4-ю строку делим на -10.5268730

1.0000000 -0.5360825 0.3298969 0.0927835

0.0000000 1.0000000 -0.3314347 -0.0354136

-0.0000000 -0.0000000 1.0000000 1.4580592

-0.0000000 -0.0000000 -0.0000000 1.0000000

Вычисляем матрицу L:

ШАГ 1

Ставим на позицию (1,1) матрицы L, элемент -9.7000000, который берем с той же позиции матрицы А

-9.7000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 2

Ставим на позицию (2,1) матрицы L, элемент 8.0000000, который берем с той же позиции матрицы А

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Ставим на позицию (2,2) матрицы L, элемент 8.2000000, который берем с той же позиции матрицы А

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 8.2000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Вычитаем из L(2,2) = 8.2000000 произведение L(2,1)\*U(1,2) = (8.0000000)\*(-0.5360825)

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 3

Ставим на позицию (3,1) матрицы L, элемент 1.1000000, который берем с той же позиции матрицы А

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Ставим на позицию (3,2) матрицы L, элемент -8.3000000, который берем с той же позиции матрицы А

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -8.3000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Вычитаем из L(3,2) = -8.3000000 произведение L(3,1)\*U(1,2) = (1.1000000)\*(-0.5360825)

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Ставим на позицию (3,3) матрицы L, элемент -3.1000000, который берем с той же позиции матрицы А

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -3.1000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Вычитаем из L(3,3) = -3.1000000 произведение L(3,1)\*U(1,3) = (1.1000000)\*(0.3298969)

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -3.4628866 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Вычитаем из L(3,3) = -3.4628866 произведение L(3,2)\*U(2,3) = (-7.7103093)\*(-0.3314347)

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 4

Ставим на позицию (4,1) матрицы L, элемент 8.0000000, который берем с той же позиции матрицы А

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

8.0000000 0.0000000 0.0000000 0.0000000

Ставим на позицию (4,2) матрицы L, элемент -2.0000000, который берем с той же позиции матрицы А

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

8.0000000 -2.0000000 0.0000000 0.0000000

Вычитаем из L(4,2) = -2.0000000 произведение L(4,1)\*U(1,2) = (8.0000000)\*(-0.5360825)

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

8.0000000 2.2886598 0.0000000 0.0000000

Ставим на позицию (4,3) матрицы L, элемент 2.2000000, который берем с той же позиции матрицы А

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

8.0000000 2.2886598 2.2000000 0.0000000

Вычитаем из L(4,3) = 2.2000000 произведение L(4,1)\*U(1,3) = (8.0000000)\*(0.3298969)

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

8.0000000 2.2886598 -0.4391753 0.0000000

Вычитаем из L(4,3) = -0.4391753 произведение L(4,2)\*U(2,3) = (2.2886598)\*(-0.3314347)

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

8.0000000 2.2886598 0.3193660 0.0000000

Ставим на позицию (4,4) матрицы L, элемент -9.4000000, который берем с той же позиции матрицы А

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

8.0000000 2.2886598 0.3193660 -9.4000000

Вычитаем из L(4,4) = -9.4000000 произведение L(4,1)\*U(1,4) = (8.0000000)\*(0.0927835)

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

8.0000000 2.2886598 0.3193660 -10.1422680

Вычитаем из L(4,4) = -10.1422680 произведение L(4,2)\*U(2,4) = (2.2886598)\*(-0.0354136)

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

8.0000000 2.2886598 0.3193660 -10.0612184

Вычитаем из L(4,4) = -10.0612184 произведение L(4,3)\*U(3,4) = (0.3193660)\*(1.4580592)

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

8.0000000 2.2886598 0.3193660 -10.5268730

Ранг исходной матрицы А: 4

Определитель матрицы A: 7674.7536000

Вектор x:

1.0000000 2.0000000 3.0000000 4.0000000

Вычисление обратной матрицы:

Единичная матрица:

1.0000000 0.0000000 0.0000000 0.0000000

0.0000000 1.0000000 0.0000000 0.0000000

0.0000000 0.0000000 1.0000000 0.0000000

0.0000000 0.0000000 0.0000000 1.0000000

ШАГ 1

Решим исходную систему, где вектор правых частей равен 1-й строке из матрицы E

Запишем полученный вектор в строку результирующей матрицы

-0.0618881 0.0618151 -0.0055726 -0.0671271

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 2

Решим исходную систему, где вектор правых частей равен 2-й строке из матрицы E

Запишем полученный вектор в строку результирующей матрицы

-0.0618881 0.0618151 -0.0055726 -0.0671271

0.0245459 -0.0528132 -0.1588085 -0.0050409

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 3

Решим исходную систему, где вектор правых частей равен 3-й строке из матрицы E

Запишем полученный вектор в строку результирующей матрицы

-0.0618881 0.0618151 -0.0055726 -0.0671271

0.0245459 -0.0528132 -0.1588085 -0.0050409

0.0606589 0.0396703 -0.1234288 0.0142965

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 4

Решим исходную систему, где вектор правых частей равен 4-й строке из матрицы E

Запишем полученный вектор в строку результирующей матрицы

-0.0618881 0.0618151 -0.0055726 -0.0671271

0.0245459 -0.0528132 -0.1588085 -0.0050409

0.0606589 0.0396703 -0.1234288 0.0142965

-0.0140733 0.0425423 0.1385083 -0.0949950

ШАГ 5: транспонируем полученную матрицу

-0.0618881 0.0245459 0.0606589 -0.0140733

0.0618151 -0.0528132 0.0396703 0.0425423

-0.0055726 -0.1588085 -0.1234288 0.1385083

-0.0671271 -0.0050409 0.0142965 -0.0949950

Итак, имеем:

===============

L:

-9.7000000 0.0000000 0.0000000 0.0000000

8.0000000 12.4886598 0.0000000 0.0000000

1.1000000 -7.7103093 -6.0183507 0.0000000

8.0000000 2.2886598 0.3193660 -10.5268730

U:

1.0000000 -0.5360825 0.3298969 0.0927835

0.0000000 1.0000000 -0.3314347 -0.0354136

-0.0000000 -0.0000000 1.0000000 1.4580592

-0.0000000 -0.0000000 -0.0000000 1.0000000

P:

1.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 1.0000000 0.0000000

0.0000000 1.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 1.0000000

A:

-9.7000000 5.2000000 -3.2000000 -0.9000000

8.0000000 8.2000000 -1.5000000 0.3000000

1.1000000 -8.3000000 -3.1000000 -8.4000000

8.0000000 -2.0000000 2.2000000 -9.4000000

A^(-1):

-0.0618881 0.0245459 0.0606589 -0.0140733

0.0618151 -0.0528132 0.0396703 0.0425423

-0.0055726 -0.1588085 -0.1234288 0.1385083

-0.0671271 -0.0050409 0.0142965 -0.0949950

b:

-12.5000000 -58.4000000 21.1000000 -27.0000000

x:

1.0000000 2.0000000 3.0000000 4.0000000

===============

P\*A:

-9.7000000 5.2000000 -3.2000000 -0.9000000

8.0000000 8.2000000 -1.5000000 0.3000000

1.1000000 -8.3000000 -3.1000000 -8.4000000

8.0000000 -2.0000000 2.2000000 -9.4000000

L\*U:

-9.7000000 5.2000000 -3.2000000 -0.9000000

8.0000000 8.2000000 -1.5000000 0.3000000

1.1000000 -8.3000000 -3.1000000 -8.4000000

8.0000000 -2.0000000 2.2000000 -9.4000000

PA - LU:

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Норма невязки: ||PA - LU|| = 0.0000000

Ax: -12.5000000 -58.4000000 21.1000000 -27.0000000

Ax-b: -0.0000000 -0.0000000 -0.0000000 0.0000000

Норма невязки: ||Ax - b|| = 0.0000000

A\*A^(-1):

1.0000000 -0.0000000 -0.0000000 -0.0000000

0.0000000 1.0000000 0.0000000 0.0000000

-0.0000000 -0.0000000 1.0000000 0.0000000

0.0000000 0.0000000 0.0000000 1.0000000

A\*A^(-1) - E:

-0.0000000 -0.0000000 -0.0000000 -0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

-0.0000000 -0.0000000 -0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Норма невязки: ||A\*A^(-1) - E|| = 0.0000000

Число обусловленности исходной матрицы

В кубической норме: 9.208473

В октаэдрической норме: 7.775186

В евклидовой норме: 4.561914

Набор №6. а)

Файл «data.txt»:

-7.8 -2.4 -0.3 -3.1

-8.0 4.5 -9.2 2.3

-4.8 5.5 1.7 -9.5

-2.5 0.3 -8.8 7.5

1 2 3 4

Файл «output.txt»:

Вектор x:

1.0000000 2.0000000 3.0000000 4.0000000

Исходная матрица А:

-7.8000000 -2.4000000 -0.3000000 -3.1000000

-8.0000000 4.5000000 -9.2000000 2.3000000

-4.8000000 5.5000000 1.7000000 -9.5000000

-2.5000000 0.3000000 -8.8000000 7.5000000

Вектор b:

-25.9000000 -17.4000000 -26.7000000 1.7000000

Построение матрицы U:

ШАГ 1

Максимальный элемент в столбце 1: 8.0000000

Меняем местами 1-ю и 2-ю строки

-8.0000000 4.5000000 -9.2000000 2.3000000

-7.8000000 -2.4000000 -0.3000000 -3.1000000

-4.8000000 5.5000000 1.7000000 -9.5000000

-2.5000000 0.3000000 -8.8000000 7.5000000

1-ю строку делим на -8.0000000

1.0000000 -0.5625000 1.1500000 -0.2875000

-7.8000000 -2.4000000 -0.3000000 -3.1000000

-4.8000000 5.5000000 1.7000000 -9.5000000

-2.5000000 0.3000000 -8.8000000 7.5000000

Из 2-й строки вычитаем 1-ю, умноженную на -7.8000000

1.0000000 -0.5625000 1.1500000 -0.2875000

0.0000000 -6.7875000 8.6700000 -5.3425000

-4.8000000 5.5000000 1.7000000 -9.5000000

-2.5000000 0.3000000 -8.8000000 7.5000000

Из 3-й строки вычитаем 1-ю, умноженную на -4.8000000

1.0000000 -0.5625000 1.1500000 -0.2875000

0.0000000 -6.7875000 8.6700000 -5.3425000

0.0000000 2.8000000 7.2200000 -10.8800000

-2.5000000 0.3000000 -8.8000000 7.5000000

Из 4-й строки вычитаем 1-ю, умноженную на -2.5000000

1.0000000 -0.5625000 1.1500000 -0.2875000

0.0000000 -6.7875000 8.6700000 -5.3425000

0.0000000 2.8000000 7.2200000 -10.8800000

0.0000000 -1.1062500 -5.9250000 6.7812500

ШАГ 2

Максимальный элемент в столбце 2: 6.7875000

Перестановка строк не требуется

2-ю строку делим на -6.7875000

1.0000000 -0.5625000 1.1500000 -0.2875000

-0.0000000 1.0000000 -1.2773481 0.7871087

0.0000000 2.8000000 7.2200000 -10.8800000

0.0000000 -1.1062500 -5.9250000 6.7812500

Из 3-й строки вычитаем 2-ю, умноженную на 2.8000000

1.0000000 -0.5625000 1.1500000 -0.2875000

-0.0000000 1.0000000 -1.2773481 0.7871087

0.0000000 0.0000000 10.7965746 -13.0839042

0.0000000 -1.1062500 -5.9250000 6.7812500

Из 4-й строки вычитаем 2-ю, умноженную на -1.1062500

1.0000000 -0.5625000 1.1500000 -0.2875000

-0.0000000 1.0000000 -1.2773481 0.7871087

0.0000000 0.0000000 10.7965746 -13.0839042

0.0000000 0.0000000 -7.3380663 7.6519890

ШАГ 3

Максимальный элемент в столбце 3: 10.7965746

Перестановка строк не требуется

3-ю строку делим на 10.7965746

1.0000000 -0.5625000 1.1500000 -0.2875000

-0.0000000 1.0000000 -1.2773481 0.7871087

0.0000000 0.0000000 1.0000000 -1.2118570

0.0000000 0.0000000 -7.3380663 7.6519890

Из 4-й строки вычитаем 3-ю, умноженную на -7.3380663

1.0000000 -0.5625000 1.1500000 -0.2875000

-0.0000000 1.0000000 -1.2773481 0.7871087

0.0000000 0.0000000 1.0000000 -1.2118570

0.0000000 0.0000000 0.0000000 -1.2406979

ШАГ 4

Максимальный элемент в столбце 4: 1.2406979

Перестановка строк не требуется

4-ю строку делим на -1.2406979

1.0000000 -0.5625000 1.1500000 -0.2875000

-0.0000000 1.0000000 -1.2773481 0.7871087

0.0000000 0.0000000 1.0000000 -1.2118570

-0.0000000 -0.0000000 -0.0000000 1.0000000

Вычисляем матрицу L:

ШАГ 1

Ставим на позицию (1,1) матрицы L, элемент -8.0000000, который берем с той же позиции матрицы А

-8.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 2

Ставим на позицию (2,1) матрицы L, элемент -7.8000000, который берем с той же позиции матрицы А

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Ставим на позицию (2,2) матрицы L, элемент -2.4000000, который берем с той же позиции матрицы А

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -2.4000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Вычитаем из L(2,2) = -2.4000000 произведение L(2,1)\*U(1,2) = (-7.8000000)\*(-0.5625000)

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 3

Ставим на позицию (3,1) матрицы L, элемент -4.8000000, который берем с той же позиции матрицы А

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Ставим на позицию (3,2) матрицы L, элемент 5.5000000, который берем с той же позиции матрицы А

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 5.5000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Вычитаем из L(3,2) = 5.5000000 произведение L(3,1)\*U(1,2) = (-4.8000000)\*(-0.5625000)

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Ставим на позицию (3,3) матрицы L, элемент 1.7000000, который берем с той же позиции матрицы А

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 1.7000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Вычитаем из L(3,3) = 1.7000000 произведение L(3,1)\*U(1,3) = (-4.8000000)\*(1.1500000)

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 7.2200000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

Вычитаем из L(3,3) = 7.2200000 произведение L(3,2)\*U(2,3) = (2.8000000)\*(-1.2773481)

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 4

Ставим на позицию (4,1) матрицы L, элемент -2.5000000, который берем с той же позиции матрицы А

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

-2.5000000 0.0000000 0.0000000 0.0000000

Ставим на позицию (4,2) матрицы L, элемент 0.3000000, который берем с той же позиции матрицы А

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

-2.5000000 0.3000000 0.0000000 0.0000000

Вычитаем из L(4,2) = 0.3000000 произведение L(4,1)\*U(1,2) = (-2.5000000)\*(-0.5625000)

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

-2.5000000 -1.1062500 0.0000000 0.0000000

Ставим на позицию (4,3) матрицы L, элемент -8.8000000, который берем с той же позиции матрицы А

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

-2.5000000 -1.1062500 -8.8000000 0.0000000

Вычитаем из L(4,3) = -8.8000000 произведение L(4,1)\*U(1,3) = (-2.5000000)\*(1.1500000)

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

-2.5000000 -1.1062500 -5.9250000 0.0000000

Вычитаем из L(4,3) = -5.9250000 произведение L(4,2)\*U(2,3) = (-1.1062500)\*(-1.2773481)

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

-2.5000000 -1.1062500 -7.3380663 0.0000000

Ставим на позицию (4,4) матрицы L, элемент 7.5000000, который берем с той же позиции матрицы А

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

-2.5000000 -1.1062500 -7.3380663 7.5000000

Вычитаем из L(4,4) = 7.5000000 произведение L(4,1)\*U(1,4) = (-2.5000000)\*(-0.2875000)

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

-2.5000000 -1.1062500 -7.3380663 6.7812500

Вычитаем из L(4,4) = 6.7812500 произведение L(4,2)\*U(2,4) = (-1.1062500)\*(0.7871087)

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

-2.5000000 -1.1062500 -7.3380663 7.6519890

Вычитаем из L(4,4) = 7.6519890 произведение L(4,3)\*U(3,4) = (-7.3380663)\*(-1.2118570)

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

-2.5000000 -1.1062500 -7.3380663 -1.2406979

Ранг исходной матрицы А: 4

Определитель матрицы A: 727.3641000

Вектор x:

1.0000000 2.0000000 3.0000000 4.0000000

Вычисление обратной матрицы:

Единичная матрица E:

1.0000000 0.0000000 0.0000000 0.0000000

0.0000000 1.0000000 0.0000000 0.0000000

0.0000000 0.0000000 1.0000000 0.0000000

0.0000000 0.0000000 0.0000000 1.0000000

ШАГ 1

Решим исходную систему, где вектор правых частей равен 1-й строке из матрицы E

Запишем полученный вектор в строку результирующей матрицы

-0.0351928 -0.1705157 -0.0764569 -0.0946197

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 2

Решим исходную систему, где вектор правых частей равен 2-й строке из матрицы E

Запишем полученный вектор в строку результирующей матрицы

-0.0351928 -0.1705157 -0.0764569 -0.0946197

-0.4604173 0.5369883 0.7225281 0.6728143

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 3

Решим исходную систему, где вектор правых частей равен 3-й строке из матрицы E

Запишем полученный вектор в строку результирующей матрицы

-0.0351928 -0.1705157 -0.0764569 -0.0946197

-0.4604173 0.5369883 0.7225281 0.6728143

0.3315341 -0.2984929 -0.5712449 -0.5478095

0.0000000 0.0000000 0.0000000 0.0000000

ШАГ 4

Решим исходную систему, где вектор правых частей равен 4-й строке из матрицы E

Запишем полученный вектор в строку результирующей матрицы

-0.0351928 -0.1705157 -0.0764569 -0.0946197

-0.4604173 0.5369883 0.7225281 0.6728143

0.3315341 -0.2984929 -0.5712449 -0.5478095

0.5465915 -0.6132472 -0.9767543 -0.8059980

ШАГ 5: транспонируем полученную матрицу

-0.0351928 -0.4604173 0.3315341 0.5465915

-0.1705157 0.5369883 -0.2984929 -0.6132472

-0.0764569 0.7225281 -0.5712449 -0.9767543

-0.0946197 0.6728143 -0.5478095 -0.8059980

Итак, имеем:

===============

L:

-8.0000000 0.0000000 0.0000000 0.0000000

-7.8000000 -6.7875000 0.0000000 0.0000000

-4.8000000 2.8000000 10.7965746 0.0000000

-2.5000000 -1.1062500 -7.3380663 -1.2406979

U:

1.0000000 -0.5625000 1.1500000 -0.2875000

-0.0000000 1.0000000 -1.2773481 0.7871087

0.0000000 0.0000000 1.0000000 -1.2118570

-0.0000000 -0.0000000 -0.0000000 1.0000000

P:

0.0000000 1.0000000 0.0000000 0.0000000

1.0000000 0.0000000 0.0000000 0.0000000

0.0000000 0.0000000 1.0000000 0.0000000

0.0000000 0.0000000 0.0000000 1.0000000

A:

-8.0000000 4.5000000 -9.2000000 2.3000000

-7.8000000 -2.4000000 -0.3000000 -3.1000000

-4.8000000 5.5000000 1.7000000 -9.5000000

-2.5000000 0.3000000 -8.8000000 7.5000000

A^(-1):

-0.0351928 -0.4604173 0.3315341 0.5465915

-0.1705157 0.5369883 -0.2984929 -0.6132472

-0.0764569 0.7225281 -0.5712449 -0.9767543

-0.0946197 0.6728143 -0.5478095 -0.8059980

b:

-25.9000000 -17.4000000 -26.7000000 1.7000000

x:

1.0000000 2.0000000 3.0000000 4.0000000

===============

P\*A:

-8.0000000 4.5000000 -9.2000000 2.3000000

-7.8000000 -2.4000000 -0.3000000 -3.1000000

-4.8000000 5.5000000 1.7000000 -9.5000000

-2.5000000 0.3000000 -8.8000000 7.5000000

L\*U:

-8.0000000 4.5000000 -9.2000000 2.3000000

-7.8000000 -2.4000000 -0.3000000 -3.1000000

-4.8000000 5.5000000 1.7000000 -9.5000000

-2.5000000 0.3000000 -8.8000000 7.5000000

PA - LU:

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 -0.0000000 0.0000000 -0.0000000

0.0000000 0.0000000 0.0000000 0.0000000

0.0000000 -0.0000000 0.0000000 0.0000000

Норма невязки: ||PA - LU|| = 0.0000000

Ax: -25.9000000 -17.4000000 -26.7000000 1.7000000

Ax-b: 0.0000000 0.0000000 -0.0000000 0.0000000

Норма невязки: ||Ax - b|| = 0.0000000

A\*A^(-1):

1.0000000 0.0000000 -0.0000000 -0.0000000

-0.0000000 1.0000000 0.0000000 -0.0000000

0.0000000 -0.0000000 1.0000000 0.0000000

-0.0000000 0.0000000 0.0000000 1.0000000

A\*A^(-1) - E:

-0.0000000 0.0000000 -0.0000000 -0.0000000

-0.0000000 -0.0000000 0.0000000 -0.0000000

0.0000000 -0.0000000 0.0000000 0.0000000

-0.0000000 0.0000000 0.0000000 0.0000000

Норма невязки: ||A\*A^(-1) - E|| = 0.0000000

Число обусловленности исходной матрицы

В кубической норме: 56.327619

В октаэдрической норме: 67.973850

В евклидовой норме: 36.283895

1. **Краткие выводы**

Программа написана, тестирование проведено и можно сказать, что метод LU-разложения довольно хороший метод решения СЛАУ. Его несомненным преимуществом является простота с точки зрения программирования, а также быстрота выполнения.

1. **Текст программы**

#include <iostream>

#include <fstream>

#include <vector>

#include <iomanip>

#include <limits>

#include <cmath>

#include "linalg.h"

using namespace std;

using namespace alglib;

int const N = 4;

typedef vector<vector<double>> myMatrix;

// Перегрузка оператора >> для вектора

istream &operator >> (istream &in, vector<double> &v) {

double num;

for (int i = 0; i < N; i++) {

in >> num;

v.push\_back(num);

}

return in;

}

// Перегрузка оператора << для вектора

ostream &operator << (ostream &out, vector<double> &v) {

for (int i = 0; i < N; i++)

// 8 знаков после запятой и ширина столбца 15 символов

out << fixed << setprecision(7) << setw(13) << v[i];

return out;

}

// Умножение вектора на число (переопределение оператора \*)

vector<double> operator \* (vector<double> v, double num) {

vector<double> res;

for (int i = 0; i < N; i++)

res.push\_back(v[i] \* num);

return res;

}

// Деление вектора на число (переопределение оператора /)

vector<double> operator / (vector<double> v, double num) {

vector<double> res;

for (int i = 0; i < N; i++)

res.push\_back(v[i] / num);

return res;

}

// Разность 2-х векторов поэлементно (переопределение оператора -)

vector<double> operator - (vector<double> v1, vector<double> v2) {

vector<double> res;

for (int i = 0; i < N; i++)

res.push\_back(v1[i] - v2[i]);

return res;

}

// Перегрузка оператора >> для матрицы

istream &operator >> (istream &in, myMatrix &v) {

vector<double> str;

for (int i = 0; i < N; i++) {

in >> str;

v.push\_back(str);

str.clear();

}

return in;

}

// Перегрузка оператора << для матрицы

ostream &operator << (ostream &out, myMatrix &v) {

for (int i = 0; i < N; i++)

out << fixed << setprecision(7) << setw(13) << v[i];

return out;

}

// Перегрузка оператора \* для перемножения двух матриц

myMatrix operator \* (myMatrix &M1, myMatrix &M2) {

myMatrix res = EmptyMatrix();

for (int i = 0; i < M1.size(); i++)

for (int j = 0; j < M1[0].size(); j++)

for (int k = 0; k < M1[0].size(); k++)

res[i][j] += M1[i][k] \* M2[k][j];

return res;

}

// Перегрузка оператора \* для умножения матрицы на вектор

vector<double> operator \* (myMatrix &M, vector<double> &v) {

vector<double> res(v.size());

for (int i = 0; i < M.size(); i++)

for (int j = 0; j < v.size(); j++)

res[i] += v[j] \* M[i][j];

return res;

}

// Перегрузка оператора - для перемножения двух матриц

myMatrix operator - (myMatrix &M1, myMatrix &M2) {

myMatrix res = EmptyMatrix();

for (int i = 0; i < M1.size(); i++)

for (int j = 0; j < M1[0].size(); j++)

res[i][j] += M1[i][j] - M2[i][j];

return res;

}

// Создание пустой матрицы

myMatrix EmptyMatrix() {

vector<double> v(N);

myMatrix emp;

for (int i = 0; i < N; i++)

emp.push\_back(v);

return emp;

}

// Функция поиска индекса строки в которой лежит максимальный элемент нужного столбцы

int IndexMaxNumInColumn(myMatrix M, int columnIndex) {

int ind = columnIndex;

for (int i = columnIndex; i < N; i++)

if (fabs(M[i][columnIndex]) > fabs(M[ind][columnIndex])) ind = i;

if (M[ind][columnIndex] == 0) return -1;

return ind;

}

// Ф-я, которая меняет местами строки

void Swap(myMatrix &M, int index1, int index2)

{

auto tmp = M[index1];

M[index1] = M[index2];

M[index2] = tmp;

}

// Создаём матрицу U

myMatrix CreateMatrixU(myMatrix &A, myMatrix &P, ofstream &fout, int &rang, int &k){

auto U = A;

fout << "Построение матрицы U: " << endl;

int k = 0, ind;

// Пробегаем по всем строкам

for (int i = 0; i < N; i++) {

fout << "\nШАГ " << i + 1 << endl << endl;

// Ищем индекс максимального элемента в столбце

ind = IndexMaxNumInColumn(U, i);

fout << "Максимальный элемент в столбце " << i + 1 << ": " << fabs(U[ind][i]) << endl;

// Значит весь столбец (начиная со строки i и до N) состоит из 0, а значит ранг равен пройденному числу строк

if (ind == -1) {

rang = i;

return U; // Что-то вернуто-то надо

}

// Если найденный индекс не явл. индексом тек. строки, то меняем строки

if (ind != i) {

K \*= -1;

Swap(U, i, ind);

Swap(A, i, ind);

Swap(P, i, ind);

fout << "Меняем местами " << i + 1 << "-ю и " << ind + 1 << "-ю строки \n" << U;

}

else fout << "Перестановка строк не требуется \n";

fout << endl << i + 1 << "-ю строку делим на " << U[i][i] << endl;

// Диагональный элемент делаем 1

U[i] = U[i] / U[i][i];

fout << U;

// Зануляем всё, что под диагональю

for (int j = i + 1; j < N; j++) {

fout << "\nИз " << j + 1 << "-й строки вычитаем " << i + 1 << "-ю, умноженную на " << U[j][i] << endl;

U[j] = U[j] - U[i] \* U[j][i];

fout << U;

}

}

return U;

}

// Создаём матрицу L

myMatrix CreateMatrixL(myMatrix A, myMatrix U, ofstream &fout) {

auto L = EmptyMatrix();

fout << "\nВычисляем матрицу L: \n\n";

for (int i = 0; i < N; i++)

{

fout << "ШАГ " << i + 1 << endl << endl;

for (int j = 0; j <= i; j++)

{

fout << "Ставим на позицию (" << i + 1 << "," << j + 1 << ") матрицы L, элемент " << A[i][j] << ", \nкоторый берем с той же позиции матрицы А \n";

L[i][j] = A[i][j];

fout << L << endl;

for (int k = 0; k < j; k++)

{

fout << "Вычитаем из L("<< i + 1 << "," << j + 1 << ") = "<< L[i][j] << " произведение L(" << i + 1 << "," << k + 1 << ")\*U(" << k + 1 << "," << j + 1 << ") = (" << L[i][k] <<")\*("<< U[k][j] << ")\n";

L[i][j] -= L[i][k] \* U[k][j];

fout << L << endl;

}

}

}

return L;

}

// Вычисление определителя

double Determinant(myMatrix L)

{

double det = 1;

for (int i = 0; i < N; i++)

det \*= L[i][i];

return det;

}

// Создаем матрицу перестановок P

myMatrix CreateMatrixP()

{

auto P = EmptyMatrix();

for (int i = 0; i < N; i++)

P[i][i] = 1;

return P;

}

// Решаем систему, находим вектор X

vector<double> SolveX(vector<double> &b, myMatrix P, myMatrix L, myMatrix U) {

// Делаем перестановку элементов вектора b

auto b1 = P \* b;

// Решение уравнение Ly = b (Находим вектор y)

vector<double> y;

for (int i = 0; i < N; i++) {

for (int j = 0; j < i; j++)

b1[i] -= L[i][j] \* y[j];

y.push\_back(b1[i] / L[i][i]);

}

// Решение уравнение Ux=y (находим вектор x)

auto x = y;

for (int i = N - 1; i >= 0; i--)

for (int j = N - 1; j > i; j--)

x[i] -= U[i][j] \* x[j];

return x;

}

// Находим вектор правых частей

vector<double> SolveB(vector<double> x, myMatrix A) {

vector<double> b(N);

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++)

b[i] += A[i][j] \* x[j];

return b;

}

// Транспонирование матрицы

myMatrix TransposeMatrix(myMatrix M) {

myMatrix res = M;

for (int i = 0; i < N; i++)

for (int j = i + 1; j < N; j++) {

double tmp = res[i][j];

res[i][j] = res[j][i];

res[j][i] = tmp;

}

return res;

}

// Создание обратной матрицы

myMatrix ReverseMatrix(myMatrix P, myMatrix L, myMatrix U, ofstream &fout) {

fout << "Вычисление обратной матрицы: \n";

myMatrix A\_ = EmptyMatrix();

myMatrix E = CreateMatrixP();

fout << "Единичная матрица: \n" << E << endl;

for (int i = 0; i < N; i++) {

// Решение исходной системы уравнений, где вектор b = строке из Е

A\_[i] = SolveX(E[i], P, L, U);

fout << A\_ << endl;

}

return TransposeMatrix(A\_);

}

// Максимальная сумма столбцов/строк в матрице

double MaxSumRowOrColumn(myMatrix A, int numNorm) {

double max(0), sum(0);

for (int i = 0; i < A.size(); i++) {

for (int j = 0; j < A.size(); j++)

sum += numNorm == 1 ? fabs(A[i][j]) : fabs(A[j][i])

if (sum > max) max = sum;

sum = 0;

}

return max;

}

// Вычисление нормы I

double NormI(myMatrix A) {

return MaxSumRowOrColumn(A, 1);

}

// Вычисление нормы II

double NormII(myMatrix A) {

return MaxSumRowOrColumn(A, 2);

}

// Вычисление нормы III

double NormIII(myMatrix A) {

// Эрмитово сопряженная матрица А\*.

auto At = TransposeMatrix(A);

auto AtA = At \* A;

real\_2d\_array a, vl, vr;

a.setlength(N, N);

real\_1d\_array wl, wi;

wl.setlength(N);

wi.setlength(N);

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++)

a[i][j] = AtA[j][i];

rmatrixevd(a , N, 0, wl, wi, vl, vr);

double maxVal(DBL\_MIN);

for (int i = 0; i < N; i++)

if (fabs(wl[i]) > maxVal)

maxVal = fabs(wl[i]);

return maxVal;

}

// ОСНОВНАЯ ФУНКЦИЯ

int main() {

setlocale(LC\_ALL, "Russian");

// Файл для записи

ofstream fout;

// Файл для считывания

ifstream fin;

// A - исх. матрица, AnotP - исходная матрица, строки которой не переставляем, LU = L\*U, PA = P\*A, \_A - Обратная матрица и т.д.

myMatrix A, AnotP, L, U, LU, PA, PA\_LU, A\_, AA\_, AA\_\_E, P(CreateMatrixP()), E(CreateMatrixP());

// Исходный вектор b и вектор решений х

vector<double> b, x, Ax, Ax\_b;

// Ранг матрицы A (предполагаем, что он равен 4-м)

int rang = 4;

double k = 1;

// Открываем файл для считывания данных

fin.open("3naborZimin.txt");

// Считываем матрицу целиком

fin >> A;

AnotP = A;

// Считываем вектор целиком

fin >> x;

// Закрываем файл для считывания

fin.close();

fout.open("output.txt");

// Выведем матрицу и вектор

fout << "Вектор x: \n" << x << endl;

fout << "Исходная матрица А: \n" << A << endl;

b = SolveB(x, A);

fout << "Вектор b: \n" << b << endl;

// Создадим матрицу перестановок

P = CreateMatrixP();

// Создадим матрицу U

U = CreateMatrixU(A, P, fout, rang, k);

if (rang != 4)

{

fout << "Ранг матрицы А = " << rang << "\n\nПродолжать решение невозможно :(";

return 0;

}

// Создадим матрицу L

L = CreateMatrixL(A, U, fout);

fout << "\nРанг исходной матрицы А: " << rang;

fout << "\nОпределитель матрицы A: " << Determinant(L) \* k;

// Решим СЛАУ

x = SolveX(b, P, L, U);

fout << "\n\nВектор x: \n" << x << endl;

A\_ = ReverseMatrix(P, L, U, fout);

fout << endl << A\_ << endl;

fout << "Итак, имеем: \n=============== \nL:\n" << L << "\nU:\n" << U << "\nP:\n" << P << "\nA:\n" << A << "\nA^(-1):\n" << A\_ << "\nb:\n" << b << "\nx:\n" << x << "===============\n\n";

PA = P \* AnotP;

fout << "P\*A:" << endl << PA << endl;

LU = L \* U;

fout << "L\*U:" << endl << LU << endl;

PA\_LU = PA - LU;

fout << "PA - LU:" << endl << PA\_LU << "\nНорма невязки: ||PA - LU|| = " << NormII(PA\_LU) << endl << endl;

Ax = AnotP \* x;

Ax\_b = Ax - b;

fout << "Ax: " << Ax << "\nAx-b: " << Ax\_b;

// Найдём норму вектора:

double sum(0);

for (int i = 0; i < N; i++)

sum += fabs(Ax\_b[i]);

fout << "\nНорма невязки: ||Ax - b|| = " << sum;

AA\_ = AnotP \* A\_;

fout << "\n\nA\*A^(-1):\n" << AA\_ << endl;

AA\_\_E = AA\_ - E;

fout << "A\*A^(-1) - E:\n" << AA\_\_E << "\nНорма невязки: ||A\*A^(-1) - E|| = " << NormII(AA\_\_E) << "\n\n";

fout << "Число обусловленности исходной матрицы \n";

fout << setw(25) << setiosflags(ios::left) << "В кубической норме: " << setprecision(6) << NormI(A) \* NormI(A\_) << endl;

fout << setw(25) << "В октаэдрической норме: " << NormII(A) \* NormII(A\_) << endl;

fout << setw(25) << "В евклидовой норме: " << sqrt(NormIII(A)) \* sqrt(NormIII(A\_)) << endl;

fout.close();

system("pause");

return;

}