|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическое задание № 1 | | |
| по дисциплине «Численные методы» | | |
| **ПРЯМЫЕ МЕТОДЫ РЕШЕНИЯ СЛАУ** | | |
|  | | |
|  |  |  |
| Группа ПМ-12 | Лойченко данила, Овчинников иван |
| Вариант 7 |  |
|  |  |
|  |  |
|  |  |
| Преподаватели | задорожный александр генадьевич |
|  |  |
| Новосибирск, 2024 | | |

1. **Задание**

Разработать программу решения СЛАУ прямым методом с хранением матрицы в профильном или ленточном формате. Исследовать накопление погрешности и ее зависимость от числа обусловленности. Сравнить реализованный метод по точности получаемого решения и количеству действий с методом Гаусса.

1. **Математическая модель**

Пусть задана система линейный алгебраический уравнений:



По полученному варианту матрица А раскладывается на LU, где L – нижнетреугольная матрица, U – верхнетреугольная матрица с единицами на главной диагонали.

Предположим, что нам удалось разложить матрицу:



Подставляя (1.2) (1.1), получаем:



Обозначим:



Тогда подставляя (1.4) в (1.3), получим:



Таким образом, решение системы (1.1) сводится к трем основным этапам:

1. Из элементов А найти элементы матрицы L и U;
2. Решить систему (1.5) с нижнетреугольной матрицей L (прямой ход);
3. Решить систему (1.4) с верхнетреугольной матрицей U (обратный ход).

Рассмотри алгоритм получения LU – разложения. Матрицы L и U будем искать в следующем виде:



Учитывая равенство (1.2) и умножая последовательно строки матрицы L на столбцы матрицы U , получаем систему, состоящую из  уравнений с  неизвестными  и  ( n – размерность СЛАУ):



Решая систему (1.7), можно получить общие формулы для нахождения элементов матриц L и U:



При помощи получившихся матриц (1.6) необходимо решить две СЛАУ следующего вида





и





1. **Исследование матриц с регулируемым числом обусловленности**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| k | xk (один. точность) | |x\*-xk| (один. точность) | xk (дв. точность) | |x\*-xk| (дв. точность) | xk (скал. произв.) | |x\*-xk| (скал. произв.) |
| 0 | 1,0000128 | 0,128E-05 | 1,00000000000001 | 9,992E-15 | 1,0000088 | 8,800E-06 |
| 2,0000134 | 1,340E-05 | 2,00000000000001 | 1,021E-14 | 2,0000098 | 9,800E-06 |
| 3,0000136 | 1,360E-05 | 3,00000000000001 | 1,021E-14 | 3,0000091 | 9,100E-06 |
| 4,0000134 | 1,340E-05 | 4,00000000000001 | 9,770E-15 | 4,0000095 | 9,500E-06 |
| 5,0000134 | 1,340E-05 | 5,00000000000001 | 9,770E-15 | 5,0000095 | 9,500E-06 |
| 6,0000134 | 1,340E-05 | 6,00000000000001 | 9,770E-15 | 6,0000095 | 9,500E-06 |
| 7,0000134 | 1,340E-05 | 7,00000000000001 | 9,770E-15 | 7,0000095 | 9,500E-06 |
| 8,0000134 | 1,340E-05 | 8,00000000000001 | 1,066E-14 | 8,0000095 | 9,500E-06 |
| 9,0000143 | 1,430E-05 | 9,00000000000001 | 1,066E-14 | 9,0000095 | 9,500E-06 |
| 10,0000134 | 1,340E-05 | 10,00000000000000 | 0,000E+00 | 10,0000095 | 9,500E-06 |
| 1 | 1,0000840 | 8,400E-05 | 0,99999999999981 | 1,890E-13 | 1,0000460 | 4,600E-05 |
| 2,0000844 | 8,440E-05 | 1,99999999999981 | 1,901E-13 | 2,0000458 | 4,580E-05 |
| 3,0000844 | 8,440E-05 | 2,99999999999981 | 1,901E-13 | 3,0000458 | 4,580E-05 |
| 4,0000844 | 8,440E-05 | 3,99999999999981 | 1,901E-13 | 4,0000458 | 4,580E-05 |
| 5,0000844 | 8,440E-05 | 4,99999999999981 | 1,901E-13 | 5,0000458 | 4,580E-05 |
| 6,0000849 | 8,490E-05 | 5,99999999999981 | 1,901E-13 | 6,0000458 | 4,580E-05 |
| 7,0000844 | 8,440E-05 | 6,99999999999981 | 1,901E-13 | 7,0000458 | 4,580E-05 |
| 8,0000839 | 8,390E-05 | 7,99999999999981 | 1,901E-13 | 8,0000458 | 4,580E-05 |
| 9,0000849 | 8,490E-05 | 8,99999999999981 | 1,901E-13 | 9,0000458 | 4,580E-05 |
| 10,0000849 | 8,490E-05 | 9,99999999999981 | 1,901E-13 | 10,0000458 | 4,580E-05 |
| 2 | 0,9989979 | 1,002E-03 | 1,00000000000093 | 9,299E-13 | 0,9993006 | 6,994E-04 |
| 1,9989972 | 1,003E-03 | 2,00000000000093 | 9,299E-13 | 1,9993002 | 6,998E-04 |
| 2,9989974 | 1,003E-03 | 3,00000000000093 | 9,299E-13 | 2,9993000 | 7,000E-04 |
| 3,9989970 | 1,003E-03 | 4,00000000000093 | 9,299E-13 | 3,9993000 | 7,000E-04 |
| 4,9989967 | 1,003E-03 | 5,00000000000093 | 9,299E-13 | 4,9993000 | 7,000E-04 |
| 5,9989967 | 1,003E-03 | 6,00000000000093 | 9,299E-13 | 5,9993000 | 7,000E-04 |
| 6,9989977 | 1,002E-03 | 7,00000000000093 | 9,299E-13 | 6,9993000 | 7,000E-04 |
| 7,9989967 | 1,003E-03 | 8,00000000000093 | 9,308E-13 | 7,9993000 | 7,000E-04 |
| 8,9989967 | 1,003E-03 | 9,00000000000093 | 9,308E-13 | 8,9993000 | 7,000E-04 |
| 9,9989967 | 1,003E-03 | 10,00000000000090 | 9,006E-13 | 9,9993000 | 7,000E-04 |
| 3 | 0,9883238 | 1,168E-02 | 1,00000000001865 | 1,865E-11 | 1,0054047 | 5,405E-03 |
| 1,9883232 | 1,168E-02 | 2,00000000001865 | 1,865E-11 | 2,0054054 | 5,405E-03 |
| 2,9883232 | 1,168E-02 | 3,00000000001865 | 1,865E-11 | 3,0054052 | 5,405E-03 |
| 3,9883232 | 1,168E-02 | 4,00000000001865 | 1,865E-11 | 4,0054054 | 5,405E-03 |
| 4,9883232 | 1,168E-02 | 5,00000000001865 | 1,865E-11 | 5,0054059 | 5,406E-03 |
| 5,9883232 | 1,168E-02 | 6,00000000001865 | 1,865E-11 | 6,0054054 | 5,405E-03 |
| 6,9883232 | 1,168E-02 | 7,00000000001865 | 1,865E-11 | 7,0054059 | 5,406E-03 |
| 7,9883232 | 1,168E-02 | 8,00000000001865 | 1,865E-11 | 8,0054054 | 5,405E-03 |
| 8,9883223 | 1,168E-02 | 9,00000000001865 | 1,865E-11 | 9,0054054 | 5,405E-03 |
| 9,9883232 | 1,168E-02 | 10,00000000001860 | 1,860E-11 | 10,0054054 | 5,405E-03 |
| 4 | 0,8360667 | 1,639E-01 | 0,99999999993781 | 6,219E-11 | 0,9512349 | 4,877E-02 |
| 1,8360656 | 1,639E-01 | 1,99999999993780 | 6,220E-11 | 1,9512348 | 4,877E-02 |
| 2,8360655 | 1,639E-01 | 2,99999999993780 | 6,220E-11 | 2,9512351 | 4,876E-02 |
| 3,8360655 | 1,639E-01 | 3,99999999993780 | 6,220E-11 | 3,9512348 | 4,877E-02 |
| 4,8360658 | 1,639E-01 | 4,99999999993780 | 6,220E-11 | 4,9512348 | 4,877E-02 |
| 5,8360653 | 1,639E-01 | 5,99999999993780 | 6,220E-11 | 5,9512348 | 4,877E-02 |
| 6,8360653 | 1,639E-01 | 6,99999999993780 | 6,220E-11 | 6,9512348 | 4,877E-02 |
| 7,8360653 | 1,639E-01 | 7,99999999993780 | 6,220E-11 | 7,9512348 | 4,877E-02 |
| 8,8360653 | 1,639E-01 | 8,99999999993780 | 6,220E-11 | 8,9512348 | 4,877E-02 |
| 9,8360653 | 1,639E-01 | 9,99999999993780 | 6,220E-11 | 9,9512358 | 4,876E-02 |
| 5 | 0,9999996 | 4,000E-07 | 1,00000000000000 | 9,992E-16 | 0,1969315 | 8,031E-01 |
| 2,0000000 | 0,000E+00 | 1,99999999999999 | 9,992E-15 | 1,1969316 | 8,031E-01 |
| 3,0000002 | 2,000E-07 | 3,00000000000000 | 0,000E+00 | 2,1969314 | 8,031E-01 |
| 3,9999998 | 2,000E-07 | 3,99999999999999 | 1,021E-14 | 3,1969314 | 8,031E-01 |
| 4,9999995 | 5,000E-07 | 4,99999999999999 | 9,770E-15 | 4,1969309 | 8,031E-01 |
| 6,0000000 | 0,000E+00 | 5,99999999999999 | 9,770E-15 | 5,1969314 | 8,031E-01 |
| 7,0000000 | 0,000E+00 | 7,00000000000000 | 0,000E+00 | 6,1969318 | 8,031E-01 |
| 8,0000000 | 0,000E+00 | 7,99999999999999 | 9,770E-15 | 7,1969314 | 8,031E-01 |
| 9,0000000 | 0,000E+00 | 9,00000000000000 | 0,000E+00 | 8,1969309 | 8,031E-01 |
| 10,0000000 | 0,000E+00 | 10,00000000000000 | 0,000E+00 | 9,1969309 | 8,031E-01 |
| 6 | 2,9999995 | 2,000E+00 | 0,99999997512307 | 2,488E-08 | -11,9941521 | 1,299E+01 |
| 4,0000005 | 2,000E+00 | 1,99999997512306 | 2,488E-08 | -10,9941530 | 1,299E+01 |
| 5,0000000 | 2,000E+00 | 2,99999997512306 | 2,488E-08 | -9,9941521 | 1,299E+01 |
| 6,0000000 | 2,000E+00 | 3,99999997512306 | 2,488E-08 | -8,9941521 | 1,299E+01 |
| 6,9999990 | 2,000E+00 | 4,99999997512306 | 2,488E-08 | -7,9941521 | 1,299E+01 |
| 8,0000000 | 2,000E+00 | 5,99999997512306 | 2,488E-08 | -6,9941521 | 1,299E+01 |
| 9,0000000 | 2,000E+00 | 6,99999997512306 | 2,488E-08 | -5,9941521 | 1,299E+01 |
| 10,0000000 | 2,000E+00 | 7,99999997512306 | 2,488E-08 | -4,9941521 | 1,299E+01 |
| 11,0000000 | 2,000E+00 | 8,99999997512306 | 2,488E-08 | -3,9941518 | 1,299E+01 |
| 12,0000000 | 2,000E+00 | 9,99999997512306 | 2,488E-08 | -2,9941521 | 1,299E+01 |
| 7 | -0,9999995 | 2,000E+00 | 1,00000043534627 | 4,353E-07 | -3,4063563 | 4,406E+00 |
| 0,0000008 | 2,000E+00 | 2,00000043534627 | 4,353E-07 | -2,4063568 | 4,406E+00 |
| 1,0000001 | 2,000E+00 | 3,00000043534627 | 4,353E-07 | -1,4063562 | 4,406E+00 |
| 2,0000005 | 2,000E+00 | 4,00000043534627 | 4,353E-07 | -0,4063569 | 4,406E+00 |
| 3,0000005 | 2,000E+00 | 5,00000043534627 | 4,353E-07 | 0,5936434 | 4,406E+00 |
| 4,0000005 | 2,000E+00 | 6,00000043534627 | 4,353E-07 | 1,5936434 | 4,406E+00 |
| 5,0000005 | 2,000E+00 | 7,00000043534627 | 4,353E-07 | 2,5936432 | 4,406E+00 |
| 6,0000005 | 2,000E+00 | 8,00000043534627 | 4,353E-07 | 3,5936437 | 4,406E+00 |
| 7,0000000 | 2,000E+00 | 9,00000043534627 | 4,353E-07 | 4,5936432 | 4,406E+00 |
| 8,0000000 | 2,000E+00 | 10,00000043534620 | 4,353E-07 | 5,5936437 | 4,406E+00 |
| 8 | -0,9999995 | 2,000E+00 | 1,00000124384607 | 1,244E-06 | -3,4063563 | 4,406E+00 |
| 0,0000008 | 2,000E+00 | 2,00000124384607 | 1,244E-06 | -2,4063568 | 4,406E+00 |
| 1,0000001 | 2,000E+00 | 3,00000124384607 | 1,244E-06 | -1,4063562 | 4,406E+00 |
| 2,0000005 | 2,000E+00 | 4,00000124384607 | 1,244E-06 | -0,4063569 | 4,406E+00 |
| 3,0000005 | 2,000E+00 | 5,00000124384607 | 1,244E-06 | 0,5936434 | 4,406E+00 |
| 4,0000005 | 2,000E+00 | 6,00000124384607 | 1,244E-06 | 1,5936434 | 4,406E+00 |
| 5,0000005 | 2,000E+00 | 7,00000124384607 | 1,244E-06 | 2,5936432 | 4,406E+00 |
| 6,0000005 | 2,000E+00 | 8,00000124384607 | 1,244E-06 | 3,5936437 | 4,406E+00 |
| 7,0000000 | 2,000E+00 | 9,00000124384607 | 1,244E-06 | 4,5936432 | 4,406E+00 |
| 8,0000000 | 2,000E+00 | 10,00000124384600 | 1,244E-06 | 5,5936437 | 4,406E+00 |
| 9 |  |  | 0,99998756157023 | 1,244E-05 |  |  |
|  |  | 1,99998756157022 | 1,244E-05 |  |  |
|  |  | 2,99998756157022 | 1,244E-05 |  |  |
|  |  | 3,99998756157022 | 1,244E-05 |  |  |
|  |  | 4,99998756157022 | 1,244E-05 |  |  |
|  |  | 5,99998756157022 | 1,244E-05 |  |  |
|  |  | 6,99998756157022 | 1,244E-05 |  |  |
|  |  | 7,99998756157022 | 1,244E-05 |  |  |
|  |  | 8,99998756157022 | 1,244E-05 |  |  |
|  |  | 9,99998756157022 | 1,244E-05 |  |  |
| 10 |  |  | 1,00009328793320 | 9,329E-05 |  |  |
|  |  | 2,00009328793320 | 9,329E-05 |  |  |
|  |  | 3,00009328793320 | 9,329E-05 |  |  |
|  |  | 4,00009328793320 | 9,329E-05 |  |  |
|  |  | 5,00009328793320 | 9,329E-05 |  |  |
|  |  | 6,00009328793320 | 9,329E-05 |  |  |
|  |  | 7,00009328793320 | 9,329E-05 |  |  |
|  |  | 8,00009328793320 | 9,329E-05 |  |  |
|  |  | 9,00009328793320 | 9,329E-05 |  |  |
|  |  | 10,00009328793320 | 9,329E-05 |  |  |
| 11 |  |  | 0,99875621890548 | 1,244E-03 |  |  |
|  |  | 1,99875621890547 | 1,244E-03 |  |  |
|  |  | 2,99875621890547 | 1,244E-03 |  |  |
|  |  | 3,99875621890547 | 1,244E-03 |  |  |
|  |  | 4,99875621890547 | 1,244E-03 |  |  |
|  |  | 5,99875621890547 | 1,244E-03 |  |  |
|  |  | 6,99875621890547 | 1,244E-03 |  |  |
|  |  | 7,99875621890547 | 1,244E-03 |  |  |
|  |  | 8,99875621890547 | 1,244E-03 |  |  |
|  |  | 9,99875621890547 | 1,244E-03 |  |  |
| 12 |  |  | 1,00311041990668 | 3,110E-03 |  |  |
|  |  | 2,00311041990668 | 3,110E-03 |  |  |
|  |  | 3,00311041990668 | 3,110E-03 |  |  |
|  |  | 4,00311041990668 | 3,110E-03 |  |  |
|  |  | 5,00311041990668 | 3,110E-03 |  |  |
|  |  | 6,00311041990668 | 3,110E-03 |  |  |
|  |  | 7,00311041990668 | 3,110E-03 |  |  |
|  |  | 8,00311041990668 | 3,110E-03 |  |  |
|  |  | 9,00311041990668 | 3,110E-03 |  |  |
|  |  | 10,00311041990660 | 3,110E-03 |  |  |
| 13 |  |  | 0,81818181818182 | 1,818E-01 |  |  |
|  |  | 1,81818181818182 | 1,818E-01 |  |  |
|  |  | 2,81818181818181 | 1,818E-01 |  |  |
|  |  | 3,81818181818181 | 1,818E-01 |  |  |
|  |  | 4,81818181818181 | 1,818E-01 |  |  |
|  |  | 5,81818181818181 | 1,818E-01 |  |  |
|  |  | 6,81818181818182 | 1,818E-01 |  |  |
|  |  | 7,81818181818181 | 1,818E-01 |  |  |
|  |  | 8,81818181818181 | 1,818E-01 |  |  |
|  |  | 9,81818181818181 | 1,818E-01 |  |  |
| 14 |  |  | 0,00000000000000 | 1,000E+00 |  |  |
|  |  | 1,00000000000000 | 1,000E+00 |  |  |
|  |  | 2,00000000000000 | 1,000E+00 |  |  |
|  |  | 3,00000000000000 | 1,000E+00 |  |  |
|  |  | 4,00000000000000 | 1,000E+00 |  |  |
|  |  | 5,00000000000000 | 1,000E+00 |  |  |
|  |  | 6,00000000000000 | 1,000E+00 |  |  |
|  |  | 7,00000000000000 | 1,000E+00 |  |  |
|  |  | 8,00000000000000 | 1,000E+00 |  |  |
|  |  | 9,00000000000000 | 1,000E+00 |  |  |

Исследование матриц с регулируемым числом обусловленностей показало следующие результаты:

* Двойная точность обеспечивает гораздо меньшую погрешность при вычислениях по сравнению с одинарной точностью.
* Подсчет скалярного произведения, реализованный в двойной точности, дал более точный результат по сравнению одинарной точностью, поскольку уменьшил погрешность вычисления.
* Если подойди аналитически к решению данного феномена, то зная, что у типа float всего лишь 7 (7.5) значащих знаков, то при достижении k, например равного 7, в памяти вместо числа 13.0000001 будет хранится всего лишь 13.00000. Если после теперь проверить матрицу А на невырожденность, то выясниться, что она вырождена, а это противоречит нашей модели. Аналогично и для double

1. **Исследование матриц Гильберта с различной размерностью**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| k | xk (один. точность) | |x\*-xk| (один. точность) | xk (дв. точность) | |x\*-xk| (дв. точность) | xk (скал. произв.) | | |x\*-xk| (скал. произв.) |
| 1 | 1,0000000 | 0,000E+00 | 1,00000000000000 | 0,000E+00 | 1,0000000 | | 0,000E+00 |
| 2 | 1,0000004 | 4,000E-07 | 1,00000000000000 | 0,000E+00 | 1,0000004 | | 4,000E-07 |
| 1,9999993 | 7,000E-07 | 1,99999999999999 | 9,992E-15 | 1,9999993 | | 7,000E-07 |
| 3 | 1,0000020 | 2,000E-06 | 1,00000000000000 | 0,000E+00 | 1,0000002 | | 2,000E-07 |
| 1,9999888 | 1,120E-05 | 1,99999999999995 | 4,996E-14 | 1,9999995 | | 5,000E-07 |
| 3,0000107 | 1,070E-05 | 3,00000000000004 | 3,997E-14 | 3,0000000 | | 0,000E+00 |
| 4 | 0,9999881 | 1,190E-05 | 1,00000000000006 | 5,995E-14 | 0,9999816 | | 1,840E-05 |
| 2,0001221 | 1,221E-04 | 1,99999999999923 | 7,701E-13 | 2,0001903 | | 1,903E-04 |
| 2,9997277 | 2,723E-04 | 3,00000000000188 | 1,880E-12 | 2,9995661 | | 4,339E-04 |
| 4,0001669 | 1,669E-04 | 3,99999999999875 | 1,250E-12 | 4,0002713 | | 2,713E-04 |
| 5 | 0,9999735 | 2,650E-05 | 1,00000000000020 | 2,001E-13 | 1,0000367 | | 3,670E-05 |
| 2,0005670 | 5,670E-04 | 1,99999999999608 | 3,920E-12 | 1,9993157 | | 6,843E-04 |
| 2,9974289 | 2,571E-03 | 3,00000000001745 | 1,745E-11 | 3,0030003 | | 3,000E-03 |
| 4,0039711 | 3,971E-03 | 3,99999999997299 | 2,701E-11 | 3,9953833 | | 4,617E-03 |
| 4,9980354 | 1,965E-03 | 5,00000000001346 | 1,346E-11 | 5,0022969 | | 2,297E-03 |
| 6 | 1,0005798 | 5,798E-04 | 1,00000000000007 | 6,994E-14 | 0,9995766 | | 4,234E-04 |
| 1,9862485 | 1,375E-02 | 2,00000000000057 | 5,702E-13 | 2,0147848 | | 1,478E-02 |
| 3,0808086 | 8,081E-02 | 2,99999999998465 | 1,535E-11 | 2,8886299 | | 1,114E-01 |
| 3,8124685 | 1,875E-01 | 4,00000000006091 | 6,091E-11 | 4,3101234 | | 3,101E-01 |
| 5,1881227 | 1,881E-01 | 4,99999999991529 | 8,471E-11 | 4,6410217 | | 3,590E-01 |
| 5,9317074 | 6,829E-02 | 6,00000000003877 | 3,877E-11 | 6,1465139 | | 1,465E-01 |
| 7 | 0,9974604 | 2,540E-03 | 0,99999999996237 | 3,763E-11 | 1,0036564 | | 3,656E-03 |
| 2,1022983 | 1,023E-01 | 2,00000000149128 | 1,491E-09 | 1,8512831 | | 1,487E-01 |
| 2,0161734 | 9,838E-01 | 2,99999998570114 | 1,430E-08 | 4,4428596 | | 1,443E+00 |
| 7,8061004 | 3,806E+00 | 4,00000005540247 | 5,540E-08 | -1,6213303 | | 5,621E+00 |
| -1,9383221 | 6,938E+00 | 4,99999989868240 | 1,013E-07 | 15,3048325 | | 1,030E+01 |
| 11,9620018 | 5,962E+00 | 6,00000008738392 | 8,738E-08 | -2,8947506 | | 8,895E+00 |
| 5,0526314 | 1,947E+00 | 6,99999997135027 | 2,865E-08 | 9,9160147 | | 2,916E+00 |
| 8 |  |  |  |  |  | |  |
| 1,0026910 | 2,691E-03 | 0,99999999994405 | 5,595E-11 | 0,9970577 | | 2,942E-03 |
| 1,7188716 | 2,811E-01 | 2,00000000325125 | 3,251E-09 | 2,1044617 | | 1,045E-01 |
| 7,4371758 | 4,437E+00 | 2,99999995499108 | 4,501E-08 | 2,1310921 | | 8,689E-01 |
| -21,6409607 | 2,564E+01 | 4,00000025452796 | 2,545E-07 | 6,6950002 | | 2,695E+00 |
| 75,0598145 | 7,006E+01 | 4,99999929087063 | 7,091E-07 | 1,9433756 | | 3,057E+00 |
| -92,1386566 | 9,814E+01 | 6,00000103128222 | 1,031E-06 | 5,5763111 | | 4,237E-01 |
| 75,2280731 | 6,823E+01 | 6,99999924942991 | 7,506E-07 | 10,0641518 | | 3,064E+00 |
| -10,6666670 | 1,867E+01 | 8,00000021576885 | 2,158E-07 | 6,4864092 | | 1,514E+00 |
| 9 | 0,9772214 | 2,278E-02 | 1,00000000078764 | 7,876E-10 | 0,9743173 | | 2,568E-02 |
| 2,8003037 | 8,003E-01 | 1,99999994555405 | 5,445E-08 | 3,1679173 | | 1,168E+00 |
| -3,5404968 | 6,540E+00 | 3,00000092126951 | 9,213E-07 | -9,6653214 | | 1,267E+01 |
| 22,5013065 | 1,850E+01 | 3,99999343103765 | 6,569E-06 | 57,9063873 | | 5,391E+01 |
| -3,4563446 | 8,456E+00 | 5,00002405680736 | 2,406E-05 | -93,0517273 | | 9,805E+01 |
| -45,6929932 | 5,169E+01 | 5,99995096397421 | 4,904E-05 | 50,3957825 | | 4,440E+01 |
| 109,0701981 | 1,021E+02 | 7,00005621865872 | 5,622E-05 | 96,2949219 | | 8,929E+01 |
| -65,6666718 | 7,367E+01 | 7,99996609989613 | 3,390E-05 | -113,7114868 | | 1,217E+02 |
| 28,0000000 | 1,900E+01 | 9,00000836254427 | 8,363E-06 | 52,6992035 | | 4,370E+01 |
| 10 |  |  | 1,00000000661383 | 6,614E-09 |  |  | |
|  |  | 1,99999940632912 | 5,937E-07 |  |  | |
|  |  | 3,00001300262259 | 1,300E-05 |  |  | |
|  |  | 3,99987924328774 | 1,208E-04 |  |  | |
|  |  | 5,00058578301321 | 5,858E-04 |  |  | |
|  |  | 5,99836753057913 | 1,632E-03 |  |  | |
|  |  | 7,00270880453366 | 2,709E-03 |  |  | |
|  |  | 7,99735729618038 | 2,643E-03 |  |  | |
|  |  | 9,00139865726001 | 1,399E-03 |  |  | |
|  |  | 9,99969026365293 | 3,097E-04 |  |  | |
| 11 |  |  | 1,00000000918692 | 9,187E-09 |  |  | |
|  |  | 1,99999920941194 | 7,906E-07 |  |  | |
|  |  | 3,00001677381544 | 1,677E-05 |  |  | |
|  |  | 3,99984798647462 | 1,520E-04 |  |  | |
|  |  | 5,00072368506224 | 7,237E-04 |  |  | |
|  |  | 5,99801085288293 | 1,989E-03 |  |  | |
|  |  | 7,00327236862120 | 3,272E-03 |  |  | |
|  |  | 7,99681271333029 | 3,187E-03 |  |  | |
|  |  | 9,00170693346601 | 1,707E-03 |  |  | |
|  |  | 9,99959954435299 | 4,005E-04 |  |  | |
|  |  | 11,00000991778150 | 9,918E-06 |  |  | |
| 12 |  |  | 1,00000050262499 | 5,026E-07 |  |  | |
|  |  | 1,99993686260080 | 6,314E-05 |  |  | |
|  |  | 3,00197056027053 | 1,971E-03 |  |  | |
|  |  | 3,97332634403008 | 2,667E-02 |  |  | |
|  |  | 5,19443072976356 | 1,944E-01 |  |  | |
|  |  | 5,14997416132602 | 8,500E-01 |  |  | |
|  |  | 9,35793099888848 | 2,358E+00 |  |  | |
|  |  | 3,74866865229763 | 4,251E+00 |  |  | |
|  |  | 13,96648768527710 | 4,966E+00 |  |  | |
|  |  | 6,37429602508643 | 3,626E+00 |  |  | |
|  |  | 12,50307363531880 | 1,503E+00 |  |  | |
|  |  | 11,72990353697740 | 2,701E-01 |  |  | |

Число обусловленности матрицы показывает то, насколько может измениться решение при небольшом изменении входных данных, то есть насколько матрица чувствительна к погрешности во входных данных. Как правило, если число обусловленности , то можно потерять до “p” цифр точности сверх того, что будет потеряно для числового значения из-за арифметических методов. Однако число обусловленности не дает точного значения максимальной неточности, которая может возникнуть в алгоритме.

Один из способов вычисления числа обусловленности матрицы:

Оценим число обусловленности для матриц Гильберта различной размерности:



Теперь обратившись к полученной таблице, мы заметим, что при k=7 решение уже не получено, а для двойной при k=12 соответственно. То есть, если порядок числа обусловленности для некоторой матрицы превышает максимальное количество значащих знаков в формате (float/double), то можно сказать, что решения нет.

Двойная точность обеспечивает гораздо меньшую погрешность при вычислениях по сравнению с одинарной точностью.

Подсчет скалярного произведения, реализованный в двойной точности, дал более точный результат по сравнению одинарной точностью, поскольку уменьшил погрешность вычисления.

1. **Сравнение метода Гаусса с прямым методом**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | LU | Гаусс | LU | Гаусс | LU | Гаусс |
| k | |x\*-xk| (один. точность) | |x\*-xk| (один. точность) | |x\*-xk| (дв. точность) | |x\*-xk| (дв. точность) | |x\*-xk| (скал. произв.) | |x\*-xk| (скал. произв.) |
| 0 | 1,280E-05 | 8,140E-05 | 9,992E-15 | 3,797E-14 | 8,800E-06 | 8,140E-05 |
| 1,340E-05 | 8,490E-05 | 1,021E-14 | 4,996E-14 | 9,800E-06 | 8,490E-05 |
| 1,360E-05 | 8,630E-05 | 1,021E-14 | 5,018E-14 | 9,100E-06 | 8,630E-05 |
| 1,340E-05 | 8,630E-05 | 9,770E-15 | 5,018E-14 | 9,500E-06 | 8,630E-05 |
| 1,340E-05 | 9,010E-05 | 9,770E-15 | 3,997E-14 | 9,500E-06 | 9,010E-05 |
| 1,340E-05 | 8,960E-05 | 9,770E-15 | 4,974E-14 | 9,500E-06 | 8,960E-05 |
| 1,340E-05 | 8,580E-05 | 9,770E-15 | 4,974E-14 | 9,500E-06 | 8,580E-05 |
| 1,340E-05 | 8,580E-05 | 1,066E-14 | 4,974E-14 | 9,500E-06 | 8,580E-05 |
| 1,430E-05 | 8,580E-05 | 1,066E-14 | 4,974E-14 | 9,500E-06 | 8,580E-05 |
| 1,340E-05 | 8,580E-05 | 0,000E+00 | 4,974E-14 | 9,500E-06 | 8,580E-05 |
| 1 | 8,400E-05 | 4,309E-04 | 1,890E-13 | 1,504E-12 | 4,600E-05 | 4,309E-04 |
| 8,440E-05 | 4,339E-04 | 1,901E-13 | 1,520E-12 | 4,580E-05 | 4,339E-04 |
| 8,440E-05 | 4,344E-04 | 1,901E-13 | 1,520E-12 | 4,580E-05 | 4,344E-04 |
| 8,440E-05 | 4,339E-04 | 1,901E-13 | 1,520E-12 | 4,580E-05 | 4,339E-04 |
| 8,440E-05 | 4,339E-04 | 1,901E-13 | 1,520E-12 | 4,580E-05 | 4,339E-04 |
| 8,490E-05 | 4,358E-04 | 1,901E-13 | 1,530E-12 | 4,580E-05 | 4,358E-04 |
| 8,440E-05 | 4,330E-04 | 1,901E-13 | 1,520E-12 | 4,580E-05 | 4,330E-04 |
| 8,390E-05 | 4,358E-04 | 1,901E-13 | 1,520E-12 | 4,580E-05 | 4,358E-04 |
| 8,490E-05 | 4,349E-04 | 1,901E-13 | 1,521E-12 | 4,580E-05 | 4,349E-04 |
| 8,490E-05 | 4,339E-04 | 1,901E-13 | 1,521E-12 | 4,580E-05 | 4,339E-04 |
| 2 | 1,002E-03 | 5,372E-04 | 9,299E-13 | 1,003E-11 | 6,994E-04 | 5,372E-04 |
| 1,003E-03 | 5,376E-04 | 9,299E-13 | 1,004E-11 | 6,998E-04 | 5,376E-04 |
| 1,003E-03 | 5,374E-04 | 9,299E-13 | 1,004E-11 | 7,000E-04 | 5,374E-04 |
| 1,003E-03 | 5,374E-04 | 9,299E-13 | 1,004E-11 | 7,000E-04 | 5,374E-04 |
| 1,003E-03 | 5,345E-04 | 9,299E-13 | 1,004E-11 | 7,000E-04 | 5,345E-04 |
| 1,003E-03 | 5,398E-04 | 9,299E-13 | 1,005E-11 | 7,000E-04 | 5,398E-04 |
| 1,002E-03 | 5,369E-04 | 9,299E-13 | 1,004E-11 | 7,000E-04 | 5,369E-04 |
| 1,003E-03 | 5,379E-04 | 9,308E-13 | 1,004E-11 | 7,000E-04 | 5,379E-04 |
| 1,003E-03 | 5,379E-04 | 9,308E-13 | 1,004E-11 | 7,000E-04 | 5,379E-04 |
| 1,003E-03 | 5,388E-04 | 9,006E-13 | 1,004E-11 | 7,000E-04 | 5,388E-04 |
| 3 | 1,168E-02 | 1,428E-01 | 1,865E-11 | 1,806E-10 | 5,405E-03 | 1,428E-01 |
| 1,168E-02 | 1,429E-01 | 1,865E-11 | 1,806E-10 | 5,405E-03 | 1,429E-01 |
| 1,168E-02 | 1,429E-01 | 1,865E-11 | 1,806E-10 | 5,405E-03 | 1,429E-01 |
| 1,168E-02 | 1,429E-01 | 1,865E-11 | 1,806E-10 | 5,405E-03 | 1,429E-01 |
| 1,168E-02 | 1,429E-01 | 1,865E-11 | 1,806E-10 | 5,406E-03 | 1,429E-01 |
| 1,168E-02 | 1,429E-01 | 1,865E-11 | 1,806E-10 | 5,405E-03 | 1,429E-01 |
| 1,168E-02 | 1,429E-01 | 1,865E-11 | 1,806E-10 | 5,406E-03 | 1,429E-01 |
| 1,168E-02 | 1,429E-01 | 1,865E-11 | 1,806E-10 | 5,405E-03 | 1,429E-01 |
| 1,168E-02 | 1,429E-01 | 1,865E-11 | 1,806E-10 | 5,405E-03 | 1,429E-01 |
| 1,168E-02 | 1,429E-01 | 1,860E-11 | 1,806E-10 | 5,405E-03 | 1,429E-01 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 1,639E-01 | 2,778E-01 | 6,219E-11 | 3,110E-09 | 4,877E-02 | 2,778E-01 |
| 1,639E-01 | 2,778E-01 | 6,220E-11 | 3,110E-09 | 4,877E-02 | 2,778E-01 |
| 1,639E-01 | 2,778E-01 | 6,220E-11 | 3,110E-09 | 4,876E-02 | 2,778E-01 |
| 1,639E-01 | 2,778E-01 | 6,220E-11 | 3,110E-09 | 4,877E-02 | 2,778E-01 |
| 1,639E-01 | 2,778E-01 | 6,220E-11 | 3,110E-09 | 4,877E-02 | 2,778E-01 |
| 1,639E-01 | 2,778E-01 | 6,220E-11 | 3,110E-09 | 4,877E-02 | 2,778E-01 |
| 1,639E-01 | 2,778E-01 | 6,220E-11 | 3,110E-09 | 4,877E-02 | 2,778E-01 |
| 1,639E-01 | 2,778E-01 | 6,220E-11 | 3,110E-09 | 4,877E-02 | 2,778E-01 |
| 1,639E-01 | 2,778E-01 | 6,220E-11 | 3,110E-09 | 4,877E-02 | 2,778E-01 |
| 1,639E-01 | 2,778E-01 | 6,220E-11 | 3,110E-09 | 4,876E-02 | 2,778E-01 |
| 5 | 4,000E-07 | 2,000E+00 | 9,992E-16 | 2,207E-08 | 8,031E-01 | 2,000E+00 |
| 0,000E+00 | 2,000E+00 | 9,992E-15 | 2,207E-08 | 8,031E-01 | 2,000E+00 |
| 2,000E-07 | 2,000E+00 | 0,000E+00 | 2,207E-08 | 8,031E-01 | 2,000E+00 |
| 2,000E-07 | 2,000E+00 | 1,021E-14 | 2,207E-08 | 8,031E-01 | 2,000E+00 |
| 5,000E-07 | 2,000E+00 | 9,770E-15 | 2,207E-08 | 8,031E-01 | 2,000E+00 |
| 0,000E+00 | 2,000E+00 | 9,770E-15 | 2,207E-08 | 8,031E-01 | 2,000E+00 |
| 0,000E+00 | 2,000E+00 | 0,000E+00 | 2,207E-08 | 8,031E-01 | 2,000E+00 |
| 0,000E+00 | 2,000E+00 | 9,770E-15 | 2,207E-08 | 8,031E-01 | 2,000E+00 |
| 0,000E+00 | 2,000E+00 | 0,000E+00 | 2,207E-08 | 8,031E-01 | 2,000E+00 |
| 0,000E+00 | 2,000E+00 | 0,000E+00 | 2,207E-08 | 8,031E-01 | 2,000E+00 |
| 6 | 2,000E+00 | 1,050E+01 | 2,488E-08 | 1,204E-07 | 1,299E+01 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 2,488E-08 | 1,204E-07 | 1,299E+01 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 2,488E-08 | 1,204E-07 | 1,299E+01 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 2,488E-08 | 1,204E-07 | 1,299E+01 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 2,488E-08 | 1,204E-07 | 1,299E+01 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 2,488E-08 | 1,204E-07 | 1,299E+01 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 2,488E-08 | 1,204E-07 | 1,299E+01 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 2,488E-08 | 1,204E-07 | 1,299E+01 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 2,488E-08 | 1,204E-07 | 1,299E+01 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 2,488E-08 | 1,204E-07 | 1,299E+01 | 1,050E+01 |
| 7 | 2,000E+00 | 1,050E+01 | 4,353E-07 | 0,000E+00 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 4,353E-07 | 0,000E+00 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 4,353E-07 | 1,021E-14 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 4,353E-07 | 0,000E+00 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 4,353E-07 | 0,000E+00 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 4,353E-07 | 9,770E-15 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 4,353E-07 | 9,770E-15 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 4,353E-07 | 9,770E-15 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 4,353E-07 | 0,000E+00 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 4,353E-07 | 0,000E+00 | 4,406E+00 | 1,050E+01 |
| 8 | 2,000E+00 | 1,050E+01 | 1,244E-06 | 5,017E-06 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 1,244E-06 | 5,017E-06 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 1,244E-06 | 5,017E-06 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 1,244E-06 | 5,017E-06 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 1,244E-06 | 5,017E-06 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 1,244E-06 | 5,017E-06 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 1,244E-06 | 5,017E-06 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 1,244E-06 | 5,017E-06 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 1,244E-06 | 5,017E-06 | 4,406E+00 | 1,050E+01 |
| 2,000E+00 | 1,050E+01 | 1,244E-06 | 5,017E-06 | 4,406E+00 | 1,050E+01 |
| 9 |  |  | 1,244E-05 | 1,706E-04 |  |  |
|  |  | 1,244E-05 | 1,706E-04 |  |  |
|  |  | 1,244E-05 | 1,706E-04 |  |  |
|  |  | 1,244E-05 | 1,706E-04 |  |  |
|  |  | 1,244E-05 | 1,706E-04 |  |  |
|  |  | 1,244E-05 | 1,706E-04 |  |  |
|  |  | 1,244E-05 | 1,706E-04 |  |  |
|  |  | 1,244E-05 | 1,706E-04 |  |  |
|  |  | 1,244E-05 | 1,706E-04 |  |  |
|  |  | 1,244E-05 | 1,706E-04 |  |  |
| 10 |  |  | 9,329E-05 | 2,407E-03 |  |  |
|  |  | 9,329E-05 | 2,407E-03 |  |  |
|  |  | 9,329E-05 | 2,407E-03 |  |  |
|  |  | 9,329E-05 | 2,407E-03 |  |  |
|  |  | 9,329E-05 | 2,407E-03 |  |  |
|  |  | 9,329E-05 | 2,407E-03 |  |  |
|  |  | 9,329E-05 | 2,407E-03 |  |  |
|  |  | 9,329E-05 | 2,407E-03 |  |  |
|  |  | 9,329E-05 | 2,407E-03 |  |  |
|  |  | 9,329E-05 | 2,407E-03 |  |  |
| 11 |  |  | 1,244E-03 | 9,036E-03 |  |  |
|  |  | 1,244E-03 | 9,036E-03 |  |  |
|  |  | 1,244E-03 | 9,036E-03 |  |  |
|  |  | 1,244E-03 | 9,036E-03 |  |  |
|  |  | 1,244E-03 | 9,036E-03 |  |  |
|  |  | 1,244E-03 | 9,036E-03 |  |  |
|  |  | 1,244E-03 | 9,036E-03 |  |  |
|  |  | 1,244E-03 | 9,036E-03 |  |  |
|  |  | 1,244E-03 | 9,036E-03 |  |  |
|  |  | 1,244E-03 | 9,036E-03 |  |  |
| 12 |  |  | 3,110E-03 | 1,000E+01 |  |  |
|  |  | 3,110E-03 | 1,000E+01 |  |  |
|  |  | 3,110E-03 | 1,000E+01 |  |  |
|  |  | 3,110E-03 | 1,000E+01 |  |  |
|  |  | 3,110E-03 | 1,000E+01 |  |  |
|  |  | 3,110E-03 | 1,000E+01 |  |  |
|  |  | 3,110E-03 | 1,000E+01 |  |  |
|  |  | 3,110E-03 | 1,000E+01 |  |  |
|  |  | 3,110E-03 | 1,000E+01 |  |  |
|  |  | 3,110E-03 | 1,000E+01 |  |  |

В большинстве тестах Гаусс оказывался на 1 или более порядков менее точным, чем LU разложение. Из 31 вычислений при помощи LU и 31 при помощи Гаусса лишь в 4-ех Гаусс оказался точнее, но это можно списать на случайность («самоустранение» накопительной погрешности при вычислении).

1. **Расчет количества действий**

LU-разложение

LU-разложение: 

Прямой ход: 

Обратный ход: 

Итого: Сложность LU-разложения: 



Гаусс  
Прямой ход: 

Обратный ход: 

Итого: Сложность Гаусса 

LDU-разложение

Разложение L и U: 

Разложение D: 

Прямой ход: 

Центральный ход: 

Обратный ход: 

Итого: Сложность LDU-разложения: 

**Вывод по работе:**

LU-разложение и Гаусс во многом похожи. LU-разложение представляет из себя две матрицы: U – верхнетреугольная матрица полученная преобразованиями, коэффициенты преобразования которой хранятся в матрице L.

Исходя из вышесказанного и полученных алгоритмов сложности можно сделать вывод о том, что количество операций, необходимых для решения СЛАУ LU-разложением, почти такое же как у Гаусса. Отличие в том, что если понадобиться решить последовательно несколько систем с одинаковыми матрицами А, но разной правой частью, то при помощи LU-разложения можно ускорить эти расчеты, путем того, что не надо будет многократно приводить матрицу к верхнетреугольному виду. Достоинством Гаусса помимо того, что он является аналитически более простым, является то, что при возникновении нулей на диагонали, в том числе вовремя решения, Гаусс все так же работает из-за того, что использует выбор ведущего элемента.

По результатам тестирования LU-разложение оказалось более точным чем Гаусс.

**Приложение**

Matrix.h

#pragma once

#include <cstdio>

#include <math.h>

typedef float real;

typedef double realSum;

#define REALIN "%f"

#define REALOUT "%.7f\n"

#define REALOUTD "%.4f\t"

#define EPS 1e-13

class BaseMatrix {

public:

int n;

real \*x, \*b;

virtual void Input(FILE\* matrixFile, FILE\* vectorFile) {}

virtual void SolveSLAE() {}

virtual void OutputDense() {}

virtual void GenerateHilbertMatrix(int n) {}

void OutputSolutionVector(FILE\* out);

protected:

virtual void AllocateMemory() {}

};

class ProfileMatrix : public virtual BaseMatrix {

public:

real \*al, \*au, \*di;

int \*ia, nProfile;

void Input(FILE\* matrixFile, FILE\* vectorFile);

void GenerateHilbertMatrix(int n);

void SolveSLAE();

void OutputDense();

ProfileMatrix() {};

private:

real \*y;

void AllocateMemory();

void CalculateLU();

void CalculateY();

void CalculateX();

};

class DenseMatrix : public virtual BaseMatrix {

public:

real \*\*matrix;

void Input(FILE\* matrixFile, FILE\* vectorFile);

void SolveSLAE();

void OutputDense();

private:

void AllocateMemory();

void GaussForwardElimination();

void GaussBackSubstitution();

};

Matrix.cpp

#include "Matrix.h"

void ProfileMatrix::Input(FILE\* matrixFile, FILE\* vectorFile) {

fscanf\_s(matrixFile, "%d", &n);

ia = new int[n + 1];

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

fscanf\_s(matrixFile, "%d", &ia[i]);

nProfile = ia[n] - ia[0];

AllocateMemory();

if (ia[0]) {

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

ia[i]--;

}

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

fscanf\_s(matrixFile, REALIN, &al[i]);

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

fscanf\_s(matrixFile, REALIN, &au[i]);

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

fscanf\_s(matrixFile, REALIN, &di[i]);

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

fscanf\_s(vectorFile, REALIN, &b[i]);

}

void ProfileMatrix::AllocateMemory() {

al = new real[nProfile];

au = new real[nProfile];

di = new real[n];

b = new real[n];

x = b;

y = b;

}

void ProfileMatrix::CalculateLU()

{

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

{

int i0 = ia[i];

int i1 = ia[i + 1];

realSum sumD = 0;

int j = i - (i1 - i0);

for (int k = i0; k < i1; j++, k++)

{

realSum sumL = 0, sumU = 0;

int j0 = ia[j], j1 = ia[j + 1];

int difI = i1 - i0 - (i - j);

int difJ = j1 - j0;

int difIJ = difI - difJ;

int ki = i0, kj = j0;

if (difIJ < 0)

kj -= difIJ;

else

ki += difIJ;

for (; ki < k; ki++, kj++)

{

sumL += al[ki] \* au[kj];

sumU += al[kj] \* au[ki];

}

al[k] = al[k] - sumL;

au[k] = (au[k] - sumU) / di[j];

sumD += al[k] \* au[k];

}

di[i] -= sumD;

}

}

void ProfileMatrix::CalculateY() {

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

{

realSum sum = 0;

int i0 = ia[i], i1 = ia[i + 1];

int j = i - (i1 - i0);

for (int k = i0; k < i1; k++, j++)

sum += y[j] \* al[k];

y[i] = (b[i] - sum) / di[i];

}

}

void ProfileMatrix::CalculateX() {

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

{

int i0 = ia[i], i1 = ia[i + 1];

int j = i - (i1 - i0);

real xi = y[i];

for (int ij = i0; ij < i1; ij++, j++)

y[j] -= au[ij] \* xi;

x[i] = xi;

}

}

void ProfileMatrix::GenerateHilbertMatrix(int size)

{

n = size;

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

{

nProfile += i;

}

ia = new int[n + 1];

AllocateMemory();

ia[0] = 0;

for (int i = 1, k = 0; i <= n; i++)

{

ia[i] = ia[i - 1] + (i - 1);

di[i-1] = (double)1 / (2\*i - 1);

for (int j = 1; j < i; j++, k++)

{

al[k] = (double)1 / (i + j - 1);

au[k] = (double)1 / (i + j - 1);

}

}

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

{

double sum = 0;

for (int xk = 1; xk <= n; xk++)

{

sum += (double)1 / (i + xk) \* xk;

}

b[i] = sum;

}

}

void ProfileMatrix::SolveSLAE() {

CalculateLU();

CalculateY();

CalculateX();

}

void ProfileMatrix::OutputDense()

{

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

{

for (int j = 0; j < (i - (ia[i + 1] - ia[i])); j++)

{

printf(REALOUTD, 0.0);

}

for (int j = ia[i]; j < ia[i + 1]; j++)

{

printf(REALOUTD, al[j]);

}

printf(REALOUTD, di[i]);

for (int j = i + 1; j < n; j++)

{

int k = ia[j + 1] - ia[j];

if ((j - i) <= k) {

printf(REALOUTD, au[ia[j] + k + i - j]);

}

else

{

printf(REALOUTD, 0.0);

}

}

printf("\n");

}

}

void BaseMatrix::OutputSolutionVector(FILE\* out) {

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

fprintf\_s(out, REALOUT, x[i]);

}

void DenseMatrix::Input(FILE\* matrixFile, FILE\* vectorFile)

{

fscanf\_s(matrixFile, "%d", &n);

AllocateMemory();

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

{

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

fscanf\_s(matrixFile, REALIN, &matrix[i][j]);

}

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

fscanf\_s(vectorFile, REALIN, &b[i]);

}

void DenseMatrix::SolveSLAE()

{

GaussForwardElimination();

GaussBackSubstitution();

}

void DenseMatrix::OutputDense()

{

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

{

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

printf\_s(REALOUTD, matrix[i][j]);

printf\_s("\n");

}

}

void DenseMatrix::AllocateMemory()

{

matrix = new real\*[n];

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

matrix[i] = new real[n];

}

b = new real[n];

x = b;

}

void DenseMatrix::GaussForwardElimination()

{

real max;

int indexMax;

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

{

// Поиск строки с максимальным a[i][k]

max = fabsf(matrix[k][k]);

indexMax = k;

for (int i = k + 1; i < n; i++)

{

if (fabsf(matrix[i][k]) > max)

{

max = fabsf(matrix[i][k]);

indexMax = i;

}

}

// Перестановка строк

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

{

real temp = matrix[k][j];

matrix[k][j] = matrix[indexMax][j];

matrix[indexMax][j] = temp;

}

real temp = b[k];

b[k] = b[indexMax];

b[indexMax] = temp;

// Нормализация уравнений

for (int i = k; i < n; i++)

{

real temp = matrix[i][k];

if (fabsf(temp) < EPS)

continue; // для нулевого коэффициента пропустить

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

matrix[i][j] = matrix[i][j] / temp;

b[i] /= temp;

if (i == k)

continue; // уравнение не вычитать само из себя

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

matrix[i][j] -= matrix[k][j];

b[i] -= b[k];

}

}

}

void DenseMatrix::GaussBackSubstitution()

{

for (int k = n - 1; k >= 0; k--)

{

x[k] = b[k];

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

b[i] -= matrix[i][k] \* x[k];

}

}

PZ1.cpp

#include <iostream>

#include "Matrix.h"

#include <math.h>

int main()

{

setlocale(LC\_ALL, "Russian");

FILE \*inputMatrix, \*inputVector;

int menuInput;

printf\_s("1 - Считать профильную матрицу из файла и решить ее с помощью LU разложения\n2 - Считать плотную матрицу из файла и решить ее с помощью метода Гаусса\n3 - Сгенерировать матрирцу Гильберта и решить ее с помощью LU разложения\n");

scanf\_s("%d", &menuInput);

if (menuInput == 1) {

ProfileMatrix matrix;

int numOfCondition;

fopen\_s(&inputMatrix, "matrix.txt", "r");

fopen\_s(&inputVector, "vector.txt", "r");

matrix.Input(inputMatrix, inputVector);

fclose(inputMatrix);

fclose(inputVector);

printf\_s("Исследовать матрицу с регулируемым числом обусловленности?\n");

scanf\_s("%d", &numOfCondition);

if (numOfCondition)

{

int k;

printf\_s("Какой k для a(11)+10^(-k)?\n");

scanf\_s("%d", &k);

real tmp = pow(10.0f, -k);

matrix.di[0] += tmp;

matrix.b[0] += tmp;

}

matrix.OutputDense();

matrix.SolveSLAE();

printf("\n");

//matrix.OutputDense();

FILE\* out;

fopen\_s(&out, "out.txt", "w");

matrix.OutputSolutionVector(out);

fclose(out);

}

else if (menuInput == 2) {

DenseMatrix matrix;

int numOfCondition;

fopen\_s(&inputMatrix, "matrixDense.txt", "r");

fopen\_s(&inputVector, "vectorDA.txt", "r");

matrix.Input(inputMatrix, inputVector);

fclose(inputMatrix);

fclose(inputVector);

printf\_s("Исследовать матрицу с регулируемым числом обусловленности?\n");

scanf\_s("%d", &numOfCondition);

if (numOfCondition)

{

int k;

printf\_s("Какой k для a(11)+10^(-k)?\n");

scanf\_s("%d", &k);

real tmp = pow(10.0f, -k);

matrix.matrix[0][0] += tmp;

matrix.b[0] += tmp;

}

matrix.OutputDense();

matrix.SolveSLAE();

FILE\* out;

fopen\_s(&out, "out.txt", "w");

matrix.OutputSolutionVector(out);

fclose(out);

}

else if (menuInput == 3) {

ProfileMatrix matrix;

int m;

printf\_s("Введите размерность: ");

scanf\_s("%d", &m);

matrix.GenerateHilbertMatrix(m);

//matrix.OutputDense();

matrix.SolveSLAE();

printf("\n");

//matrix.OutputDense();

FILE\* out2;

fopen\_s(&out2, "out2.txt", "w");

matrix.OutputSolutionVector(out2);

fclose(out2);

}

}