|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| 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-81 |
| Бригада | 2 |
| Студенты | Басов денис |
|  | юрганов егор |
|  |  |
| Преподаватель | задорожный а.г. |
| Дата | 29.10.2020 |
|  | | | |
| Новосибирск | | | |

1. Цель

Разработать программу решения СЛАУ методом с LLT-разложением и хранением матрицы в 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. Анализ

Пусть необходимо решить СЛАУ вида:

(1.1)

Тогда матрица А может быть представлена в виде:

(1.2)

Подставляя (1.2) в (1.1), получаем:

(1.3)

Обозначим:

(1.4)

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

(1.5)

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

1) из элементов матрицы  найти элементы матриц L(LT) ;

2) решить систему (1.5) с нижнетреугольной матрицей L (прямой ход);

3) решить систему (1.4) с верхнетреугольной матрицей LT (обратный ход).

1. *Разложение*

Рассмотрим алгоритм получения LLT – разложения. Матрицы L и LT будем искать в следующем виде:

, (1.6)

1. *Прямой проход*

Учитывая равенство (1.2) и умножая последовательно строки матрицы L на столбцы матрицы LT, получаем систему, состоящую из уравнений с неизвестными, т.к.

Решая данную систему, можно получить общие формулы для нахождения элементов матриц L и LT:

, 1 ≤ ≤ n; (1.8)

1. *Обратный проход*

Учитывая равенства (1.5) ,(1.4) и умножая последовательно строки матриц на вектора, получаем системы:

Последовательно решаем по формулам:

1 ≤ ≤ n (“сверху вниз”) ;

, 1 ≤ ≤ n(“снизу вверх”)

1. Влияние увеличения числа обусловленности на точность решения

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 16+10-k | -1 | -2 | 0 | -3 | 0 | -1 | -4 | -3 | -2 |
| -1 | 18 | -4 | -4 | -3 | 0 | 0 | -2 | 0 | -4 |
| -2 | -4 | 20 | -4 | -2 | -1 | -2 | -3 | -2 | 0 |
| 0 | -4 | -4 | 23 | -4 | -2 | -2 | -2 | -1 | -4 |
| -3 | -3 | -2 | -4 | 20 | -1 | -1 | -2 | -1 | -3 |
| 0 | 0 | -1 | -2 | -1 | 10 | -1 | -1 | -3 | -1 |
| -1 | 0 | -2 | -2 | -1 | -1 | 14 | -1 | -4 | -2 |
| -4 | -2 | -3 | -2 | -2 | -1 | -1 | 20 | -2 | -3 |
| -3 | 0 | -2 | -1 | -1 | -3 | -4 | -2 | 19 | -3 |
| -2 | -4 | 0 | -4 | -3 | -1 | -2 | -3 | -3 | 22 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

|  |
| --- |
| -92-10-k |
| -64 |
| -38 |
| -39 |
| 1 |
| -8 |
| 8 |
| 64 |
| 61 |
| 108 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| k | xk (один. точность) | x\*-xk (один. точность) | xk (двой. точность) | x\*-xk (двой. точность) | xk (скалярное пр.) | x\*-xk (скалярное пр.) |
| 1 | 1.000278 | -0.000278 | 0.999999999998987 | 0.000000000001013 | 1.000289 | -0.000289 |
|  | 2.000280 | -0.000280 | 1.999999999998980 | 0.000000000001020 | 2.000290 | -0.000290 |
|  | 3.000280 | -0.000280 | 2.999999999998981 | 0.000000000001019 | 3.000291 | -0.000291 |
|  | 4.000280 | -0.000280 | 3.999999999998980 | 0.000000000001020 | 4.000290 | -0.000290 |
|  | 5.000280 | -0.000280 | 4.999999999998982 | 0.000000000001018 | 5.000291 | -0.000291 |
|  | 6.000281 | -0.000281 | 5.999999999998980 | 0.000000000001020 | 6.000292 | -0.000292 |
|  | 7.000280 | -0.000280 | 6.999999999998980 | 0.000000000001020 | 7.000291 | -0.000291 |
|  | 8.000280 | -0.000280 | 7.999999999998982 | 0.000000000001018 | 8.000291 | -0.000291 |
|  | 9.000280 | -0.000280 | 8.999999999998982 | 0.000000000001018 | 9.000291 | -0.000291 |
|  | 10.000280 | -0.000280 | 9.999999999998979 | 0.000000000001021 | 10.000291 | -0.000291 |
| 2 | 1.003246 | -0.003246 | 0.999999999996091 | 0.000000000003909 | 1.002825 | -0.002825 |
|  | 2.003248 | -0.003248 | 1.999999999996089 | 0.000000000003911 | 2.002826 | -0.002826 |
|  | 3.003248 | -0.003248 | 2.999999999996088 | 0.000000000003912 | 3.002826 | -0.002826 |
|  | 4.003247 | -0.003247 | 3.999999999996088 | 0.000000000003912 | 4.002826 | -0.002826 |
|  | 5.003247 | -0.003247 | 4.999999999996088 | 0.000000000003912 | 5.002826 | -0.002826 |
|  | 6.003247 | -0.003247 | 5.999999999996088 | 0.000000000003912 | 6.002825 | -0.002825 |
|  | 7.003247 | -0.003247 | 6.999999999996088 | 0.000000000003912 | 7.002826 | -0.002826 |
|  | 8.003248 | -0.003248 | 7.999999999996089 | 0.000000000003911 | 8.002827 | -0.002827 |
|  | 9.003247 | -0.003247 | 8.999999999996088 | 0.000000000003912 | 9.002826 | -0.002826 |
|  | 10.003247 | -0.003247 | 9.999999999996090 | 0.000000000003910 | 10.002826 | -0.002826 |
| 3 | 1.004769 | -0.004769 | 1.000000000024869 | -0.000000000024869 | 1.009256 | -0.009256 |
|  | 2.004770 | -0.004770 | 2.000000000024869 | -0.000000000024869 | 2.009257 | -0.009257 |
|  | 3.004769 | -0.004769 | 3.000000000024870 | -0.000000000024870 | 3.009257 | -0.009257 |
|  | 4.004769 | -0.004769 | 4.000000000024869 | -0.000000000024869 | 4.009257 | -0.009257 |
|  | 5.004769 | -0.004769 | 5.000000000024870 | -0.000000000024870 | 5.009257 | -0.009257 |
|  | 6.004769 | -0.004769 | 6.000000000024869 | -0.000000000024869 | 6.009257 | -0.009257 |
|  | 7.004769 | -0.004769 | 7.000000000024870 | -0.000000000024870 | 7.009256 | -0.009256 |
|  | 8.004770 | -0.004770 | 8.000000000024871 | -0.000000000024871 | 8.009257 | -0.009257 |
|  | 9.004769 | -0.004769 | 9.000000000024869 | -0.000000000024869 | 9.009256 | -0.009256 |
|  | 10.004770 | -0.004770 | 10.000000000024871 | -0.000000000024871 | 10.009257 | -0.009257 |
| 4 | 1.086538 | -0.086538 | 1.000000000852656 | -0.000000000852656 | 1.017126 | -0.017126 |
|  | 2.086538 | -0.086538 | 2.000000000852661 | -0.000000000852661 | 2.017126 | -0.017126 |
|  | 3.086538 | -0.086538 | 3.000000000852661 | -0.000000000852661 | 3.017125 | -0.017125 |
|  | 4.086538 | -0.086538 | 4.000000000852661 | -0.000000000852661 | 4.017126 | -0.017126 |
|  | 5.086538 | -0.086538 | 5.000000000852659 | -0.000000000852659 | 5.017126 | -0.017126 |
|  | 6.086538 | -0.086538 | 6.000000000852661 | -0.000000000852661 | 6.017126 | -0.017126 |
|  | 7.086538 | -0.086538 | 7.000000000852662 | -0.000000000852662 | 7.017126 | -0.017126 |
|  | 8.086538 | -0.086538 | 8.000000000852660 | -0.000000000852660 | 8.017126 | -0.017126 |
|  | 9.086538 | -0.086538 | 9.000000000852662 | -0.000000000852662 | 9.017126 | -0.017126 |
|  | 10.086538 | -0.08653 | 10.000000000852662 | -0.000000000852662 | 10.017126 | -0.017126 |
| 5 | 2.833331 | -1.833331 | 0.999999994670929 | 0.000000005329071 | 4.651037 | -3.651037 |
|  | 3.833332 | -1.833332 | 1.999999994670926 | 0.000000005329074 | 5.651039 | -3.651039 |
|  | 4.833332 | -1.833332 | 2.999999994670926 | 0.000000005329074 | 6.651039 | -3.651039 |
|  | 5.833331 | -1.833331 | 3.999999994670925 | 0.000000005329075 | 7.651039 | -3.651039 |
|  | 6.833332 | -1.833332 | 4.999999994670925 | 0.000000005329075 | 8.651039 | -3.651039 |
|  | 7.833331 | -1.833331 | 5.999999994670925 | 0.000000005329075 | 9.651038 | -3.651038 |
|  | 8.833332 | -1.833332 | 6.999999994670925 | 0.000000005329075 | 10.651039 | -3.651039 |
|  | 9.833332 | -1.833332 | 7.999999994670926 | 0.000000005329074 | 11.651038 | -3.651038 |
|  | 10.833331 | -1.833331 | 8.999999994670926 | 0.000000005329074 | 12.651037 | -3.651037 |
|  | 11.833333 | -1.833333 | 9.999999994670924 | 0.000000005329076 | 13.651040 | -3.651040 |
| 6 |  |  | 0.999999962696508 | 0.000000037303492 |  |  |
|  |  |  | 1.999999962696505 | 0.000000037303495 |  |  |
|  |  |  | 2.999999962696505 | 0.000000037303495 |  |  |
|  |  |  | 3.999999962696506 | 0.000000037303494 |  |  |
|  |  |  | 4.999999962696506 | 0.000000037303494 |  |  |
|  |  |  | 5.999999962696505 | 0.000000037303495 |  |  |
|  |  |  | 6.999999962696505 | 0.000000037303495 |  |  |
|  |  |  | 7.999999962696505 | 0.000000037303495 |  |  |
|  |  |  | 8.999999962696505 | 0.000000037303495 |  |  |
|  |  |  | 9.999999962696503 | 0.000000037303497 |  |  |
| 7 |  |  | 0.999999964472861 | 0.000000035527139 |  |  |
|  |  |  | 1.999999964472861 | 0.000000035527139 |  |  |
|  |  |  | 2.999999964472861 | 0.000000035527139 |  |  |
|  |  |  | 3.999999964472860 | 0.000000035527140 |  |  |
|  |  |  | 4.999999964472861 | 0.000000035527139 |  |  |
|  |  |  | 5.999999964472861 | 0.000000035527139 |  |  |
|  |  |  | 6.999999964472862 | 0.000000035527138 |  |  |
|  |  |  | 7.999999964472862 | 0.000000035527138 |  |  |
|  |  |  | 8.999999964472861 | 0.000000035527139 |  |  |
|  |  |  | 9.999999964472861 | 0.000000035527139 |  |  |
| 8 |  |  | 1.000002842171716 | -0.000002842171716 |  |  |
|  |  |  | 2.000002842171718 | -0.000002842171718 |  |  |
|  |  |  | 3.000002842171718 | -0.000002842171718 |  |  |
|  |  |  | 4.000002842171718 | -0.000002842171718 |  |  |
|  |  |  | 5.000002842171718 | -0.000002842171718 |  |  |
|  |  |  | 6.000002842171718 | -0.000002842171718 |  |  |
|  |  |  | 7.000002842171719 | -0.000002842171719 |  |  |
|  |  |  | 8.000002842171716 | -0.000002842171716 |  |  |
|  |  |  | 9.000002842171719 | -0.000002842171719 |  |  |
|  |  |  | 10.000002842171718 | -0.000002842171718 |  |  |
| 9 |  |  | 1.000058620395631 | -0.000058620395631 |  |  |
|  |  |  | 2.000058620395634 | -0.000058620395634 |  |  |
|  |  |  | 3.000058620395635 | -0.000058620395635 |  |  |
|  |  |  | 4.000058620395635 | -0.000058620395635 |  |  |
|  |  |  | 5.000058620395635 | -0.000058620395635 |  |  |
|  |  |  | 6.000058620395635 | -0.000058620395635 |  |  |
|  |  |  | 7.000058620395635 | -0.000058620395635 |  |  |
|  |  |  | 8.000058620395633 | -0.000058620395633 |  |  |
|  |  |  | 9.000058620395636 | -0.000058620395636 |  |  |
|  |  |  | 10.000058620395633 | -0.000058620395633 |  |  |
| 10 |  |  | 0.999449358769408 | 0.000550641230592 |  |  |
|  |  |  | 1.999449358769404 | 0.000550641230596 |  |  |
|  |  |  | 2.999449358769405 | 0.000550641230595 |  |  |
|  |  |  | 3.999449358769404 | 0.000550641230596 |  |  |
|  |  |  | 4.999449358769406 | 0.000550641230594 |  |  |
|  |  |  | 5.999449358769406 | 0.000550641230594 |  |  |
|  |  |  | 6.999449358769406 | 0.000550641230594 |  |  |
|  |  |  | 7.999449358769405 | 0.000550641230595 |  |  |
|  |  |  | 8.999449358769406 | 0.000550641230594 |  |  |
|  |  |  | 9.999449358769404 | 0.000550641230596 |  |  |
| 11 |  |  | 1.014595941616223 | -0.014595941616223 |  |  |
|  |  |  | 2.014595941616232 | -0.014595941616232 |  |  |
|  |  |  | 3.014595941616230 | -0.014595941616230 |  |  |
|  |  |  | 4.014595941616233 | -0.014595941616233 |  |  |
|  |  |  | 5.014595941616231 | -0.014595941616231 |  |  |
|  |  |  | 6.014595941616233 | -0.014595941616233 |  |  |
|  |  |  | 7.014595941616231 | -0.014595941616231 |  |  |
|  |  |  | 8.014595941616232 | -0.014595941616232 |  |  |
|  |  |  | 9.014595941616232 | -0.014595941616232 |  |  |
|  |  |  | 10.014595941616234 | -0.014595941616234 |  |  |
| 12 |  |  | 1.043010752688166 | -0.043010752688166 |  |  |
|  |  |  | 2.043010752688168 | -0.043010752688168 |  |  |
|  |  |  | 3.043010752688169 | -0.043010752688169 |  |  |
|  |  |  | 4.043010752688169 | -0.043010752688169 |  |  |
|  |  |  | 5.043010752688168 | -0.043010752688168 |  |  |
|  |  |  | 6.043010752688170 | -0.043010752688170 |  |  |
|  |  |  | 7.043010752688169 | -0.043010752688169 |  |  |
|  |  |  | 8.043010752688170 | -0.043010752688170 |  |  |
|  |  |  | 9.043010752688172 | -0.043010752688172 |  |  |
|  |  |  | 10.043010752688170 | -0.043010752688170 |  |  |
| 13 |  |  | 1.634615384615379 | -0.634615384615379 |  |  |
|  |  |  | 2.634615384615383 | -0.634615384615383 |  |  |
|  |  |  | 3.634615384615382 | -0.634615384615382 |  |  |
|  |  |  | 4.634615384615383 | -0.634615384615383 |  |  |
|  |  |  | 5.634615384615383 | -0.634615384615383 |  |  |
|  |  |  | 6.634615384615383 | -0.634615384615383 |  |  |
|  |  |  | 7.634615384615383 | -0.634615384615383 |  |  |
|  |  |  | 8.634615384615383 | -0.634615384615383 |  |  |
|  |  |  | 9.634615384615381 | -0.634615384615381 |  |  |
|  |  |  | 10.634615384615385 | -0.634615384615385 |  |  |
| 14 |  |  | -4.833333333333329 | 5.833333333333329 |  |  |
|  |  |  | -3.833333333333333 | 5.833333333333332 |  |  |
|  |  |  | -2.833333333333332 | 5.833333333333332 |  |  |
|  |  |  | -1.833333333333332 | 5.833333333333332 |  |  |
|  |  |  | -0.833333333333332 | 5.833333333333332 |  |  |
|  |  |  | 0.166666666666668 | 5.833333333333332 |  |  |
|  |  |  | 1.166666666666667 | 5.833333333333332 |  |  |
|  |  |  | 2.166666666666668 | 5.833333333333332 |  |  |
|  |  |  | 3.166666666666667 | 5.833333333333332 |  |  |
|  |  |  | 4.166666666666667 | 5.833333333333333 |  |  |

Как видно из таблиц, с увеличением величины *k* увеличивается погрешность вычислений. Это связано с тем, что все элементы матриц зависят от первого элемента диагонали.

При k=5 (для float) и k=14 (для double) СЛАУ не решается. Это связано со свойствами арифметики и размером мантиссы. По итогу получаем, что A + 10-k = A.

Вычисления со смешанной точностью не показывали значительных результатов из-за того, что все величины по-прежнему хранятся с одинарной точностью, а избавляемся лишь от погрешности при суммировании.

Двойная точность обеспечивает гораздо меньшую погрешность, по сравнению с одинарной и смешанной точностью.

1. Исследование матриц Гильберта

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| k | xk (один. точность) | x\*-xk (один. точность) | xk (двой. точность) | x\*-xk (двой. точность) | xk (скалярное пр.) | x\*-xk (скалярное пр.) |
| 1 | 1.000000 | 0.000000 | 1.000000000000000 | 0.000000000000000 | 1.000000 | 0.000000 |
| 2 | 0.999994 | 0.000006 | 0.999999999999994 | 0.000000000000006 | 0.999994 | 0.000006 |
|  | 2.000012 | -0.000012 | 2.000000000000012 | -0.000000000000012 | 2.000012 | -0.000012 |
| 3 | 0.999976 | 0.000024 | 0.999999999999974 | 0.000000000000026 | 0.999975 | 0.000025 |
|  | 2.000137 | -0.000137 | 2.000000000000151 | -0.000000000000151 | 2.000142 | -0.000142 |
|  | 2.999869 | 0.000131 | 2.99999999999985 | 0.000000000000145 | 2.999863 | 0.000137 |
| 4 | 0.999649 | 0.000351 | 0.999999999999816 | 0.000000000000184 | 0.999649 | 0.000351 |
|  | 2.003859 | -0.003859 | 2.000000000001945 | -0.000000000001945 | 2.003859 | -0.003859 |
|  | 2.990870 | 0.009130 | 2.999999999995540 | 0.000000000004460 | 2.990870 | 0.009130 |
|  | 4.005863 | -0.005863 | 4.000000000002797 | -0.000000000002797 | 4.005863 | -0.005863 |
| 5 | 1.000089 | -0.000089 | 0.999999999999746 | 0.000000000000254 | 1.000468 | -0.000468 |
|  | 1.999515 | 0.000485 | 2.000000000005295 | -0.000000000005295 | 1.992298 | 0.007702 |
|  | 2.999536 | 0.000464 | 2.999999999976150 | 0.000000000023850 | 3.030993 | -0.030993 |
|  | 4.003055 | -0.003055 | 4.000000000036916 | -0.000000000036916 | 3.955230 | 0.044770 |
|  | 4.997726 | 0.002274 | 4.999999999981640 | 0.000000000018360 | 5.021229 | -0.021229 |
| 6 |  |  | 1.000000000004901 | -0.000000000004901 |  |  |
|  |  |  | 1.999999999861638 | 0.000000000138362 |  |  |
|  |  |  | 3.000000000935562 | -0.000000000935562 |  |  |
|  |  |  | 3.999999997561440 | 0.000000002438560 |  |  |
|  |  |  | 5.000000002698965 | -0.000000002698965 |  |  |
|  |  |  | 5.999999998933633 | 0.000000001066367 |  |  |
| 7 |  |  | 0.999999999929649 | 0.000000000070351 |  |  |
|  |  |  | 2.000000002869204 | -0.000000002869204 |  |  |
|  |  |  | 2.999999971981904 | 0.000000028018096 |  |  |
|  |  |  | 4.000000109931106 | -0.000000109931106 |  |  |
|  |  |  | 4.999999797126004 | 0.000000202873996 |  |  |
|  |  |  | 6.000000176176755 | -0.000000176176755 |  |  |
|  |  |  | 6.999999941930391 | 0.000000058069609 |  |  |
| 8 |  |  | 1.000000001056607 | -0.000000001056607 |  |  |
|  |  |  | 1.999999942660109 | 0.000000057339891 |  |  |
|  |  |  | 3.000000755944484 | -0.000000755944484 |  |  |
|  |  |  | 3.999995877470097 | 0.000004122529903 |  |  |
|  |  |  | 5.000011169229773 | -0.000011169229773 |  |  |
|  |  |  | 5.999984112107199 | 0.000015887892801 |  |  |
|  |  |  | 7.000011357596156 | -0.000011357596156 |  |  |
|  |  |  | 7.999996783094077 | 0.000003216905923 |  |  |
| 9 |  |  | 0.999999993387710 | 0.000000006612290 |  |  |
|  |  |  | 2.000000449839666 | -0.000000449839666 |  |  |
|  |  |  | 2.999992464385602 | 0.000007535614398 |  |  |
|  |  |  | 4.000053370775004 | -0.000053370775004 |  |  |
|  |  |  | 4.999805468246255 | 0.000194531753745 |  |  |
|  |  |  | 6.000395161117470 | -0.000395161117470 |  |  |
|  |  |  | 6.999548119891914 | 0.000451880108086 |  |  |
|  |  |  | 8.000271951057250 | -0.000271951057250 |  |  |
|  |  |  | 8.999933017083066 | 0.000066982916934 |  |  |
| 10 |  |  | 1.000000009370616 | -0.000000009370616 |  |  |
|  |  |  | 1.999999282531497 | 0.000000717468503 |  |  |
|  |  |  | 3.000013769412790 | -0.000013769412790 |  |  |
|  |  |  | 3.999885791605348 | 0.000114208394652 |  |  |
|  |  |  | 5.000501913252191 | -0.000501913252191 |  |  |
|  |  |  | 5.998718612513328 | 0.001281387486672 |  |  |
|  |  |  | 7.001965369958641 | -0.001965369958641 |  |  |
|  |  |  | 7.998214569560300 | 0.001785430439700 |  |  |
|  |  |  | 9.000885341364761 | -0.000885341364761 |  |  |
|  |  |  | 9.999815339939715 | 0.000184660060285 |  |  |
| 11 |  |  | 1.000000213273592 | -0.000000213273592 |  |  |
|  |  |  | 1.999977019426607 | 0.000022980573393 |  |  |
|  |  |  | 3.000609230026621 | -0.000609230026621 |  |  |
|  |  |  | 3.993072170766137 | 0.006927829233863 |  |  |
|  |  |  | 5.041838424927234 | -0.041838424927234 |  |  |
|  |  |  | 5.851244419957516 | 0.148755580042484 |  |  |
|  |  |  | 7.326941542311549 | -0.326941542311549 |  |  |
|  |  |  | 7.550723488155828 | 0.449276511844172 |  |  |
|  |  |  | 9.375753320664467 | -0.375753320664467 |  |  |
|  |  |  | 9.825112932827761 | 0.174887067172239 |  |  |
|  |  |  | 11.034727400987604 | -0.034727400987604 |  |  |
| 12 |  |  | 0.999999946440155 | 0.000000053559845 |  |  |
|  |  |  | 1.999999398370371 | 0.000000601629629 |  |  |
|  |  |  | 3.000209110606525 | -0.000209110606525 |  |  |
|  |  |  | 3.994992335769425 | 0.005007664230575 |  |  |
|  |  |  | 5.050088680200561 | -0.050088680200561 |  |  |
|  |  |  | 5.729594762516243 | 0.270405237483757 |  |  |
|  |  |  | 7.874784778616590 | -0.874784778616590 |  |  |
|  |  |  | 6.224580200142591 | 1.775419799857409 |  |  |
|  |  |  | 11.279706664129769 | -2.279706664129769 |  |  |
|  |  |  | 8.201531881359008 | 1.798468118640992 |  |  |
|  |  |  | 11.795608705922694 | -0.795608705922694 |  |  |
|  |  |  | 11.848903182742196 | 0.151096817257804 |  |  |

Результаты аналогичны, погрешность вычислений со смешанной точностью также незначительно меньше погрешности с одинарной точностью, погрешность вычислений с двойной точностью гораздо ниже погрешности вычислений с одинарной точностью.

Решение по причинам округления (отбрасывания знаков при приведении double к float в смешанной тоности). Как раз по причине наличия данных ошибок всегда более точно будет решать только программа, в которой всюду используются типы данных, в которых числа можно представить с большей точностью.

Для примера возьмём матрицы размерностью 1 и 3. При к=1 матрица выглядит так (1), умножаем на вектор x1 = 1 и получаем вектор F = (1).

https://sun9-2.userapi.com/impf/kG25GudzpvmW-hquOt6QtJpV4MFs-Ff12ncg6g/c0K2p62Xpc0.jpg?size=0x0&quality=90&proxy=1&sign=db790ffaf6e122a7df2a014b59cb0b48При к = 3 матрица выглядит так , умножим вторую строку на вектор х и получим F2.

Однако из-за ограниченной мантиссы получаем 1.916667

1. Метод гаусса с ведущим элементом

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | xk llt(двой. точность) | x\*-xk llt(двой. точность) | xk гаусс(двой. точность) | x\*-xk гаусс(двой. точность) |
| 1 | 0.999999999998987 | 0.000000000001013 | 1.000000000000248 | -0.000000000000248 |
|  | 1.999999999998980 | 0.000000000001020 | 2.000000000000250 | -0.000000000000250 |
|  | 2.999999999998981 | 0.000000000001019 | 3.000000000000250 | -0.000000000000250 |
|  | 3.999999999998980 | 0.000000000001020 | 4.000000000000250 | -0.000000000000250 |
|  | 4.999999999998982 | 0.000000000001018 | 5.000000000000250 | -0.000000000000250 |
|  | 5.999999999998980 | 0.000000000001020 | 6.000000000000253 | -0.000000000000253 |
|  | 6.999999999998980 | 0.000000000001020 | 7.000000000000252 | -0.000000000000252 |
|  | 7.999999999998982 | 0.000000000001018 | 8.000000000000250 | -0.000000000000250 |
|  | 8.999999999998982 | 0.000000000001018 | 9.000000000000252 | -0.000000000000252 |
|  | 9.999999999998979 | 0.000000000001021 | 10.000000000000250 | -0.000000000000250 |
| 2 | 0.999999999996091 | 0.000000000003909 | 1.000000000003376 | -0.000000000003376 |
|  | 1.999999999996089 | 0.000000000003911 | 2.000000000003376 | -0.000000000003376 |
|  | 2.999999999996088 | 0.000000000003912 | 3.000000000003378 | -0.000000000003378 |
|  | 3.999999999996088 | 0.000000000003912 | 4.000000000003377 | -0.000000000003377 |
|  | 4.999999999996088 | 0.000000000003912 | 5.000000000003377 | -0.000000000003377 |
|  | 5.999999999996088 | 0.000000000003912 | 6.000000000003380 | -0.000000000003380 |
|  | 6.999999999996088 | 0.000000000003912 | 7.000000000003379 | -0.000000000003379 |
|  | 7.999999999996089 | 0.000000000003911 | 8.000000000003377 | -0.000000000003377 |
|  | 8.999999999996088 | 0.000000000003912 | 9.000000000003380 | -0.000000000003380 |
|  | 9.999999999996090 | 0.000000000003910 | 10.000000000003379 | -0.000000000003379 |
| 3 | 1.000000000024869 | -0.000000000024869 | 1.000000000037306 | -0.000000000037306 |
|  | 2.000000000024869 | -0.000000000024869 | 2.000000000037307 | -0.000000000037307 |
|  | 3.000000000024870 | -0.000000000024870 | 3.000000000037307 | -0.000000000037307 |
|  | 4.000000000024869 | -0.000000000024869 | 4.000000000037307 | -0.000000000037307 |
|  | 5.000000000024870 | -0.000000000024870 | 5.000000000037307 | -0.000000000037307 |
|  | 6.000000000024869 | -0.000000000024869 | 6.000000000037310 | -0.000000000037310 |
|  | 7.000000000024870 | -0.000000000024870 | 7.000000000037309 | -0.000000000037309 |
|  | 8.000000000024871 | -0.000000000024871 | 8.000000000037307 | -0.000000000037307 |
|  | 9.000000000024869 | -0.000000000024869 | 9.000000000037311 | -0.000000000037311 |
|  | 10.000000000024871 | -0.000000000024871 | 10.000000000037307 | -0.000000000037307 |
| 4 | 1.000000000852656 | -0.000000000852656 | 1.000000000248691 | -0.000000000248691 |
|  | 2.000000000852661 | -0.000000000852661 | 2.000000000248692 | -0.000000000248692 |
|  | 3.000000000852661 | -0.000000000852661 | 3.000000000248693 | -0.000000000248693 |
|  | 4.000000000852661 | -0.000000000852661 | 4.000000000248692 | -0.000000000248692 |
|  | 5.000000000852659 | -0.000000000852659 | 5.000000000248692 | -0.000000000248692 |
|  | 6.000000000852661 | -0.000000000852661 | 6.000000000248694 | -0.000000000248694 |
|  | 7.000000000852662 | -0.000000000852662 | 7.000000000248694 | -0.000000000248694 |
|  | 8.000000000852660 | -0.000000000852660 | 8.000000000248692 | -0.000000000248692 |
|  | 9.000000000852662 | -0.000000000852662 | 9.000000000248695 | -0.000000000248695 |
|  | 10.000000000852662 | -0.000000000852662 | 10.000000000248694 | -0.000000000248694 |
| 5 | 0.999999994670929 | 0.000000005329071 | 1.000000003730350 | -0.000000003730350 |
|  | 1.999999994670926 | 0.000000005329074 | 2.000000003730352 | -0.000000003730352 |
|  | 2.999999994670926 | 0.000000005329074 | 3.000000003730353 | -0.000000003730353 |
|  | 3.999999994670925 | 0.000000005329075 | 4.000000003730353 | -0.000000003730353 |
|  | 4.999999994670925 | 0.000000005329075 | 5.000000003730352 | -0.000000003730352 |
|  | 5.999999994670925 | 0.000000005329075 | 6.000000003730356 | -0.000000003730356 |
|  | 6.999999994670925 | 0.000000005329075 | 7.000000003730354 | -0.000000003730354 |
|  | 7.999999994670926 | 0.000000005329074 | 8.000000003730351 | -0.000000003730351 |
|  | 8.999999994670926 | 0.000000005329074 | 9.000000003730356 | -0.000000003730356 |
|  | 9.999999994670924 | 0.000000005329076 | 10.000000003730353 | -0.000000003730353 |
| 6 | 0.999999962696508 | 0.000000037303492 | 1.000000014210854 | -0.000000014210854 |
|  | 1.999999962696505 | 0.000000037303495 | 2.000000014210853 | -0.000000014210853 |
|  | 2.999999962696505 | 0.000000037303495 | 3.000000014210854 | -0.000000014210854 |
|  | 3.999999962696506 | 0.000000037303494 | 4.000000014210855 | -0.000000014210855 |
|  | 4.999999962696506 | 0.000000037303494 | 5.000000014210854 | -0.000000014210854 |
|  | 5.999999962696505 | 0.000000037303495 | 6.000000014210857 | -0.000000014210857 |
|  | 6.999999962696505 | 0.000000037303495 | 7.000000014210856 | -0.000000014210856 |
|  | 7.999999962696505 | 0.000000037303495 | 8.000000014210855 | -0.000000014210855 |
|  | 8.999999962696505 | 0.000000037303495 | 9.000000014210858 | -0.000000014210858 |
|  | 9.999999962696503 | 0.000000037303497 | 10.000000014210855 | -0.000000014210855 |
| 7 | 0.999999964472861 | 0.000000035527139 | 1.000000088817835 | -0.000000088817835 |
|  | 1.999999964472861 | 0.000000035527139 | 2.000000088817835 | -0.000000088817835 |
|  | 2.999999964472861 | 0.000000035527139 | 3.000000088817837 | -0.000000088817837 |
|  | 3.999999964472860 | 0.000000035527140 | 4.000000088817837 | -0.000000088817837 |
|  | 4.999999964472861 | 0.000000035527139 | 5.000000088817836 | -0.000000088817836 |
|  | 5.999999964472861 | 0.000000035527139 | 6.000000088817838 | -0.000000088817838 |
|  | 6.999999964472862 | 0.000000035527138 | 7.000000088817838 | -0.000000088817838 |
|  | 7.999999964472862 | 0.000000035527138 | 8.000000088817837 | -0.000000088817837 |
|  | 8.999999964472861 | 0.000000035527139 | 9.000000088817838 | -0.000000088817838 |
|  | 9.999999964472861 | 0.000000035527139 | 10.000000088817837 | -0.000000088817837 |
| 8 | 1.000002842171716 | -0.000002842171716 | 1.000003730348718 | -0.000003730348718 |
|  | 2.000002842171718 | -0.000002842171718 | 2.000003730348719 | -0.000003730348719 |
|  | 3.000002842171718 | -0.000002842171718 | 3.000003730348720 | -0.000003730348720 |
|  | 4.000002842171718 | -0.000002842171718 | 4.000003730348720 | -0.000003730348720 |
|  | 5.000002842171718 | -0.000002842171718 | 5.000003730348719 | -0.000003730348719 |
|  | 6.000002842171718 | -0.000002842171718 | 6.000003730348722 | -0.000003730348722 |
|  | 7.000002842171719 | -0.000002842171719 | 7.000003730348721 | -0.000003730348721 |
|  | 8.000002842171716 | -0.000002842171716 | 8.000003730348720 | -0.000003730348720 |
|  | 9.000002842171719 | -0.000002842171719 | 9.000003730348723 | -0.000003730348723 |
|  | 10.000002842171718 | -0.000002842171718 | 10.000003730348720 | -0.000003730348720 |
| 9 | 1.000058620395631 | -0.000058620395631 | 1.000023092575467 | -0.000023092575467 |
|  | 2.000058620395634 | -0.000058620395634 | 2.000023092575467 | -0.000023092575467 |
|  | 3.000058620395635 | -0.000058620395635 | 3.000023092575468 | -0.000023092575468 |
|  | 4.000058620395635 | -0.000058620395635 | 4.000023092575468 | -0.000023092575468 |
|  | 5.000058620395635 | -0.000058620395635 | 5.000023092575467 | -0.000023092575467 |
|  | 6.000058620395635 | -0.000058620395635 | 6.000023092575471 | -0.000023092575471 |
|  | 7.000058620395635 | -0.000058620395635 | 7.000023092575470 | -0.000023092575470 |
|  | 8.000058620395633 | -0.000058620395633 | 8.000023092575468 | -0.000023092575468 |
|  | 9.000058620395636 | -0.000058620395636 | 9.000023092575471 | -0.000023092575471 |
|  | 10.000058620395633 | -0.000058620395633 | 10.000023092575468 | -0.000023092575468 |
| 10 | 0.999449358769408 | 0.000550641230592 | 1.000301977955606 | -0.000301977955606 |
|  | 1.999449358769404 | 0.000550641230596 | 2.000301977955606 | -0.000301977955606 |
|  | 2.999449358769405 | 0.000550641230595 | 3.000301977955606 | -0.000301977955606 |
|  | 3.999449358769404 | 0.000550641230596 | 4.000301977955607 | -0.000301977955607 |
|  | 4.999449358769406 | 0.000550641230594 | 5.000301977955606 | -0.000301977955606 |
|  | 5.999449358769406 | 0.000550641230594 | 6.000301977955609 | -0.000301977955609 |
|  | 6.999449358769406 | 0.000550641230594 | 7.000301977955608 | -0.000301977955608 |
|  | 7.999449358769405 | 0.000550641230595 | 8.000301977955607 | -0.000301977955607 |
|  | 8.999449358769406 | 0.000550641230594 | 9.000301977955608 | -0.000301977955608 |
|  | 9.999449358769404 | 0.000550641230596 | 10.000301977955607 | -0.000301977955607 |
| 11 | 1.014595941616223 | -0.014595941616223 | 1.019520851818985 | -0.019520851818985 |
|  | 2.014595941616232 | -0.014595941616232 | 2.019520851818985 | -0.019520851818985 |
|  | 3.014595941616230 | -0.014595941616230 | 3.019520851818986 | -0.019520851818986 |
|  | 4.014595941616233 | -0.014595941616233 | 4.019520851818985 | -0.019520851818985 |
|  | 5.014595941616231 | -0.014595941616231 | 5.019520851818985 | -0.019520851818985 |
|  | 6.014595941616233 | -0.014595941616233 | 6.019520851818989 | -0.019520851818989 |
|  | 7.014595941616231 | -0.014595941616231 | 7.019520851818988 | -0.019520851818988 |
|  | 8.014595941616232 | -0.014595941616232 | 8.019520851818985 | -0.019520851818985 |
|  | 9.014595941616232 | -0.014595941616232 | 9.019520851818989 | -0.019520851818989 |
|  | 10.014595941616234 | -0.014595941616234 | 10.019520851818985 | -0.019520851818985 |
| 12 | 1.043010752688166 | -0.043010752688166 | 1.019520851818985 | -0.019520851818985 |
|  | 2.043010752688168 | -0.043010752688168 | 2.019520851818985 | -0.019520851818985 |
|  | 3.043010752688169 | -0.043010752688169 | 3.019520851818986 | -0.019520851818986 |
|  | 4.043010752688169 | -0.043010752688169 | 4.019520851818985 | -0.019520851818985 |
|  | 5.043010752688168 | -0.043010752688168 | 5.019520851818985 | -0.019520851818985 |
|  | 6.043010752688170 | -0.043010752688170 | 6.019520851818989 | -0.019520851818989 |
|  | 7.043010752688169 | -0.043010752688169 | 7.019520851818988 | -0.019520851818988 |
|  | 8.043010752688170 | -0.043010752688170 | 8.019520851818985 | -0.019520851818985 |
|  | 9.043010752688172 | -0.043010752688172 | 9.019520851818989 | -0.019520851818989 |
|  | 10.043010752688170 | -0.043010752688170 | 10.019520851818985 | -0.019520851818985 |
| 13 | 1.634615384615379 | -0.634615384615379 | 1.336283185840704 | -0.336283185840704 |
|  | 2.634615384615383 | -0.634615384615383 | 2.336283185840706 | -0.336283185840706 |
|  | 3.634615384615382 | -0.634615384615382 | 3.336283185840705 | -0.336283185840705 |
|  | 4.634615384615383 | -0.634615384615383 | 4.336283185840706 | -0.336283185840706 |
|  | 5.634615384615383 | -0.634615384615383 | 5.336283185840705 | -0.336283185840705 |
|  | 6.634615384615383 | -0.634615384615383 | 6.336283185840708 | -0.336283185840708 |
|  | 7.634615384615383 | -0.634615384615383 | 7.336283185840708 | -0.336283185840708 |
|  | 8.634615384615383 | -0.634615384615383 | 8.336283185840706 | -0.336283185840706 |
|  | 9.634615384615381 | -0.634615384615381 | 9.336283185840710 | -0.336283185840710 |
|  | 10.634615384615385 | -0.634615384615385 | 10.336283185840706 | -0.336283185840706 |
| 14 | -4.833333333333329 | 5.833333333333329 | 1.749999999999998 | -0.749999999999998 |
|  | -3.833333333333333 | 5.833333333333332 | 2.749999999999996 | -0.749999999999996 |
|  | -2.833333333333332 | 5.833333333333332 | 3.749999999999997 | -0.749999999999997 |
|  | -1.833333333333332 | 5.833333333333332 | 4.749999999999998 | -0.749999999999998 |
|  | -0.833333333333332 | 5.833333333333332 | 5.749999999999997 | -0.749999999999997 |
|  | 0.166666666666668 | 5.833333333333332 | 6.750000000000000 | -0.750000000000000 |
|  | 1.166666666666667 | 5.833333333333332 | 7.749999999999999 | -0.749999999999999 |
|  | 2.166666666666668 | 5.833333333333332 | 8.749999999999998 | -0.749999999999998 |
|  | 3.166666666666667 | 5.833333333333332 | 9.750000000000000 | -0.750000000000000 |
|  | 4.166666666666667 | 5.833333333333333 | 10.749999999999998 | -0.749999999999998 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| метод | L | Lt | диагональ | прямой ход | обратный ход | всего |
| LLt |  | - | - | n2+n | n2 |  |
| Метод Гаусса | - | - | - |  | n2 |  |
| LDLt |  | - |  | n2+n | n2 |  |
| LDLt( |  | - |  | n2+n | n2 |  |

Количество действий при использовании LLt разложения меньше чем в методе Гаусса, однако, метод Гаусса дает меньшую погрешность.

1. Разложение LDLt

,

, 1 ≤ ≤ n;

,

; , 1 ≤ ≤ n;

1. Текст программы

Main

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

#include "subroutine.h"

#include "gauss.h"

#include "hylbert.h"

#include "Ak.h"

using namespace std;

typedef double real;

//typedef float real;

int main()

{

real \*\*matrix\_L, \*vector\_di, \* vector\_b, \* vector\_a;

int n, m, f;

printf("<1> LLt\n");

printf("<2> Gauss\n");

printf("<3> HYlbertM\_generation\n");

printf("<4> AkM\_generation\n");

scanf("%d", &f);

switch(f){

case(1):

FILE\* IN;

IN = fopen("h.txt", "r");

read(IN, n, m, matrix\_L, vector\_di, vector\_b);

decomposition(n, m, matrix\_L, vector\_di, vector\_b);

vector\_y(n, m, matrix\_L, vector\_di, vector\_b);

vector\_x(n, m, matrix\_L, vector\_di, vector\_b);

break;

case(2):

FILE\* INg;

INg = fopen("ingauss.txt", "r");

gauss(INg, n, matrix\_L, vector\_di, vector\_b);

break;

case(3):

FILE\* INh;

INh = fopen("h.txt", "w");

Hylb(INh, n, m, matrix\_L, vector\_di, vector\_b, vector\_a);

break;

case(4):

FILE\* INk;

INk = fopen("ak.txt", "w");

Ak(INk, n, m, matrix\_L, vector\_di, vector\_b, vector\_a);

break;

}

}

Subroutine.h

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

using namespace std;

typedef double real;

#define inform "%lf"

#define form "%-20.15f"

//typedef float real;

//

//#define inform "%f"

//#define form "%-10f"

void read(FILE \*IN, int &n, int& m, real\*\* &matrix\_L, real\*& vector\_di, real\* &vector\_b) {

fscanf(IN, "%d" "%d", &n, &m);

vector\_b = new real[n];

vector\_di = new real[n];

matrix\_L = new real\*[n];

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

matrix\_L[i] = new real [m];

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

fscanf(IN, inform, &vector\_b[i]);

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

fscanf(IN, inform, &vector\_di[i]);

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

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

fscanf(IN, inform, &matrix\_L[i][j]);

}

void decomposition(int& n, int& m, real\*\*& matrix\_L, real\*& vector\_di, real\*& vector\_b) {

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

int f = 0; //отвечает за смещение от первого индекса если ширина ленты < n-1

int t = 0;

int j =m- i;

if (j < 0) j = 0;

if (m - i < 0) f = i - m; //смещение от нач индекса строки

while ( j <= m-1) {

real sum2 = 0;

int tt = m;

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

sum2 += matrix\_L[i][k] \* matrix\_L[t+f][--tt];

matrix\_L[i][j] = (matrix\_L[i][j] - sum2) / vector\_di[f+t];

j++;

t++;

}

real sum1 = 0;

for (int k = 1; (k <= i) && (m-k>=0); k++) {

sum1 += matrix\_L[i][m-k] \* matrix\_L[i][m-k];

}

vector\_di[i] = sqrt(vector\_di[i] - sum1);

}

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

// printf(form, vector\_di[i], "\n");

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

// printf("\n");

// for (int j = 0; j < m; j++)

// printf(form, matrix\_L[i][j], "\n");

//}

//printf("\n");

//printf("\n");

}

void vector\_y(int& n, int& m, real\*\*& matrix\_L, real\*& vector\_di, real\*& vector\_b) {

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

int g = i;

for (int k = m - 1; (k > m - 1 - i) && (k >= 0); k--) {

//printf(form, vector\_b[i], " ");

vector\_b[i] -= matrix\_L[i][k] \* vector\_b[--g]; //(double) для скалярного произведения в дв точности

//printf(form, vector\_b[i] , " ");

//printf("\n");

}

vector\_b[i] /= vector\_di[i];

printf("\n");

}

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

printf(form, vector\_b[i]);

printf("\n");

}

void vector\_x(int& n, int& m, real\*\*& matrix\_L, real\*& vector\_di, real\*& vector\_b) {

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

int j = m;

vector\_b[i] /= vector\_di[i];

for (int k = i - 1; (k >= 0) && (j > 0); k--)

vector\_b[k] -= matrix\_L[i][--j] \* vector\_b[i];

}

printf("\n");

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

printf(form, vector\_b[i]);

printf("\n");

printf("\n");

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

printf(form, real(i)+1-vector\_b[i]);

FILE\* OUT = fopen("out.txt", "w"); //вывод в файл для удобства заполнения таблицы

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

fprintf(OUT,form"\n", vector\_b[i] );

fprintf(OUT,"\n");

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

fprintf(OUT, form"\n", real(i) + 1 - vector\_b[i]);

}

Gauss.h

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

using namespace std;

typedef double real;

#define inform "%lf"

#define form "%-20.15f"

//typedef float real;

//

//#define inform "%f"

//#define form "%-10f"

void gauss(FILE \*IN, int& n, real\*\*& A, real\*& vector\_a, real\*& vector\_b) {

fscanf(IN, "%d" , &n);

vector\_b = new real[n];

vector\_a = new real[n];

A = new real \*[n];

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

A[i] = new real[n+1];

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

for (int j = 0; j < n+1; j++)

fscanf(IN, inform, &A[i][j]);

/\*for (int i = 0; i < n; i++)

{

printf("\n");

for (int j = 0; j < n + 1; j++)

printf(form, A[i][j]);

}\*/

printf("\n");

real max, sum, t;

int imax, jmax;

int \*index = new int[n];

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

index[i] = i;

for (int k = 0; k < n-1; k++) {

max = 0;

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

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

if (max < abs(A[i][j])) {

max = abs(A[i][j]);

imax = i;

jmax = j;

}

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

swap(A[i][k], A[i][jmax]);

swap(index[k], index[jmax]);

for (int j = 0; j < n + 1; j++)

swap(A[k][j], A[imax][j]);

t = A[k][k];

if (t != 0) {

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

A[k][j] /= t;

}

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

t = A[i][k];

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

A[i][j] -= t \* A[k][j];

}

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

//{

// printf("\n");

// for (int j = 0; j < n + 1; j++)

// printf(form, A[i][j]);

//}

//printf("\n");

}

A[n-1][n] /= A[n - 1][n - 1];

A[n - 1][n - 1] = 1;

vector\_a[n - 1] = A[n - 1][n];

for (int i = n - 2; i >= 0; i--) {

sum = 0;

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

sum += A[i][j] \* vector\_a[j];

vector\_a[i] = A[i][n] - sum;

}

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

vector\_b[index[i]] = vector\_a[i];

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

printf(form, vector\_b[i]);

printf("\n");

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

printf(form, real(i)+1-vector\_b[i]);

FILE\* OUT = fopen("outg.txt", "w"); //вывод в файл для удобства заполнения таблицы

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

fprintf(OUT, form"\n", vector\_b[i]);

fprintf(OUT, "\n");

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

fprintf(OUT, form"\n", real(i) + 1 - vector\_b[i]);

}

Hylbert.h

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

using namespace std;

typedef double real;

#define inform "%lf"

#define form "%-20.15f"

//typedef float real;

//

//#define inform "%f"

//#define form "%-10f"

void Hylb(FILE\* IN, int& n, int& m, real\*\*& matrix\_L, real\*& vector\_di, real\*& vector\_b, real\*& vector\_x) {

printf("n?");

scanf("%d", &n);

m = n - 1;

vector\_b = new real[n];

vector\_x = new real[n];

vector\_di = new real[n];

matrix\_L = new real \* [n];

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

matrix\_L[i] = new real[m];

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

vector\_x[i - 1] = i;

vector\_di[i - 1] = 1 / (2\*real(i)-1);

int k = 1;

for (int j = 0; j < m; j++) {

if (j >= m - i + 1) {

matrix\_L[i - 1][j] = 1 / (real(i)+real(k)-1) ;

k++;

}

else

matrix\_L[i - 1][j] = 0;

}

}

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

real sum = 0;

int g = m;

sum = vector\_di[i] \* vector\_x[i];

for (int j = i + 1; j < n && m-g>=0; j++) {

sum += matrix\_L[j][--g] \* vector\_x[j];

}

g = m;

for (int k = i-1; k >= 0; k--) {

sum += matrix\_L[i][--g] \* vector\_x[k];

}

vector\_b[i] = sum;

}

fprintf(IN, "%d\n" "%d\n", n, m);

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

fprintf(IN, form, vector\_b[i]);

fprintf(IN,"\n");

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

fprintf(IN, form, vector\_di[i]);

fprintf(IN, "\n");

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

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

fprintf(IN, form, matrix\_L[i][j]);

fprintf(IN, "\n");

}

}

Ak.h

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

using namespace std;

typedef double real;

#define inform "%lf"

#define form "%-20.15f"

//typedef float real;

//

//#define inform "%f"

//#define form "%-10f"

void Ak(FILE\* IN, int& n, int& m, real\*\*& matrix\_L, real\*& vector\_di, real\*& vector\_b, real\*& vector\_x) {

printf("n?");

scanf("%d", &n);

m = n - 1;

int k;

printf("k?");

scanf("%d", &k);

vector\_b = new real[n];

vector\_x = new real[n];

vector\_di = new real[n];

matrix\_L = new real \* [n];

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

matrix\_L[i] = new real[m];

real \*\*matrix = new real \* [n];

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

matrix[i] = new real[n+1];

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

vector\_di[i] = 0;

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

vector\_x[i] = real(i)+1;

for (int j = 0; j < m; j++) {

if (j >= m - i)

matrix\_L[i][j] = -rand()%5;

else

matrix\_L[i][j] = 0;

}

}

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

int g = m;

for (int j = i + 1; j < n && m - g >= 0; j++)

vector\_di[i] -= matrix\_L[j][--g];

g = m;

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

vector\_di[i] -= matrix\_L[i][--g];

}

vector\_di[0] += pow(10,-k);

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

real sum = 0;

int g = m;

sum = vector\_di[i] \* vector\_x[i];

for (int j = i + 1; j < n && m - g >= 0; j++)

sum += matrix\_L[j][--g] \* vector\_x[j];

g = m;

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

sum += matrix\_L[i][--g] \* vector\_x[k];

vector\_b[i] = sum;

}

for (int i = 0; i < n; i++) { //построение плотной матрицы для метода гаусса

matrix[i][i] = vector\_di[i];

int t = m;

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

matrix[i][j] = matrix\_L[i][--t];

}

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

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

matrix[i][j] = matrix[j][i];

matrix[i][n] = vector\_b[i];

}

fprintf(IN, "%d\n" "%d\n", n, m);

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

fprintf(IN, form, vector\_b[i]);

fprintf(IN, "\n");

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

fprintf(IN, form, vector\_di[i]);

fprintf(IN, "\n");

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

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

fprintf(IN, form, matrix\_L[i][j]);

fprintf(IN, "\n");

}

fprintf(IN, "\n");

fprintf(IN, "%d\n" , n);

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

for (int j = 0; j < n+1; j++)

fprintf(IN, form, matrix[i][j]);

fprintf(IN, "\n");

}

}