|  |  |  |
| --- | --- | --- |
|  | | |
| 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** | | |
|  | | |
|  | Бригада | Румянцев Артём |
|  | Уваров Артём |
| Группа ПМ-23 |  |
| Вариант 6 |  |
|  |  |
|  |  |
| Преподаватели | задорожный александр геннадьевич |
|  | ЛЕОнович Дарьяна Александровна |
| Новосибирск, 2024 | | |

1. **Цель работы**

Разработать программу решения СЛАУ методом с - разложением и с 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. **Теоретическая часть**

Пусть дана система линейных алгебраических уравненийcсимметричной матрицей:

. (1.1)

Предположим, что нам удалось разложить матрицу:

(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 четырем основным этапам:

1. из элементов матрицы найти элементы матриц и ;
2. решить систему (1.5) с нижнетреугольной матрицей (прямой ход);
3. решить систему (1.4) с верхнетреугольной матрицей (обратный ход).

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

(1.6)

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

(1.7)

*…*

Решая систему (1.8), можно получить общие формулы для нахождения элементов матрицы :

(1.8)

(1.9)

1. **Набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Входные данные** | **Выходные данные** | **Назначение** |
| 1 | size: 4 2  al: 0 0  0 1  3 2  1 3  au: 0 0  0 3  1 2  6 1  di: 2  4  5  2 | al:  0.0000000 0.0000000  0.0000000 0.5000000  1.5000000 -1.0000000  0.4000000 0.4800000  au:  0.0000000 0.0000000  0.0000000 3.0000000  1.0000000 1.5000000  6.0000000 7.0000000  di: 2.0000000  2.5000000  5.0000000  -3.7600002 | Проверка работы -разложения |
| 2 | size: 4 2  al: 0 0  0 1  3 2  1 3  au: 0 0  0 3  1 2  6 1  di: 2 b: 11  4 39  5 26  2 19 | x:  1.0000000  2.0000000  3.0000000  4.0000000 | Проверка работы решения СЛАУ: |

1. **Исследование на матрицах, число обусловленности которых регулируется за счёт изменения диагонального элемента:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| k | x^k (одинарная точность) | x\* - x^k (одинарная точность) | x^k (двойная точность) | x\* - x^k (двойная точность) | x^k (скаляр. произв.) | x\* - x^k (скаляр. произв.) |
| 0 | 1,0000057 | -5,700E-06 | 0,999999999999996 | 3,997E-15 | 1,0000012 | -1,200E-06 |
| 2,0000081 | -8,100E-06 | 1,999999999999990 | 9,992E-15 | 2,0000017 | -1,700E-06 |
| 3,0000083 | -8,300E-06 | 2,999999999999990 | 1,021E-14 | 3,0000017 | -1,700E-06 |
| 4,0000081 | -8,100E-06 | 3,999999999999990 | 1,021E-14 | 4,0000019 | -1,900E-06 |
| 5,0000086 | -8,600E-06 | 4,999999999999990 | 9,770E-15 | 5,0000019 | -1,900E-06 |
| 6,0000086 | -8,600E-06 | 5,999999999999990 | 9,770E-15 | 6,0000019 | -1,900E-06 |
| 7,0000072 | -7,200E-06 | 6,999999999999990 | 9,770E-15 | 7,0000014 | -1,400E-06 |
| 8,0000095 | -9,500E-06 | 7,999999999999990 | 0,000E+00 | 8,0000019 | -1,900E-06 |
| 9,0000095 | -9,500E-06 | 8,999999999999990 | 0,000E+00 | 9,0000019 | -1,900E-06 |
| 10,0000095 | -9,500E-06 | 9,999999999999990 | 0,000E+00 | 10,0000019 | -1,900E-06 |
| 1 | 0,999926 | 7,400E-05 | 1,000000000000040 | -3,997E-14 | 0,9999671 | 3,290E-05 |
| 1,9999225 | 7,750E-05 | 2,000000000000040 | -3,997E-14 | 1,9999653 | 3,470E-05 |
| 2,9999223 | 7,770E-05 | 3,000000000000040 | -3,997E-14 | 2,9999654 | 3,460E-05 |
| 3,999923 | 7,700E-05 | 4,000000000000040 | -3,997E-14 | 3,9999654 | 3,460E-05 |
| 4,9999228 | 7,720E-05 | 5,000000000000040 | -3,997E-14 | 4,9999652 | 3,480E-05 |
| 5,9999223 | 7,770E-05 | 6,000000000000040 | -3,997E-14 | 5,9999652 | 3,480E-05 |
| 6,9999237 | 7,630E-05 | 7,000000000000040 | -3,997E-14 | 6,9999661 | 3,390E-05 |
| 7,9999223 | 7,770E-05 | 8,000000000000040 | -4,086E-14 | 7,9999652 | 3,480E-05 |
| 8,9999228 | 7,720E-05 | 9,000000000000040 | -4,086E-14 | 8,9999657 | 3,430E-05 |
| 9,9999218 | 7,820E-05 | 10,000000000000000 | 0,000E+00 | 9,9999657 | 3,430E-05 |
| 2 | 0,9993493 | 6,507E-04 | 1,000000000000060 | -5,995E-14 | 0,9994267 | 5,733E-04 |
| 1,9993467 | 6,533E-04 | 2,000000000000060 | -5,995E-14 | 1,9994243 | 5,757E-04 |
| 2,9993465 | 6,535E-04 | 3,000000000000060 | -5,995E-14 | 2,9994242 | 5,758E-04 |
| 3,999346 | 6,540E-04 | 4,000000000000060 | -6,040E-14 | 3,9994242 | 5,758E-04 |
| 4,9993467 | 6,533E-04 | 5,000000000000060 | -6,040E-14 | 4,9994245 | 5,755E-04 |
| 5,9993458 | 6,542E-04 | 6,000000000000060 | -6,040E-14 | 5,999424 | 5,760E-04 |
| 6,9993472 | 6,528E-04 | 7,000000000000060 | -6,040E-14 | 6,9994249 | 5,751E-04 |
| 7,9993467 | 6,533E-04 | 8,000000000000060 | -6,040E-14 | 7,9994245 | 5,755E-04 |
| 8,9993458 | 6,542E-04 | 9,000000000000060 | -6,040E-14 | 8,999424 | 5,760E-04 |
| 9,9993458 | 6,542E-04 | 10,000000000000000 | 0,000E+00 | 9,999424 | 5,760E-04 |
| 3 | 0,9973762 | 2,624E-03 | 1,000000000002440 | -2,440E-12 | 0,9968388 | 3,161E-03 |
| 1,9973747 | 2,625E-03 | 2,000000000002440 | -2,440E-12 | 1,9968376 | 3,162E-03 |
| 2,9973745 | 2,626E-03 | 3,000000000002440 | -2,440E-12 | 2,9968374 | 3,163E-03 |
| 3,9973748 | 2,625E-03 | 4,000000000002440 | -2,440E-12 | 3,9968374 | 3,163E-03 |
| 4,9973755 | 2,624E-03 | 5,000000000002440 | -2,440E-12 | 4,9968376 | 3,162E-03 |
| 5,9973745 | 2,625E-03 | 6,000000000002440 | -2,440E-12 | 5,9968376 | 3,162E-03 |
| 6,9973755 | 2,624E-03 | 7,000000000002440 | -2,440E-12 | 6,9968376 | 3,162E-03 |
| 7,997375 | 2,625E-03 | 8,000000000002440 | -2,441E-12 | 7,9968376 | 3,162E-03 |
| 8,9973745 | 2,626E-03 | 9,000000000002440 | -2,441E-12 | 8,9968376 | 3,162E-03 |
| 9,9973755 | 2,624E-03 | 10,000000000002400 | -2,400E-12 | 9,9968376 | 3,162E-03 |
| 4 | 0,9229438 | 7,706E-02 | 1,000000000033300 | -3,330E-11 | 0,9543321 | 4,567E-02 |
| 1,9229407 | 7,706E-02 | 2,000000000033310 | -3,331E-11 | 1,9543297 | 4,567E-02 |
| 2,9229403 | 7,706E-02 | 3,000000000033310 | -3,331E-11 | 2,9543295 | 4,567E-02 |
| 3,9229405 | 7,706E-02 | 4,000000000033310 | -3,331E-11 | 3,9543300 | 4,567E-02 |
| 4,9229398 | 7,706E-02 | 5,000000000033310 | -3,331E-11 | 4,9543295 | 4,567E-02 |
| 5,9229403 | 7,706E-02 | 6,000000000033310 | -3,331E-11 | 5,9543295 | 4,567E-02 |
| 6,9229412 | 7,706E-02 | 7,000000000033300 | -3,330E-11 | 6,9543309 | 4,567E-02 |
| 7,9229403 | 7,706E-02 | 8,000000000033310 | -3,331E-11 | 7,9543295 | 4,567E-02 |
| 8,9229403 | 7,706E-02 | 9,000000000033310 | -3,331E-11 | 8,9543295 | 4,567E-02 |
| 9,9229403 | 7,706E-02 | 10,000000000033300 | -3,330E-11 | 9,9543295 | 4,567E-02 |
| 5 | 1,7721467 | -7,721E-01 | 0,999999999267244 | 7,328E-10 | 0,75774740 | 2,423E-01 |
| 2,7721512 | -7,722E-01 | 1,999999999267240 | 7,328E-10 | 1,75774670 | 2,423E-01 |
| 3,7721517 | -7,722E-01 | 2,999999999267240 | 7,328E-10 | 2,75774670 | 2,423E-01 |
| 4,772151 | -7,722E-01 | 3,999999999267240 | 7,328E-10 | 3,75774650 | 2,423E-01 |
| 5,7721524 | -7,722E-01 | 4,999999999267240 | 7,328E-10 | 4,75774670 | 2,423E-01 |
| 6,7721519 | -7,722E-01 | 5,999999999267240 | 7,328E-10 | 5,75774620 | 2,423E-01 |
| 7,7721486 | -7,721E-01 | 6,999999999267240 | 7,328E-10 | 6,75774670 | 2,423E-01 |
| 8,7721529 | -7,722E-01 | 7,999999999267240 | 7,328E-10 | 7,75774670 | 2,423E-01 |
| 9,7721519 | -7,722E-01 | 8,999999999267240 | 7,328E-10 | 8,75774670 | 2,423E-01 |
| 10,7721519 | -7,722E-01 | 9,999999999267240 | 7,328E-10 | 9,75774670 | 2,423E-01 |
| 6 | 4,9090848 | -3,909E+00 | 0,999999999333864 | 6,661E-10 | 0,62962950 | 3,704E-01 |
| 5,9090877 | -3,909E+00 | 1,999999999333860 | 6,661E-10 | 1,62962890 | 3,704E-01 |
| 6,9090881 | -3,909E+00 | 2,999999999333860 | 6,661E-10 | 2,62962910 | 3,704E-01 |
| 7,9090867 | -3,909E+00 | 3,999999999333860 | 6,661E-10 | 3,62962890 | 3,704E-01 |
| 8,9090881 | -3,909E+00 | 4,999999999333860 | 6,661E-10 | 4,62962960 | 3,704E-01 |
| 9,9090881 | -3,909E+00 | 5,999999999333860 | 6,661E-10 | 5,62962870 | 3,704E-01 |
| 10,9090853 | -3,909E+00 | 6,999999999333860 | 6,661E-10 | 6,62962960 | 3,704E-01 |
| 11,9090891 | -3,909E+00 | 7,999999999333860 | 6,661E-10 | 7,62962960 | 3,704E-01 |
| 12,9090881 | -3,909E+00 | 8,999999999333860 | 6,661E-10 | 8,62962910 | 3,704E-01 |
| 13,909091 | -3,909E+00 | 9,999999999333860 | 6,661E-10 | 9,62963010 | 3,704E-01 |
| 7 | -16,99999810 | 1,800E+01 | 1,000000039968030 | -3,997E-08 | -5,32258030 | 6,323E+00 |
| -15,99999710 | 1,800E+01 | 2,000000039968030 | -3,997E-08 | -4,32258030 | 6,323E+00 |
| -14,99999810 | 1,800E+01 | 3,000000039968030 | -3,997E-08 | -3,32258060 | 6,323E+00 |
| -13,99999620 | 1,800E+01 | 4,000000039968030 | -3,997E-08 | -2,32258030 | 6,323E+00 |
| -12,99999900 | 1,800E+01 | 5,000000039968030 | -3,997E-08 | -1,32258060 | 6,323E+00 |
| -11,99999710 | 1,800E+01 | 6,000000039968030 | -3,997E-08 | -0,32258050 | 6,323E+00 |
| -10,99999710 | 1,800E+01 | 7,000000039968030 | -3,997E-08 | 0,67741960 | 6,323E+00 |
| -9,99999900 | 1,800E+01 | 8,000000039968030 | -3,997E-08 | 1,67741940 | 6,323E+00 |
| -8,99999810 | 1,800E+01 | 9,000000039968030 | -3,997E-08 | 2,67741940 | 6,323E+00 |
| -8,00000000 | 1,800E+01 | 10,000000039968000 | -3,997E-08 | 3,67741940 | 6,323E+00 |
| 8 |  | | 1,000000022204460 | -2,220E-08 |  | | |
| 2,000000022204460 | -2,220E-08 |
| 3,000000022204460 | -2,220E-08 |
| 4,000000022204460 | -2,220E-08 |
| 5,000000022204460 | -2,220E-08 |
| 6,000000022204460 | -2,220E-08 |
| 7,000000022204460 | -2,220E-08 |
| 8,000000022204460 | -2,220E-08 |
| 9,000000022204460 | -2,220E-08 |
| 10,000000022204400 | -2,220E-08 |
| 9 |  | | 1,000004218851820 | -4,219E-06 |  | | |
| 2,000004218851820 | -4,219E-06 |
| 3,000004218851820 | -4,219E-06 |
| 4,000004218851820 | -4,219E-06 |
| 5,000004218851820 | -4,219E-06 |
| 6,000004218851820 | -4,219E-06 |
| 7,000004218851820 | -4,219E-06 |
| 8,000004218851820 | -4,219E-06 |
| 9,000004218851830 | -4,219E-06 |
| 10,000004218851800 | -4,219E-06 |
| 10 |  | | 0,999928946994462 | 7,105E-05 |  | | |
| 1,999928946994450 | 7,105E-05 |
| 2,999928946994450 | 7,105E-05 |
| 3,999928946994450 | 7,105E-05 |
| 4,999928946994450 | 7,105E-05 |
| 5,999928946994450 | 7,105E-05 |
| 6,999928946994460 | 7,105E-05 |
| 7,999928946994450 | 7,105E-05 |
| 8,999928946994450 | 7,105E-05 |
| 9,999928946994450 | 7,105E-05 |
| 11 |  | | 1,001132565705450 | -1,133E-03 |  | | |
| 2,001132565705460 | -1,133E-03 |
| 3,001132565705460 | -1,133E-03 |
| 4,001132565705460 | -1,133E-03 |
| 5,001132565705460 | -1,133E-03 |
| 6,001132565705460 | -1,133E-03 |
| 7,001132565705460 | -1,133E-03 |
| 8,001132565705460 | -1,133E-03 |
| 9,001132565705460 | -1,133E-03 |
| 10,001132565705400 | -1,133E-03 |
| 12 |  | | 1,007552199022650 | -7,552E-03 |  | | |
| 2,007552199022650 | -7,552E-03 |
| 3,007552199022650 | -7,552E-03 |
| 4,007552199022650 | -7,552E-03 |
| 5,007552199022650 | -7,552E-03 |
| 6,007552199022650 | -7,552E-03 |
| 7,007552199022650 | -7,552E-03 |
| 8,007552199022650 | -7,552E-03 |
| 9,007552199022660 | -7,552E-03 |
| 10,007552199022600 | -7,552E-03 |
| 13 |  | | 1,103139013452900 | -1,031E-01 |  | | |
| 2,103139013452910 | -1,031E-01 |
| 3,103139013452910 | -1,031E-01 |
| 4,103139013452910 | -1,031E-01 |
| 5,103139013452910 | -1,031E-01 |
| 6,103139013452910 | -1,031E-01 |
| 7,103139013452900 | -1,031E-01 |
| 8,103139013452910 | -1,031E-01 |
| 9,103139013452910 | -1,031E-01 |
| 10,103139013452900 | -1,031E-01 |
| 14 |  | | 4,344827586206880 | -3,345E+00 |  | | |
| 5,344827586206890 | -3,345E+00 |
| 6,344827586206890 | -3,345E+00 |
| 7,344827586206890 | -3,345E+00 |
| 8,344827586206890 | -3,345E+00 |
| 9,344827586206890 | -3,345E+00 |
| 10,344827586206800 | -3,345E+00 |
| 11,344827586206800 | -3,345E+00 |
| 12,344827586206800 | -3,345E+00 |
| 13,344827586206800 | -3,345E+00 |

Матрица, рассматриваемая в задании, при малых значениях k лучше всего обусловлена и с каждым увеличением k её обусловленность становится хуже. С одинарной точностью программа работает до k = 5, при k = 6 наблюдается скачок погрешности, а при k = 7 и больше матрица становится вырожденной и значения вектора *x* остаются неизменными, так как точность не позволяет работать со столь малыми отклонениями и происходит ошибка округления. (Так как у типа float всего лишь 7 (7.5) значащих цифр после запятой, то при достижении k, например равного 7, в памяти вместо числа 3.0000001 будет хранится всего лишь 3.0000000. В данном случае мы получим вырожденную матрицу (исходную без ), число обусловленности которой стремится к бесконечности.)

Точность вносит погрешность вычисления, связанную с ошибкой округления, что является дополнительным фактором возмущения для матрицы, а так как с каждым увеличением k матрица становится всё хуже и хуже обусловленной, влияние на результат небольших ошибок вычислений становится всё больше и больше.

Если рассматривать результаты с двойной точностью только для скалярных произведений, можно заметить, что программа работает до k = 6, а с k = 7 и далее ситуация с одинарной точностью повторяется. Это происходит потому, что многократное суммирование произведений сильно подвержено накоплению ошибок округления, а скалярное произведение и является таковым суммированием. Соответственно, при увеличении точности скалярных произведений накопление ошибок округления оказывает меньшее влияние на результат, но так как точность результата остаётся одинарной, то, начиная с k = 7, матрица опять же становится вырожденной.

Двойная точность обеспечивает гораздо меньшую погрешность и работает вплоть до k = 12, затем происходит повторение ситуации с одинарной точностью, причиной этому является хранение большего количества знаков после запятой для двойной точности.

1. **Исследование на матрицах Гильберта различной размерности:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| k | x^k (одинарная точность) | x\* - x^k (одинарная точность) | x^k (двойная точность) | x\* - x^k (двойная точность) | x^k (скаляр. произв.) | x\* - x^k (скаляр. произв.) |
| 2 | 1,0000004 | -4,000E-07 | 0,9999999999999990 | 0,000E+00 | 1,0000004 | -4,000E-07 |
| 1,9999993 | 7,000E-07 | 2,0000000000000000 | 0,000E+00 | 1,9999993 | 7,000E-07 |
| 3 | 1,0000019 | -1,900E-06 | 0,9999999999999930 | 6,994E-15 | 1,0000002 | -2,000E-07 |
| 1,9999889 | 1,110E-05 | 2,0000000000000400 | -3,997E-14 | 1,9999995 | 5,000E-07 |
| 3,0000107 | -1,070E-05 | 2,9999999999999600 | 3,997E-14 | 3,0000000 | 0,000E+00 |
| 4 | 0,9999881 | 1,190E-05 | 1,0000000000000300 | -2,998E-14 | 0,9999819 | 1,810E-05 |
| 2,0001216 | -1,216E-04 | 1,9999999999996000 | 3,999E-13 | 2,0001898 | -1,898E-04 |
| 2,9997284 | 2,716E-04 | 3,0000000000009500 | -9,499E-13 | 2,9995661 | 4,339E-04 |
| 4,0001669 | -1,669E-04 | 3,9999999999993700 | 6,302E-13 | 4,0002713 | -2,713E-04 |
| 5 | 0,9999733 | 2,670E-05 | 1,0000000000002700 | -2,700E-13 | 0,9999899 | 1,010E-05 |
| 2,0005672 | -5,672E-04 | 1,9999999999952600 | 4,740E-12 | 2,0002527 | -2,527E-04 |
| 2,9974306 | 2,569E-03 | 3,0000000000193800 | -1,938E-11 | 2,9987824 | 1,218E-03 |
| 4,0039706 | -3,971E-03 | 3,9999999999717500 | 2,825E-11 | 4,0019460 | -1,946E-03 |
| 4,9980354 | 1,965E-03 | 5,0000000000134600 | -1,346E-11 | 4,9990153 | 9,847E-04 |
| 6 | 1,0005798 | -5,798E-04 | 0,9999999999984830 | 1,517E-12 | 0,9997098 | 2,902E-04 |
| 1,9862506 | 1,375E-02 | 2,0000000000438600 | -4,386E-11 | 2,0111251 | -1,113E-02 |
| 3,0808041 | -8,080E-02 | 2,9999999996993000 | 3,007E-10 | 2,9126339 | 8,737E-02 |
| 3,8124714 | 1,875E-01 | 4,0000000007904600 | -7,905E-10 | 4,2492094 | -2,492E-01 |
| 5,1881227 | -1,881E-01 | 4,9999999991201700 | 8,798E-10 | 4,7069321 | 2,931E-01 |
| 5,9317074 | 6,829E-02 | 6,0000000003489900 | -3,490E-10 | 6,1209655 | -1,210E-01 |
| 7 | 0,9932084 | 6,792E-03 | 0,9999999999551030 | 4,490E-11 | 1,0023872 | -2,387E-03 |
| 2,2732069 | -2,732E-01 | 2,0000000017699000 | -1,770E-09 | 1,9010653 | 9,893E-02 |
| 0,363138 | 2,637E+00 | 2,9999999830924300 | 1,691E-08 | 3,9734063 | -9,734E-01 |
| 14,249752 | -1,025E+01 | 4,0000000653219200 | -6,532E-08 | 0,1674219 | 3,833E+00 |
| -13,7752638 | 1,878E+01 | 4,9999998808216100 | 1,192E-07 | 12,0834398 | -7,083E+00 |
| 22,2069321 | -1,621E+01 | 6,0000001025877300 | -1,026E-07 | -0,1541072 | 6,154E+00 |
| 1,6842105 | 5,316E+00 | 6,9999999664212800 | 3,358E-08 | 9,0282660 | -2,028E+00 |
| 8 |  | | 1,0000000001050600 | -1,051E-10 | 0,9948187 | 5,181E-03 |
| 1,9999999946247000 | 5,375E-09 | 2,1978490 | -1,978E-01 |
| 3,0000000677828700 | -6,778E-08 | 1,2112877 | 1,789E+00 |
| 3,9999996430472000 | 3,570E-07 | 10,3103600 | -6,310E+00 |
| 5,0000009401145500 | -9,401E-07 | -4,7052107 | 9,705E+00 |
| 5,9999986936684800 | 1,306E-06 | 11,2836428 | -5,284E+00 |
| 7,0000009156030800 | -9,156E-07 | 8,2559834 | -1,256E+00 |
| 7,9999997450004400 | 2,550E-07 | 6,4473042 | 1,553E+00 |
| 9 |  | | 1,0000000023912600 | -2,391E-09 |  | |
| 1,9999998344843400 | 1,655E-07 |
| 3,0000028073746300 | -2,807E-06 |
| 3,9999799277027300 | 2,007E-05 |
| 5,0000737115242300 | -7,371E-05 |
| 5,9998493526130600 | 1,506E-04 |
| 7,0001731427185400 | -1,731E-04 |
| 7,9998953537201100 | 1,046E-04 |
| 9,0000258691790100 | -2,587E-05 |
| 10 |  | | 0,9999999995087570 | 4,912E-10 |  | |
| 2,0000000281022000 | -2,810E-08 |
| 2,9999996401618800 | 3,598E-07 |
| 4,0000015279135600 | -1,528E-06 |
| 4,9999995411485300 | 4,589E-07 |
| 5,9999855647448700 | 1,444E-05 |
| 7,0000455408848600 | -4,554E-05 |
| 7,9999378362680500 | 6,216E-05 |
| 9,0000409152607000 | -4,092E-05 |
| 9,9999894053965800 | 1,059E-05 |
| 11 |  | | 0,9999999785663880 | 2,143E-08 |  | |
| 2,0000021680262800 | -2,168E-06 |
| 2,9999453589249500 | 5,464E-05 |
| 4,0005960262579200 | -5,960E-04 |
| 4,9965241047139300 | 3,476E-03 |
| 6,0119952232147000 | -1,200E-02 |
| 6,9743080042682400 | 2,569E-02 |
| 8,0345175794181100 | -3,452E-02 |
| 8,9717005274035800 | 2,830E-02 |
| 10,0129400546668000 | -1,294E-02 |
| 10,9974709656943000 | 2,529E-03 |
| 12 |  | | 1,0000000877690900 | -8,777E-08 |  | |
| 1,9999894813465800 | 1,052E-05 |
| 3,0003152028944400 | -3,152E-04 |
| 3,9958824585697400 | 4,118E-03 |
| 5,0290873287582700 | -2,909E-02 |
| 5,8763272828534500 | 1,237E-01 |
| 7,3346201874952100 | -3,346E-01 |
| 7,4100594730333400 | 5,899E-01 |
| 9,6753389581457900 | -6,753E-01 |
| 9,5159907898586100 | 4,840E-01 |
| 11,1972991488980000 | -1,973E-01 |
| 11,9650895728066000 | 3,491E-02 |
| 13 |  | | 1,000006639764000 | -6,640E-06 |  | |
| 1,998947735922750 | 1,052E-03 |
| 3,040909240802390 | -4,091E-02 |
| 3,313968421135690 | 6,860E-01 |
| 11,198427773421600 | -6,198E+00 |
| -27,801817663856000 | 3,380E+01 |
| 125,479842845505000 | -1,185E+02 |
| -267,868467658179000 | 2,759E+02 |
| 440,194053270946000 | -4,312E+02 |
| -437,177449082592000 | 4,472E+02 |
| 306,080442290202000 | -2,951E+02 |
| -100,140449185705000 | 1,121E+02 |
| 31,681592039800900 | -1,868E+01 |

Числа обусловленности для матриц Гильберта разных размеров

|  |  |
| --- | --- |
| Одинарная точность | Двойная точность |
| = 1.93\*  = 5.24\*  = 1.55\*  = 4.76\*  = 1.46\*  = 3.34\*  = 8.75\*  = 9.65\* | = 1.93\*  = 5.24\*  = 1.55\*  = 4.77\*  = 1.50\*  = 4.75\*  = 1.53\*  = 4.93\*  = 1.60\*  = 5.23\*  = 1.62\*  = 4.95\* |

Строки (или столбцы) матрицы Гильберта становятся близки к линейно зависимым по мере увеличения размера матрицы (то есть с увеличением параметра k). Это означает, что небольшие изменения в элементах могут привести к большим изменениям в решениях системы уравнений.

Как правило, если число обусловленности , то вы можете потерять до m цифр точности сверх того, что будет потеряно для числового значения из-за потери точности из арифметических методов. Можно заметить, что при k = 7 для одинарной точности m будет равно 8, то есть потеря цифр точности при вычислениях будет больше количества значащих цифр для формата одинарной точности, поэтому появляются неверные значения в решении.

При двойной точности только скалярных произведений вплоть до k = 6 включительно погрешность примерно одинакова с результатами одинарной точности.

Для двойной точности то же самое, что и для одинарной, но для k = 13 и m = 17, соответственно.

1. **Расчет количества действий LU(\*)-разложения:**

Количество действий, необходимое для получения элементов матрицы U:

Количество действий, необходимое для получения элементов матрицы L:

Прямой ход:

Обратный ход:

Итоговая сложность:

1. **Сравнение метода Гаусса и метода LU(\*)-разложения**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | x^k (двойная точность) – LU(\*) | x\* - x^k (двойная точность) – LU(\*) | x^k (двойная точность) - Гаусс | x\* - x^k (двойная точность) - Гаусс |
| 0 | 0,999999999999994 | 5,995E-15 | 0,999999999999955 | 4,496E-14 |
| 1,999999999999990 | 9,992E-15 | 1,999999999999900 | 9,992E-14 |
| 2,999999999999990 | 1,021E-14 | 2,999999999999900 | 9,992E-14 |
| 3,999999999999990 | 1,021E-14 | 3,999999999999900 | 9,992E-14 |
| 4,999999999999990 | 9,770E-15 | 4,999999999999900 | 1,004E-13 |
| 5,999999999999990 | 9,770E-15 | 5,999999999999900 | 1,004E-13 |
| 6,999999999999990 | 9,770E-15 | 6,999999999999900 | 1,004E-13 |
| 7,999999999999990 | 0,000E+00 | 7,999999999999900 | 1,004E-13 |
| 8,999999999999990 | 0,000E+00 | 8,999999999999900 | 9,948E-14 |
| 9,999999999999980 | 1,954E-14 | 9,999999999999900 | 9,948E-14 |
| 1 | 0,999999999999834 | 1,660E-13 | 0,999999999999435 | 5,650E-13 |
| 1,999999999999820 | 1,801E-13 | 1,999999999999370 | 6,299E-13 |
| 2,999999999999820 | 1,799E-13 | 2,999999999999370 | 6,302E-13 |
| 3,999999999999820 | 1,799E-13 | 3,999999999999370 | 6,302E-13 |
| 4,999999999999820 | 1,803E-13 | 4,999999999999370 | 6,297E-13 |
| 5,999999999999820 | 1,803E-13 | 5,999999999999370 | 6,297E-13 |
| 6,999999999999820 | 1,803E-13 | 6,999999999999370 | 6,297E-13 |
| 7,999999999999820 | 1,803E-13 | 7,999999999999370 | 6,297E-13 |
| 8,999999999999820 | 1,794E-13 | 8,999999999999370 | 6,306E-13 |
| 9,999999999999820 | 1,794E-13 | 9,999999999999370 | 6,306E-13 |
| 2 | 1,000000000001750 | -1,750E-12 | 0,999999999999288 | 7,120E-13 |
| 2,000000000001750 | -1,750E-12 | 1,999999999999280 | 7,201E-13 |
| 3,000000000001750 | -1,750E-12 | 2,999999999999280 | 7,199E-13 |
| 4,000000000001750 | -1,750E-12 | 3,999999999999280 | 7,199E-13 |
| 5,000000000001750 | -1,750E-12 | 4,999999999999280 | 7,203E-13 |
| 6,000000000001750 | -1,750E-12 | 5,999999999999280 | 7,203E-13 |
| 7,000000000001750 | -1,750E-12 | 6,999999999999280 | 7,203E-13 |
| 8,000000000001750 | -1,750E-12 | 7,999999999999280 | 7,203E-13 |
| 9,000000000001750 | -1,750E-12 | 8,999999999999270 | 7,301E-13 |
| 10,000000000001700 | -1,700E-12 | 9,999999999999280 | 7,194E-13 |
| 3 | 1,000000000009560 | -9,560E-12 | 0,999999999989341 | 1,066E-11 |
| 2,000000000009560 | -9,560E-12 | 1,999999999989320 | 1,068E-11 |
| 3,000000000009560 | -9,560E-12 | 2,999999999989330 | 1,067E-11 |
| 4,000000000009560 | -9,560E-12 | 3,999999999989320 | 1,068E-11 |
| 5,000000000009560 | -9,560E-12 | 4,999999999989320 | 1,068E-11 |
| 6,000000000009560 | -9,560E-12 | 5,999999999989330 | 1,067E-11 |
| 7,000000000009560 | -9,560E-12 | 6,999999999989320 | 1,068E-11 |
| 8,000000000009560 | -9,560E-12 | 7,999999999989330 | 1,067E-11 |
| 9,000000000009560 | -9,560E-12 | 8,999999999989320 | 1,068E-11 |
| 10,000000000009500 | -9,500E-12 | 9,999999999989320 | 1,068E-11 |
| 4 | 1,000000000159890 | -1,599E-10 | 0,999999999600315 | 3,997E-10 |
| 2,000000000159890 | -1,599E-10 | 1,999999999600270 | 3,997E-10 |
| 3,000000000159900 | -1,599E-10 | 2,999999999600270 | 3,997E-10 |
| 4,000000000159900 | -1,599E-10 | 3,999999999600270 | 3,997E-10 |
| 5,000000000159890 | -1,599E-10 | 4,999999999600270 | 3,997E-10 |
| 6,000000000159900 | -1,599E-10 | 5,999999999600270 | 3,997E-10 |
| 7,000000000159890 | -1,599E-10 | 6,999999999600270 | 3,997E-10 |
| 8,000000000159890 | -1,599E-10 | 7,999999999600270 | 3,997E-10 |
| 9,000000000159900 | -1,599E-10 | 8,999999999600270 | 3,997E-10 |
| 10,000000000159800 | -1,598E-10 | 9,999999999600270 | 3,997E-10 |
| 5 | 1,000000001754170 | -1,754E-09 | 0,999999996891372 | 3,109E-09 |
| 2,000000001754170 | -1,754E-09 | 1,999999996891330 | 3,109E-09 |
| 3,000000001754180 | -1,754E-09 | 2,999999996891330 | 3,109E-09 |
| 4,000000001754180 | -1,754E-09 | 3,999999996891330 | 3,109E-09 |
| 5,000000001754180 | -1,754E-09 | 4,999999996891330 | 3,109E-09 |
| 6,000000001754180 | -1,754E-09 | 5,999999996891340 | 3,109E-09 |
| 7,000000001754170 | -1,754E-09 | 6,999999996891330 | 3,109E-09 |
| 8,000000001754180 | -1,754E-09 | 7,999999996891330 | 3,109E-09 |
| 9,000000001754180 | -1,754E-09 | 8,999999996891330 | 3,109E-09 |
| 10,000000001754100 | -1,754E-09 | 9,999999996891330 | 3,109E-09 |
| 6 | 0,999999972466437 | 2,753E-08 | 0,999999972466466 | 2,753E-08 |
| 1,999999972466420 | 2,753E-08 | 1,999999972466430 | 2,753E-08 |
| 2,999999972466420 | 2,753E-08 | 2,999999972466430 | 2,753E-08 |
| 3,999999972466420 | 2,753E-08 | 3,999999972466430 | 2,753E-08 |
| 4,999999972466420 | 2,753E-08 | 4,999999972466430 | 2,753E-08 |
| 5,999999972466420 | 2,753E-08 | 5,999999972466430 | 2,753E-08 |
| 6,999999972466430 | 2,753E-08 | 6,999999972466430 | 2,753E-08 |
| 7,999999972466420 | 2,753E-08 | 7,999999972466430 | 2,753E-08 |
| 8,999999972466420 | 2,753E-08 | 8,999999972466430 | 2,753E-08 |
| 9,999999972466420 | 2,753E-08 | 9,999999972466430 | 2,753E-08 |
| 7 | 1,000000073274730 | -7,327E-08 | 0,999999653610426 | 3,464E-07 |
| 2,000000073274730 | -7,327E-08 | 1,999999653610380 | 3,464E-07 |
| 3,000000073274730 | -7,327E-08 | 2,999999653610390 | 3,464E-07 |
| 4,000000073274730 | -7,327E-08 | 3,999999653610390 | 3,464E-07 |
| 5,000000073274730 | -7,327E-08 | 4,999999653610390 | 3,464E-07 |
| 6,000000073274730 | -7,327E-08 | 5,999999653610390 | 3,464E-07 |
| 7,000000073274730 | -7,327E-08 | 6,999999653610390 | 3,464E-07 |
| 8,000000073274730 | -7,327E-08 | 7,999999653610390 | 3,464E-07 |
| 9,000000073274730 | -7,327E-08 | 8,999999653610380 | 3,464E-07 |
| 10,000000073274700 | -7,327E-08 | 9,999999653610390 | 3,464E-07 |
| 8 | 1,000002020606800 | -2,021E-06 | 0,999999111821662 | 8,882E-07 |
| 2,000002020606810 | -2,021E-06 | 1,999999111821650 | 8,882E-07 |
| 3,000002020606810 | -2,021E-06 | 2,999999111821650 | 8,882E-07 |
| 4,000002020606810 | -2,021E-06 | 3,999999111821650 | 8,882E-07 |
| 5,000002020606810 | -2,021E-06 | 4,999999111821650 | 8,882E-07 |
| 6,000002020606810 | -2,021E-06 | 5,999999111821650 | 8,882E-07 |
| 7,000002020606810 | -2,021E-06 | 6,999999111821650 | 8,882E-07 |
| 8,000002020606810 | -2,021E-06 | 7,999999111821650 | 8,882E-07 |
| 9,000002020606810 | -2,021E-06 | 8,999999111821650 | 8,882E-07 |
| 10,000002020606800 | -2,021E-06 | 9,999999111821650 | 8,882E-07 |
| 9 | 0,999998223640943 | 1,776E-06 | 0,999985789171719 | 1,421E-05 |
| 1,999998223640940 | 1,776E-06 | 1,999985789171700 | 1,421E-05 |
| 2,999998223640940 | 1,776E-06 | 2,999985789171700 | 1,421E-05 |
| 3,999998223640940 | 1,776E-06 | 3,999985789171700 | 1,421E-05 |
| 4,999998223640940 | 1,776E-06 | 4,999985789171700 | 1,421E-05 |
| 5,999998223640940 | 1,776E-06 | 5,999985789171700 | 1,421E-05 |
| 6,999998223640940 | 1,776E-06 | 6,999985789171700 | 1,421E-05 |
| 7,999998223640940 | 1,776E-06 | 7,999985789171700 | 1,421E-05 |
| 8,999998223640940 | 1,776E-06 | 8,999985789171700 | 1,421E-05 |
| 9,999998223640940 | 1,776E-06 | 9,999985789171700 | 1,421E-05 |
| 10 | 0,999933386771949 | 6,661E-05 | 0,999928946363394 | 7,105E-05 |
| 1,999933386771940 | 6,661E-05 | 1,999928946363380 | 7,105E-05 |
| 2,999933386771940 | 6,661E-05 | 2,999928946363380 | 7,105E-05 |
| 3,999933386771940 | 6,661E-05 | 3,999928946363380 | 7,105E-05 |
| 4,999933386771940 | 6,661E-05 | 4,999928946363380 | 7,105E-05 |
| 5,999933386771940 | 6,661E-05 | 5,999928946363380 | 7,105E-05 |
| 6,999933386771940 | 6,661E-05 | 6,999928946363380 | 7,105E-05 |
| 7,999933386771940 | 6,661E-05 | 7,999928946363380 | 7,105E-05 |
| 8,999933386771940 | 6,661E-05 | 8,999928946363380 | 7,105E-05 |
| 9,999933386771940 | 6,661E-05 | 9,999928946363380 | 7,105E-05 |
| 11 | 1,000932732239210 | -9,327E-04 | 0,997247624966735 | 2,752E-03 |
| 2,000932732239220 | -9,327E-04 | 1,997247624966700 | 2,752E-03 |
| 3,000932732239220 | -9,327E-04 | 2,997247624966700 | 2,752E-03 |
| 4,000932732239220 | -9,327E-04 | 3,997247624966700 | 2,752E-03 |
| 5,000932732239220 | -9,327E-04 | 4,997247624966700 | 2,752E-03 |
| 6,000932732239220 | -9,327E-04 | 5,997247624966700 | 2,752E-03 |
| 7,000932732239210 | -9,327E-04 | 6,997247624966700 | 2,752E-03 |
| 8,000932732239220 | -9,327E-04 | 7,997247624966700 | 2,752E-03 |
| 9,000932732239220 | -9,327E-04 | 8,997247624966700 | 2,752E-03 |
| 10,000932732239200 | -9,327E-04 | 9,997247624966700 | 2,752E-03 |
| 12 | 0,996895098691506 | 3,105E-03 | 0,961980548187485 | 3,802E-02 |
| 1,996895098691500 | 3,105E-03 | 1,961980548187440 | 3,802E-02 |
| 2,996895098691500 | 3,105E-03 | 2,961980548187440 | 3,802E-02 |
| 3,996895098691500 | 3,105E-03 | 3,961980548187440 | 3,802E-02 |
| 4,996895098691500 | 3,105E-03 | 4,961980548187440 | 3,802E-02 |
| 5,996895098691500 | 3,105E-03 | 5,961980548187440 | 3,802E-02 |
| 6,996895098691500 | 3,105E-03 | 6,961980548187440 | 3,802E-02 |
| 7,996895098691500 | 3,105E-03 | 7,961980548187440 | 3,802E-02 |
| 8,996895098691500 | 3,105E-03 | 8,961980548187440 | 3,802E-02 |
| 9,996895098691500 | 3,105E-03 | 9,961980548187440 | 3,802E-02 |
| 13 | 0,951111111111111 | 4,889E-02 | 0,666666666666702 | 3,333E-01 |
| 1,951111111111100 | 4,889E-02 | 1,666666666666660 | 3,333E-01 |
| 2,951111111111110 | 4,889E-02 | 2,666666666666660 | 3,333E-01 |
| 3,951111111111100 | 4,889E-02 | 3,666666666666660 | 3,333E-01 |
| 4,951111111111110 | 4,889E-02 | 4,666666666666660 | 3,333E-01 |
| 5,951111111111100 | 4,889E-02 | 5,666666666666660 | 3,333E-01 |
| 6,951111111111100 | 4,889E-02 | 6,666666666666660 | 3,333E-01 |
| 7,951111111111110 | 4,889E-02 | 7,666666666666660 | 3,333E-01 |
| 8,951111111111100 | 4,889E-02 | 8,666666666666660 | 3,333E-01 |
| 9,951111111111110 | 4,889E-02 | 9,666666666666660 | 3,333E-01 |
| 14 | 0,066666666666671 | 9,333E-01 | -0,733333333333316 | 1,733E+00 |
| 1,066666666666660 | 9,333E-01 | 0,266666666666665 | 1,733E+00 |
| 2,066666666666660 | 9,333E-01 | 1,266666666666660 | 1,733E+00 |
| 3,066666666666660 | 9,333E-01 | 2,266666666666660 | 1,733E+00 |
| 4,066666666666660 | 9,333E-01 | 3,266666666666660 | 1,733E+00 |
| 5,066666666666660 | 9,333E-01 | 4,266666666666660 | 1,733E+00 |
| 6,066666666666660 | 9,333E-01 | 5,266666666666660 | 1,733E+00 |
| 7,066666666666660 | 9,333E-01 | 6,266666666666660 | 1,733E+00 |
| 8,066666666666660 | 9,333E-01 | 7,266666666666660 | 1,733E+00 |
| 9,066666666666660 | 9,333E-01 | 8,266666666666660 | 1,733E+00 |

Порядки погрешностей обоих методов примерно равны.

**Расчет действий (метод Гаусса с выделением ведущего элемента):**

Прямой ход:

Обратный ход:

Итоговая сложность:

Вычислительная сложность метода Гаусса с выбором ведущего элемента и метода Гаусса на основе LU – разложения для произвольных матриц одинакова (). Однако, если производится работа с одной и той же матрицей, то использовать вариант с LU – разложением эффективнее, так как основная вычислительная сложность алгоритма заключается в нахождении как раз LU – разложения, а для метода Гаусса, в свою очередь, с каждым изменением вектора правой части необходимо заново искать ведущие элементы и приводить матрицу к треугольному виду.

При работе с плотными матрицами в обычном формате хранения для Гаусса и ленточном для LU(\*)разложения Гаусс выигрывает так как требует *n2* памяти для хранения матрицы, в то время как LU(\*) разложение – *2\*n\*(n-1)*. В случае ленточной матрицы с шириной ленты значительно меньшей, чем количество строк в матрице, наоборот выигрывает LU(\*) разложение, так как оно требует *2\*n\*m*, где m значительно меньше n, а метод Гаусса так же *n2*.

Можно сделать вывод, что среди этих двух методов нет однозначно лучшего, так как методы имеют свои ограничения и каждый из них может являться более предпочтительным в зависимости от условий задачи.

1. **Расчёт действий для LLT-разложения**

Количество действий, необходимое для получения диагональных элементов матрицы L:

Количество действий, необходимое для получения внедиагональных элементов матрицы L:

Прямой ход:

Обратный ход:

Итоговая сложность метода:

Сложности методов LU(\*) разложения и LLT разложения совпадают и равны . При стремлении *n* к бесконечности LLT количество действий в 3 раза меньше чем для LU(\*). Но при этом, первое разложение имеет ограничения в виде положительной определенности и симметричности.

1. Текст программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <fstream>

#include <iostream>

#include <math.h>

#include <algorithm>

#include <string>

#include <stdio.h>

#define FF

#ifdef FF

**typedef** float real**,** realsum**;**

#define realout "%.7f "

#define realin "%f"

#endif

#ifdef DD

**typedef** double real**,** realsum**;**

#define realout "%.15f "

#define realin "%lf"

#endif

#ifdef FD

**typedef** float real**;**

**typedef** double realsum**;**

#define realout "%.7f "

#define realin "%f"

#endif

**using** **namespace** std**;**

class matrix **{**

public**:**

int n**,** m**,** k**;**

real**\*\*** al**,** **\*\*** au**,** **\*\*** A**;**

real**\*** di**,** **\*** b**,** **\*** x**,** **\*** y**;**

real**\*\*** mat**;**

void allocateMemory**();**

void calcLU**();**

void input**(**FILE**\*** inmat**,** FILE**\*** invec**);**

void calcY**(**real**\*\*** al**,** real**\*** b**,** real**\*** y**,** int n**);**

void calcX**(**real**\*\*** au**,** real**\*** y**,** real**\*** x**,** int n**);**

void createGilbert**(**real**\*\*** gl**,** real**\*\*** gu**,** real**\*** di**,** int n**,** int m**);**

void gauss**();**

**};**

void matrix**::**input**(**FILE**\*** inmat**,** FILE**\*** invec**)** **{**

fscanf\_s**(**inmat**,** "%d %d"**,** **&**n**,** **&**m**);**

//k = 10;

allocateMemory**();**

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

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

fscanf\_s**(**inmat**,** realin**,** **&**al**[**i**][**j**]);**

**}**

**}**

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

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

fscanf\_s**(**inmat**,** realin**,** **&**au**[**i**][**j**]);**

**}**

**}**

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

fscanf\_s**(**inmat**,** realin**,** **&**di**[**i**]);**

di**[**0**]** **+=** pow**(**10.0**,** **-**k**);**

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

fscanf\_s**(**invec**,** realin**,** **&**b**[**i**]);**

**}**

b**[**0**]** **+=** pow**(**10.0**,** **-**k**);**

fclose**(**inmat**);**

fclose**(**invec**);**

**}**

void matrix**::**calcLU**()**

**{**

**for** **(**int ij **=** 0**;** ij **<** n **-** 1**;** ij**++)**

**{**

int ijl **=** ij**;**

int ilu **=** ij **+** 1**;**

**for** **(**int i **=** ij **+** 1**;** i **<** n**;** i**++)**

**{**

int il **=** i**;**

int jl **=** ij **+** m **-** i**;**

**if** **(**il **<** 0 **||** jl **<** 0**)** **continue;**

realsum sum **=** 0.0**;**

**for** **(**int k **=** 0**;** k **<** ij**;** k**++)**

**{**

int kll **=** k **+** m **-** i**;**

int klu **=** ij**;**

**if** **(**kll **<** 0 **||** klu **<** 0**)** **continue;**

int jlu **=** k **+** m **-** ij**;**

**if** **(**jlu **<** 0**)** **continue;**

sum **+=** al**[**il**][**kll**]** **\*** au**[**klu**][**jlu**];**

**}**

al**[**il**][**jl**]** **=** **(**al**[**il**][**jl**]** **-** sum**)** **/** di**[**ij**];** // au[ijl][ijl];

**}**

int il **=** ilu**;**

**for** **(**int j **=** il**;** j **<** n**;** j**++)**

**{**

int jl **=** il **+** m **-** j**;**

**if** **(**il **<** 0 **||** jl **<** 0**)** **continue;**

realsum sum **=** 0.0**;**

**for** **(**int k **=** 0**;** k **<** ilu**;** k**++)**

**{**

int kll **=** k **+** m **-** ilu**;**

int klu **=** j**;**

**if** **(**kll **<** 0 **||** klu **<** 0**)** **continue;**

int jlu **=** k **+** m **-** j**;**

**if** **(**jlu **<** 0**)** **continue;**

sum **+=** al**[**il**][**kll**]** **\*** au**[**klu**][**jlu**];**

**}**

**if** **(**jl **==** m**)** **{**

di**[**il**]** **=** **(**di**[**il**]** **-** sum**);**

**}**

**else** **{**

int klu **=** j**;**

au**[**klu**][**jl**]** **=** **(**au**[**klu**][**jl**]** **-** sum**);**

**}**

**}**

**}**

**}**

void matrix**::**calcY**(**real**\*\*** al**,** real**\*** b**,** real**\*** y**,** int n**)**

**{**

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

**{**

realsum sum **=** 0.0**;**

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

**if** **(**j **+** m **-** i **<** 0**)** **continue;**

sum **+=** al**[**i**][**j **+** m **-** i**]** **\*** y**[**j**];**

**}**

y**[**i**]** **=** **(**b**[**i**]** **-** sum**);**

**}**

**}**

void matrix**::**calcX**(**real**\*\*** au**,** real**\*** y**,** real**\*** x**,** int n**)**

**{**

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

**{**

realsum sum **=** 0.0**;**

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

**if** **(**i **+** m **-** j **<** 0**)** **continue;**

sum **+=** au**[**j**][**i **+** m **-** j**]** **\*** x**[**j**];**

**}**

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

**}**

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

printf**(**realout "\n"**,** x**[**i**]);**

**}**

void matrix**::**createGilbert**(**real**\*\*** gl**,** real**\*\*** gu**,** real**\*** di**,** int n**,** int m**)**

**{**

m **=** n **-** 1**;**

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

**{**

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

**{**

**if** **(**j **<** 0**)** gl**[**i**][**j **+** m **-** i**]** **=** gu**[**i**][**j **+** m **-** i**]** **=** 0**;**

**else** gl**[**i**][**j **+** m **-** i**]** **=** gu**[**i**][**j **+** m **-** i**]** **=** 1.0 **/** **(**i **+** j **+** 1**);**

**}**

di**[**i**]** **=** 1.0 **/** **(**i **+** i **+** 1**);**

**}**

**}**

void matrix**::**allocateMemory**()** **{**

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

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

al**[**i**]** **=** **new** real**[**m**]{** 0 **};**

**}**

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

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

au**[**i**]** **=** **new** real**[**m**]{** 0 **};**

**}**

di **=** **new** real**[**n**]{** 0 **};**

b **=** **new** real**[**n**]{** 0 **};**

x **=** **new** real**[**n**]{** 0 **};**

y **=** **new** real**[**n**]{** 0 **};**

**}**

void matrix**::**gauss**()**

**{**

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

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

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

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

**if** **(**j **+** m **-** i **<** 0 **||** i **+** m **-** j **<** 0**)**

matrix**[**i**][**j**]** **=** 0.0**;**

**else** **if** **(**j **>** i**)** matrix**[**i**][**j**]** **=** au**[**j**][**i **+** m **-** j**];**

**else** matrix**[**i**][**j**]** **=** al**[**i**][**j **+** m **-** i**];**

matrix**[**i**][**i**]** **=** di**[**i**];**

**}**

**}**

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

int leadIndex **=** i**;**

real lead **=** matrix**[**i**][**i**];**

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

**if** **(**abs**(**matrix**[**k**][**i**])** **>** abs**(**lead**))**

**{**

lead **=** matrix**[**k**][**i**];**

leadIndex **=** k**;**

**}**

std**::**swap**(**matrix**[**i**],** matrix**[**leadIndex**]);**

std**::**swap**(**b**[**i**],** b**[**leadIndex**]);**

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

real m\_i **=** matrix**[**k**][**i**]** **/** matrix**[**i**][**i**];**

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

matrix**[**k**][**j**]** **-=** m\_i **\*** matrix**[**i**][**j**];**

b**[**k**]** **-=** m\_i **\*** b**[**i**];**

**}**

**}**

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

real sum **=** 0**;**

**for** **(**int j **=** n **-** 1**;** j **>** i**;** j**--)**

sum **+=** matrix**[**i**][**j**]** **\*** x**[**j**];**

x**[**i**]** **=** **(**b**[**i**]** **-** sum**)** **/** matrix**[**i**][**i**];**

//printf("\n%.12f", x[i]);

**}**

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

printf**(**"\n" realout**,** x**[**i**]);**

**}**

int main**()** **{**

FILE**\*** inmat**,** **\*** invec**;**

matrix matrix**{};**

int num**;**

num **=** 1**;**

**switch** **(**num**)** **{**

**case** 1**:**

printf**(**"\n\n\nN4:\n"**);**

**for** **(**matrix**.**k **=** 0**;** matrix**.**k **<** 8**;** matrix**.**k**++)**

**{**

fopen\_s**(&**inmat**,** "inmat.txt"**,** "r"**);**

fopen\_s**(&**invec**,** "invec.txt"**,** "r"**);**

matrix**.**input**(**inmat**,** invec**);**

matrix**.**calcLU**();**

matrix**.**calcY**(**matrix**.**al**,** matrix**.**b**,** matrix**.**y**,** matrix**.**n**);**

matrix**.**calcX**(**matrix**.**au**,** matrix**.**y**,** matrix**.**x**,** matrix**.**n**);**

**}**

**break;**

**case** 2**:**

printf**(**"\n\n\nN5:\n"**);**

**for** **(**matrix**.**n **=** 2**;** matrix**.**n **<** 9**;** matrix**.**n**++)** **{**

matrix**.**m **=** matrix**.**n **-** 1**;**

matrix**.**allocateMemory**();**

matrix**.**createGilbert**(**matrix**.**al**,** matrix**.**au**,** matrix**.**di**,** matrix**.**n**,** matrix**.**m**);**

matrix**.**calcLU**();**

real bbb**[**12**][**12**]** **=** **{** **{**2.0**,**7.0 **/** 6**},**

**{** 3.0**,**23.0 **/** 12**,**43.0 **/** 30 **},**

**{** 4.0**,** 163.0 **/** 60**,**21.0 **/** 10**,**241.0 **/** 140 **},**

**{** 5.0**,** 71.0 **/** 20**,** 197.0 **/** 70**,** 657.0 **/** 280**,** 1271.0 **/** 630 **},**

**{** 6.0**,**617.0 **/** 140**,**499.0 **/** 140**,**2531.0 **/** 840**,**1649.0 **/** 630**,**12847.0 **/** 5544 **},**

**{** 7.0**,**1479.0 **/** 280**,**5471.0 **/** 1260**,**3119.0 **/** 840**,**22549.0 **/** 6930**,**16081.0 **/** 5544**,**157309.0 **/** 60060 **},**

**{** 8.0**,**15551.0 **/** 2520**,**6479.0 **/** 1260**,**41029.0 **/** 9240**,**27169.0 **/** 6930**,**253405.0 **/** 72072**,**191629.0 **/** 60060**,**30089.0 **/** 10296 **},**

**{** 9.0**,**17819.0 **/** 2520**,**82609.0 **/** 13860**,**47959.0 **/** 9240**,**415567.0 **/** 90090**,**299737.0 **/** 72072**,**45533.0 **/** 12012**,**71761.0 **/** 20592**,**988277.0 **/** 306306 **},**

**{** 10.0**,**221209.0 **/** 27720**,**94159.0 **/** 13860**,**715867.0 **/** 120120**,**479917.0 **/** 90090**,**347785.0 **/** 72072**,**106081.0 **/** 24024**,**1425857.0 **/** 350064**,**386149.0 **/** 102102**,**18266753.0 **/** 5173168 **},**

**{** 11.0**,**246619.0 **/** 27720**,**1376527.0 **/** 180180**,**810247.0 **/** 120120**,**545983.0 **/** 90090**,**794669.0 **/** 144144**,**2067641.0 **/** 408408**,**546595.0 **/** 116688**,**8459953.0 **/** 1939938**,**105559977.0 **/** 25865840**,**89301449.0 **/** 23279256 **},**

**{** 12.0**,**3538687.0 **/** 360360**,**1530967.0 **/** 180180**,**906343.0 **/** 120120**,**1227101.0 **/** 180180**,**15239101.0 **/** 2450448**,**779971.0 **/** 136136**,**11785561.0 **/** 2217072**,**48119579.0 **/** 9699690**,**120340457.0 **/** 25865840**,**101999225.0 **/** 23279256**,**2015839999.0 **/** 486748080 **}**

**};**

matrix**.**b **=** bbb**[**matrix**.**n **-** 2**];**

matrix**.**calcY**(**matrix**.**al**,** matrix**.**b**,** matrix**.**y**,** matrix**.**n**);**

matrix**.**calcX**(**matrix**.**au**,** matrix**.**y**,** matrix**.**x**,** matrix**.**n**);**

**}**

**break;**

**case** 3**:**

printf**(**"\n\n\nN7:\n"**);**

**for** **(**matrix**.**k **=** 0**;** matrix**.**k **<** 15**;** matrix**.**k**++)**

**{**

fopen\_s**(&**inmat**,** "inmat.txt"**,** "r"**);**

fopen\_s**(&**invec**,** "invec.txt"**,** "r"**);**

matrix**.**input**(**inmat**,** invec**);**

matrix**.**gauss**();**

**}**

**break;**

**default:**

**return** 0**;**

**}**

**}**