**Часть 2. Решение СЛАУ прямыми методами**

**Формулировка задачи и ее формализация**

Найти решение СЛАУ методом LU-разложения, Проверить вычислительную ошибку (найти вектор невязки) для матриц с разными числами обусловленности. Для исследования выбираем квадратную матрицу при Решаем уравнение вида

**Алгоритм метода и условие его применимости**

А = LU, где

Так, задача сводится к тому, чтобы решить уравнение вида LUx=b. Эта система может быть решена в два шага. На первом шаге решается система Ly =b (непосредственно прямой подстановкой). На втором шаге решается система Ux=y (непосредственно обратной подстановкой).

Алгоритм LU-разложения:

И так далее. Получим формулы:

Ly = b (прямая подстановка):

Ux = y (обратная подстановка):

;

Условия применимости:

Все главные миноры матрицы А отличны от нуля.

Для нахождения главных миноров вычеркиваем (n – j) столбцы и (n – j) строки из матрицы А:

**Предварительный анализ задачи и условий применимости метода**

Поскольку задачей стоит исследовать матрицы с разными числами обусловленности, рассмотрим работу метода на примерах двух матриц – хорошо и плохо обусловленной.

Хорошо обусловленная матрица:

*;*

где:

, где W – вектор размера , состоящий из случайных чисел;

*–* треугольная или диагональная матрица размера , состоящая из случайных чисел.

Плохо обусловленная матрица:

В качестве плохо обусловленной матрицы выбираем матрицу Гильберта:

*;*

Условия применимости:

**Тестовый пример с детальными расчетами для задачи малой размерности**

1)

2)

3)

4)

5) ;

;

,

**Модульная структура программы**

Программа состоит из 10 модулей:

1. “bad\_generate.m“ – создает плохо обусловленную матрицу Гильберта, считает число обусловленности, записывает данные в файл.

2. “good\_random.m” - создает хорошо обусловленную матрицу, считает число обусловленности, записывает данные в файл.

3. void readB (double \*\*b, char \*string1, char \*string2) – считывает вектор b в массив b из файла с названием string1, записывает вектор b в файл с названием string2.

4. void readA (double \*\*\*matr, char \*string) – считывает матрицу в двумерный массив matr из файла с названием string.

5. void LU (double \*\*\*L, double \*\*\*U, double \*\*\*A) – раскладывает матрицу А на матрицы L и U, такие, что LU=A.

6. void solve (double \*\*\*L, double \*\*\*U, double \*\*b, double \*\*\*A, char \*string) – решает СЛАУ, записывает результаты в файл с именем string.

7. void nevyazka (double \*\*\*A, double \*\*x, double \*\*b) – считает вектор невязки, выводит на экран.

8. void errors (double \*\*\*A, double \*\*b, double \*\*\*L, double \*\*\*U, char \*string) – вносит возмущения в матрицу А, считает новое разложение, записывает результаты в файл с названием string.

9. “workspace.m“ – вызывает функцию вычисления коэффициентов для каждой из матриц.

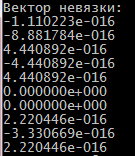
10. “errors.m” – считывает данные из матрицы, считает коэффициенты для возмущений в b и в A.

**Численный анализ решения задачи**

Для хорошо обусловленной матрицы:

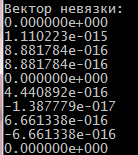
Cond(A) = 11,9671;

Внесение возмущений в b.



.

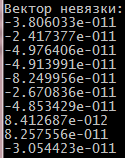
Внесение возмущений в A.

.

Для плохо обусловленной матрицы:

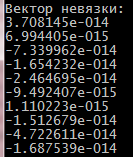
Cond(A) = 2.6791e+13;

Внесение возмущений в b.



.

Внесение возмущений в A.



.

Пример подробных расчетов для задачи малой размерности см. в Приложении.

**Краткие выводы**

Для матрицы любой обусловленности коэффициент при внесении возмущений в вектор b был меньше, чем коэффициент при внесении возмущений в матрицу А, а так же был небольшим вне зависимости от числа обусловленности матрицы. Коэффициент при внесении возмущений в матрицу А был маленьким для матрицы с маленьким числом обусловленности, и большим для матрицы с большим числом обусловленности.

Коэффициенты всегда меньше числа обусловленности матрицы.

**Приложение**

n = 10;

D = eye(n);

E = eye(n);

D(n, n) = 2; %задали диагональную

W = rand(n, 1); %зарандомили вектор W

nr = norm(W, 2);

H = E-2\*W\*W'\*(nr^2);

A = H\*D\*H'

cond = norm(A, 1)\*norm(A^(-1))

b = randn(n, 1);

Код программы 1. Задание хорошо обусловленной матрицы и вектора b.

n=10;

for i = 1:1:n

for j = 1:1:n

C(i, j) = 1/(i+j-1);

end

end

cond1 = norm(C, 1)\*norm(C^(-1))

determinant = det(C)

b1 = randn(n, 1);

Код программы 2. Задание плохо обусловленной матрицы и вектора b.

for(k = 1; k < size; k++)

{

for(i = k-1; i < size; i++)

for(j = i; j < size; j++)

(\*L)[j][i]=(\*U)[j][i]/(\*U)[i][i];

for(i = k; i < size; i++)

for(j = k-1; j < size; j++)

(\*U)[i][j]=(\*U)[i][j]-(\*L)[i][k-1]\*(\*U)[k-1][j];

}

Код программы 3. Код LU-разложения матрицы.

y[0] = (\*b)[0];

for (i=1; i<size; i++)

{

sum =0;

for (j=0; j<=i-1; j++)

sum+=(\*L)[i][j]\*y[j];

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

}

x[size-1] = y[size-1]/(\*U)[size-1][size-1];

for (i=size-2; i>=0; i--)

{

sum = 0;

for (j=size-1; j>=i+1; j--)

sum+=(\*U)[i][j]\*x[j];

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

}

Код программы 4. Код обратной и прямой подстановки для нахождения векторов х и у.

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

{

C[i] = 0;

for(k = 0; k < size; k++)

{

C[i] += (\*A)[i][k] \* (\*x)[k];

nev[i] = C[i] - (\*b)[i];

}

}

Код программы 5. Нахождение вектора невязки.

double left=1.0, right=1.01;

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

{

newb[i] = (left + (((double) rand())/(RAND\_MAX+1)) \* (right - left))\*(\*b)[i];

for (j=0; j<size; j++)

{

newA[i][j] = (left + (((double) rand())/(RAND\_MAX+1)) \* (right-left))\*(\*A)[i][j];

}

}

Код программы 6. Внесение изменений в вектор b и в матрицу A не более одного процента.

%cчитаем коэффицент для возмущений в b

delx = xb-x;

delb = newb-b;

k1 = (norm(delx, 2)\*norm(b, 2))/(norm(x, 2)\*norm(delb, 2))

%cчитаем коэффициент для возмущений в А

delA = newA-A;

delx = xA-x;

k2 = (norm(delx, 2)\*norm(A, 2))/(norm(xA, 2)\*norm(delA, 2))

Код программы 7. Вычисление коэффициентов для возмущений в b и в A.

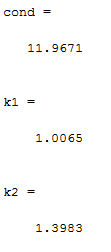


Рисунок 1. Результат вычисления коэффициентов для матрицы 10х10 с маленьким числом обусловленностей.

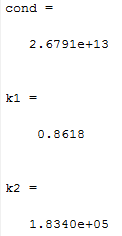


Рисунок 2. Результат вычисления коэффициентов для матрицы 10х10 с большим числом обусловленностей.

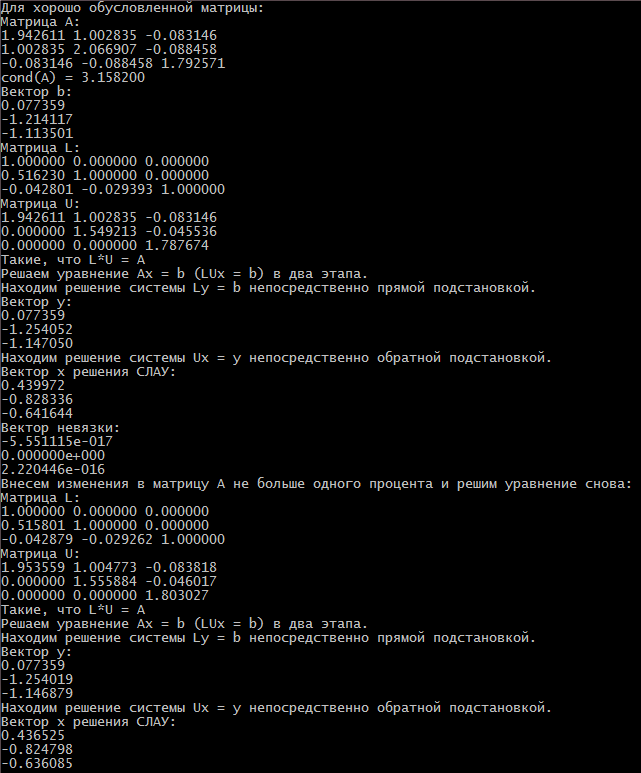


Рисунок 3. Результат работы программы для задачи малой размерности.

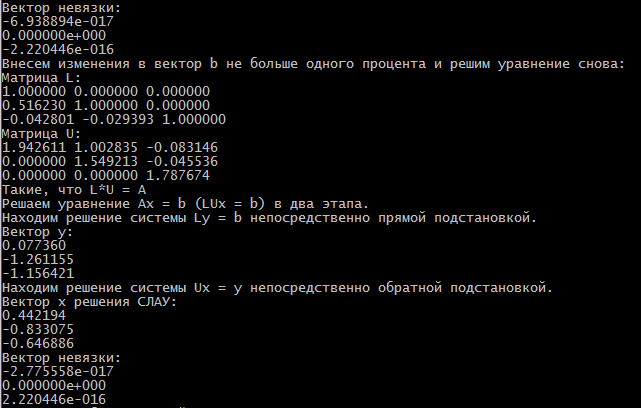


Рисунок 4. Результат работы программы для задачи малой размерности.

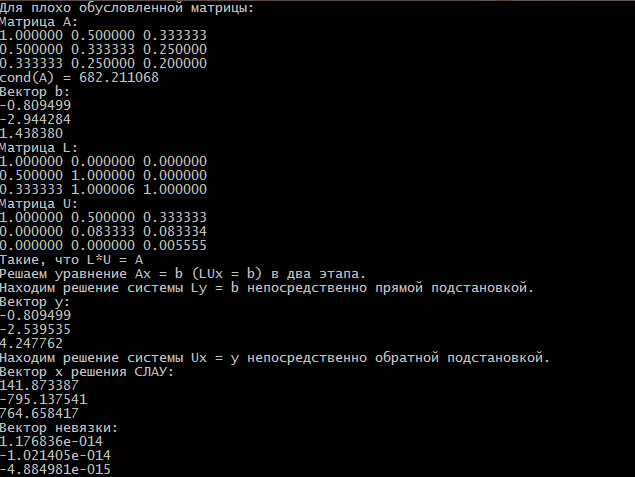


Рисунок 5. Результат работы программы для задачи малой размерности.

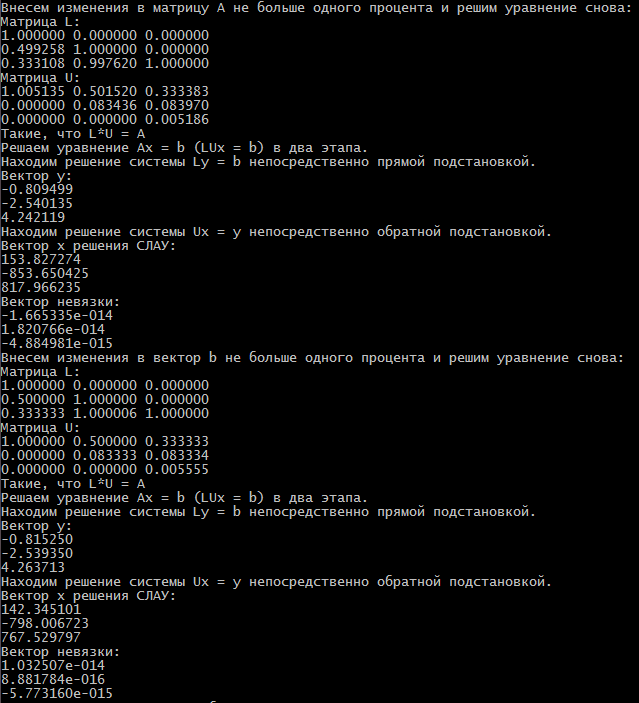


Рисунок 6. Результат работы программы для задачи малой размерности.

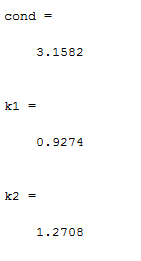


Рисунок 7. Результат вычисления коэффицентов для матрицы с маленьким числом обусловленностей для задачи малой размерности.

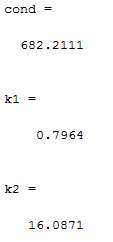


Рисунок 8. Результат вычисления коэффицентов для матрицы с большим числом обусловленностей для задачи малой размерности.