Лабораторная работа 2

«Решение систем на основе разложения симметричных матриц»

Выполнил

Студент 1 группы, 2 курса

Быченок Егор

1. **Постановка задачи**

Разработать программу численного решения СЛАУ на основе- разложения. Найти матрицы L и D после разложения, вектор приближённого решения x\*, а также рассчитать относительную погрешность вида для следующих матриц:

1. n = 11, m = 4, k = 0

Недиагональные элементы выбираются из чисел 0, -1, -2, -3, -4 произвольным образом,

Диагональные элементы:

, 2in

= +

1. n = 11, m = 4, k = 1

Недиагональные элементы выбираются из чисел 0, -1, -2, -3, -4 произвольным образом,

Диагональные элементы:

, 2in

= +

Для пунктов a), b) задать вектор решений

x = (m, m+1, …, m + n - 1),

а вектор свободных членов задать умножением:

b = Ax

1. **Входные данные**
2. n = 11, m = 4, k = 0

A =

[

26 0 0 0 0 0 0 0 0 0 0

-2 21 0 0 0 0 0 0 0 0 0

-4 -3 27 0 0 0 0 0 0 0 0

-3 0 -2 17 0 0 0 0 0 0 0

-4 -3 -3 -4 27 0 0 0 0 0 0

0 0 -2 -2 -2 18 0 0 0 0 0

-3 -2 -4 0 -2 -3 23 0 0 0 0

-4 -4 -2 0 -3 -4 -3 27 0 0 0

-1 0 -2 -1 -2 -2 0 -3 15 0 0

-4 -4 -4 -1 -2 -2 -3 -1 0 24 0

0 -3 -1 -4 -2 -1 -3 -3 -4 -3 24

]

x= ( 4 5 6 7 8 9 10 11 12 13 14 )

1. n = 11, m = 4, k = 1

A =

[

25.1 0 0 0 0 0 0 0 0 0 0

-2 21 0 0 0 0 0 0 0 0 0

-4 -3 27 0 0 0 0 0 0 0 0

-3 0 -2 17 0 0 0 0 0 0 0

-4 -3 -3 -4 27 0 0 0 0 0 0

0 0 -2 -2 -2 18 0 0 0 0 0

-3 -2 -4 0 -2 -3 23 0 0 0 0

-4 -4 -2 0 -3 -4 -3 27 0 0 0

-1 0 -2 -1 -2 -2 0 -3 15 0 0

-4 -4 -4 -1 -2 -2 -3 -1 0 24 0

0 -3 -1 -4 -2 -1 -3 -3 -4 -3 24

]

x = ( 4 5 6 7 8 9 10 11 12 13 14 )

1. **Листинг программы**

*//решение системы на основе LDLt-разложения*

**void** LinearEquationsSystem::solveWithLdLt() {  
 *//LdLt-разложение* **float** l;  
 **float**\* t = **new float**[n];  
 **for** (**int** step = 0; step < n-1; step++) {  
 **for** (**int** i = step + 1; i < n; i++) {  
 t[i] = A[i][step];

*//записываем l[i][step] сразу в A[i][step]*  
 A[i][step] /= A[step][step];

**for** (**int** j = step + 1; j <= i; j++) {  
 A[i][j] -= A[i][step] \* t[j];  
 }  
 }  
 }  
 printMatrix(13);  
 cout << endl;  
 **delete** [] t;  
  
 *//решаем Ly = b, где y = DL'x* **float** sum;  
 **float**\* yApprox = **new float**[n];

*//обратный ход для нижнетреугольной матрицы*  
 **for** (**int** i = 0; i < n; i++) {  
 sum = 0;  
 **for** (**int** j = 0; j < i; j++) {  
 sum += A[i][j] \* yApprox[j];  
 }  
 yApprox[i] = b[i] - sum;  
 }  
  
 *//находим решение DL'x = y  
 //обратный ход Гаусса* sum = 0;  
 **for** (**int** i = n-1; i >= 0; i--) {  
 sum = 0;  
 **for** (**int** j = i + 1; j < n; j++) {

*//sum += D[i][i] \* L'[i][j] \* xApprox[j]*  
 sum += A[i][i] \* A[j][i] \* xApprox[j]; }  
 xApprox[i] = (yApprox[i] - sum) / A[i][i];  
 }  
  
 **delete** [] yApprox;  
}

*//вычисляем относительную погрешность вида*

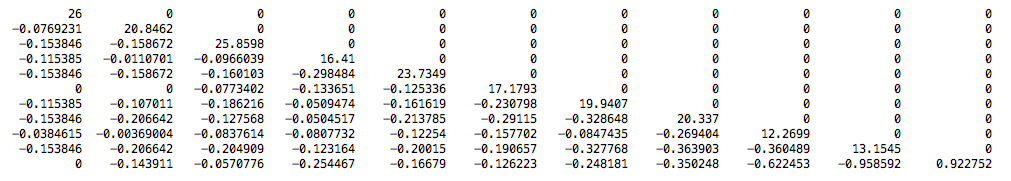
**float** calculateError() {  
 **float** error;  
 **float** normError = 0;  
 **float** xRealNorm = 0;

*//вычисляем*   
 **for** (**int** i = 0; i < n; i++) {  
 **if** (abs(xReal[i]) > xRealNorm) {  
 xRealNorm = abs(xReal[i]);  
 }  
 }

*//вычисляем*   
 **for** (**int** i = 0; i < n; i++) {  
 **if** (abs(xReal[i] - xApprox[i]) > normError) {  
 normError = abs(xReal[i] - xApprox[i]);  
 }  
 }

*//вычисляем относительную погрешность*  
 error = normError / xRealNorm;  
 **return** error;  
}

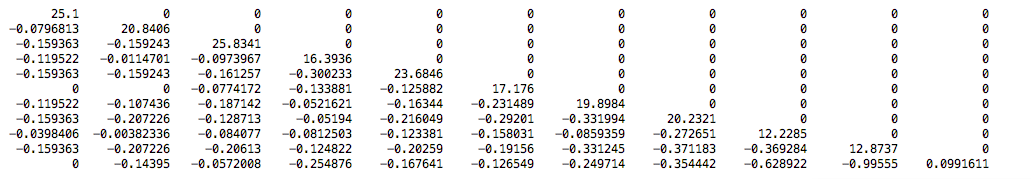
1. **Выходные данные**
2. Матрицы L (ниже главной диагонали) и D (главная диагональ) после разложения:



= ( 4.00002 5.00002 6.00002 7.00002 8.00002 9.00002 10 11 12 13 14 )

Error = 1.77111e-06

1. Матрица А после первого шага метода Гаусса



= ( 4.00103 5.00104 6.00104 7.00104 8.00104 9.00104 10.001 11.001 12.001 13.001 14.001 )

Error = 7.41822e-05

1. **Выводы**

Вычисление -разложения помогает сократить количество вычислений для нахождения решения СЛАУ используя свойства симметричекой матрицы

В пункте а) ошибка небольшая, это связано с тем, что матрица А задана с диагональным преобладанием.

В пункте b) ошибка выросла незначительно, значит устойчивость этого метода хорошая или приращение 0.1 не достаточно мало, чтобы судить об устойчивости метода.