Лабораторная работа 1

«метод Гаусса»

Выполнил

Студент 1 группы, 2 курса

Быченок Егор

1. **Постановка задачи**

Разработать программу численного решения СЛАУ методом Гаусса без выбора ведущего элемента и с выбором ведущего элемента. Найти преобразованную матрицу А после первого шага метода Гаусса, вектор приближённого решения x\*, а также рассчитать относительную погрешность вида для следующих матриц:

1. n = 13, m = 4, k = 0, (методом Гаусса без выбора ведущего элемента)

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

Диагональные элементы: , 2in

= +

1. n = 13, m = 4, k = 1, (методом Гаусса без выбора ведущего элемента)

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

Диагональные элементы: , 2in

= +

1. n = 17, m = 4, (методом Гаусса без выбора ведущего элемента и с выбора ведущего элемента)

Для заполнения матрицы A использовать случайные числа из диапазона от −100 до 100.

Для пунктов a), b), c) задать вектор решений

x = (m, m+1, …, m + n - 1),

а вектор свободных членов задать умножением:

b = Ax

1. **Входные данные**
2. n = 13, m = 4, k = 0

A =

[

29 -2 -4 -3 -3 0 -2 -4 -3 -3 -4 0 0

-2 30 -2 -2 -3 -2 -4 0 -2 -3 -4 -4 -2

0 -3 21 -4 -3 -1 0 -2 -1 -2 -2 0 -3

-4 -4 -4 25 -1 -2 -2 -3 -1 0 0 -3 -1

-4 -2 -1 -3 28 -3 -4 -3 0 -4 -1 0 -3

-2 -1 -4 -4 -1 30 -3 -2 -3 -3 -3 -3 -1

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

-2 0 0 -4 -1 -3 -3 21 -3 0 -2 -2 -1

-1 -2 -3 -4 -1 -2 -1 -3 26 -2 -3 -2 -2

-4 -4 -4 -1 0 0 0 -3 -2 27 -4 -3 -2

0 -2 -4 -3 0 -3 -3 0 -1 -2 20 -2 0

-2 -2 -2 -1 0 -2 -1 -1 0 -1 -3 17 -2

-1 0 0 -4 0 -1 -2 -3 0 -4 -4 -3 22

]

x= ( 4 5 6 7 8 9 10 11 12 13 14 15 16 )

1. n = 13, m = 4, k = 1

A =

[

28.1 -2 -4 -3 -3 0 -2 -4 -3 -3 -4 0 0

-2 30 -2 -2 -3 -2 -4 0 -2 -3 -4 -4 -2

0 -3 21 -4 -3 -1 0 -2 -1 -2 -2 0 -3

-4 -4 -4 25 -1 -2 -2 -3 -1 0 0 -3 -1

-4 -2 -1 -3 28 -3 -4 -3 0 -4 -1 0 -3

-2 -1 -4 -4 -1 30 -3 -2 -3 -3 -3 -3 -1

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

-2 0 0 -4 -1 -3 -3 21 -3 0 -2 -2 -1

-1 -2 -3 -4 -1 -2 -1 -3 26 -2 -3 -2 -2

-4 -4 -4 -1 0 0 0 -3 -2 27 -4 -3 -2

0 -2 -4 -3 0 -3 -3 0 -1 -2 20 -2 0

-2 -2 -2 -1 0 -2 -1 -1 0 -1 -3 17 -2

-1 0 0 -4 0 -1 -2 -3 0 -4 -4 -3 22

]

x = ( 4 5 6 7 8 9 10 11 12 13 14 15 16 )

1. n = 17, m = 4

A = [

-3 67 69 -4 83 10 82 -56 -22 -84 23 -8 -81 51 -92 29 81

-91 -99 9 -14 -49 -84 -55 -30 -57 10 32 -38 -28 33 66 -18 96

79 37 30 -10 95 10 -66 -68 -25 -13 74 21 58 -71 29 62 36

-86 82 18 14 21 -47 88 -50 -10 14 54 -30 -58 81 4 -44 -55

65 72 93 20 14 87 84 -98 52 -75 -56 -58 -36 2 9 49 -8

-72 47 -40 -85 -66 16 99 14 -55 -13 -53 17 52 59 -10 -64 -92

14 6 13 -39 -50 -4 3 -21 -42 27 -16 59 -55 56 -95 67 -68

68 -85 -10 91 98 -72 10 9 8 -71 -71 -36 -1 88 10 -31 30

-64 -6 -26 14 91 -98 65 -43 97 -90 -27 -98 89 -1 -80 -37 87

-98 -49 -67 70 4 -77 29 -38 -60 -35 65 -10 5 -5 70 -69 54

0 34 -73 -89 14 77 -22 90 -16 -88 -12 97 93 -58 40 -84 -40

-4 -75 1 11 -75 -12 80 34 94 -89 7 62 -90 -73 -61 -1 39

79 71 -39 -42 -3 -16 -49 -35 -1 -28 26 -46 -61 -57 -83 -93 92

-14 -62 -37 -64 69 95 -48 -25 83 18 72 43 -48 -52 40 -68 22

-15 12 58 42 43 -9 90 57 25 100 -46 -27 66 -12 32 -21 82

-99 -7 -86 47 1 68 -45 -14 41 -94 -2 80 -85 71 -68 91 -7

52 46 92 29 40 -76 -9 67 -11 80 73 88 82 50 -96 -56 90

]

x = ( 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 )

1. **Листинг программы**

*//решение системы без выбора ведущего элемента*

**void** solveWithoutSelectingMainElement() {  
 *//прямой ход* **float** l;  
 **for** (**int** step = 0; step < n-1; step++) {

*//выводим матрицу после первого шага*  
 **if** (step == 1) {  
 **this**->printMatrix(12);  
 cout << endl;  
 }

**for** (**int** i = step + 1; i < n; i++) {  
 l = A[i][step] / A[step][step]; *//вычисляем коэфициент*  
 b[i] -= l \* b[step];  
 **for** (**int** j = step; j < n; j++) {  
 A[i][j] -= l \* A[step][j]; *//исключаем j-тую переменную из i-того уравнения*  
 }  
 }  
 }  
  
 *//обратный ход* **float** sum;  
 **for** (**int** i = n-1; i >= 0; i--) {  
 sum = 0;  
 **for** (**int** j = i + 1; j < n; j++) {  
 sum += A[i][j] \* xApprox[j];  
 }  
 xApprox[i] = (b[i] - sum) / A[i][i]; *//находим приближенное значение*   
 }  
}

-----------------------------------------------------

*//решение системы с выбором ведущего элемента*

**void** solveWithoutSelectingMainElement() {  
 *//прямой ход* **float** l;

**int** mainRow;  
 **for** (**int** step = 0; step < n-1; step++) {

*//выводим матрицу и номер ведущего элемента в столбце после первого шага*  
 **if** (step == 1) {

cout << **"Main element row = "** << mainRow << endl;

**this**->printMatrix(12);  
 cout << endl;  
 }

*//выбираем ведущий элемент* mainRow = step;

*//находим максимальный по модулю элемент в столбце step*  
 **for** (**int** i = step + 1; i < n; i++) {  
 **if** (abs(A[i][step]) > abs(A[mainRow][step]))  
 mainRow = i;  
 }

*//если он равен нулю, то переходим к следующему шагу*   
 **if** (abs(A[mainRow][step]) == 0) {  
 **continue**;  
 }

*//меняем местами строки матрицы A*  
 **for** (**int** i = step; i <= n; i++) {  
 swap (A[mainRow][i], A[step][i]);  
 }

*//меняем элементы вектора b*  
 swap (b[mainRow], b[step]);

**for** (**int** i = step + 1; i < n; i++) {  
 l = A[i][step] / A[step][step]; *//вычисляем коэфициент*  
 b[i] -= l \* b[step];  
 **for** (**int** j = step; j < n; j++) {  
 A[i][j] -= l \* A[step][j]; *//исключаем j-тую переменную из i-того уравнения*  
 }  
 }  
 }  
  
 *//обратный ход* **float** sum;  
 **for** (**int** i = n-1; i >= 0; i--) {  
 sum = 0;  
 **for** (**int** j = i + 1; j < n; j++) {  
 sum += A[i][j] \* xApprox[j];  
 }  
 xApprox[i] = (b[i] - sum) / A[i][i]; *//находим приближенное значение*   
 }  
}

*//вычисляем относительную погрешность вида*

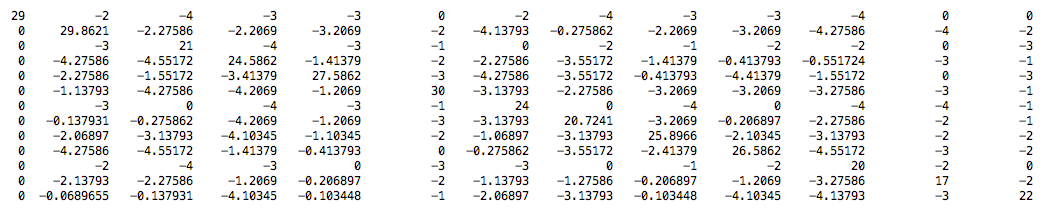
**float** calculateError() {  
 **float** error;  
 **float** normError = 0;  
 **float** xRealNorm = 0;

*//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;  
}

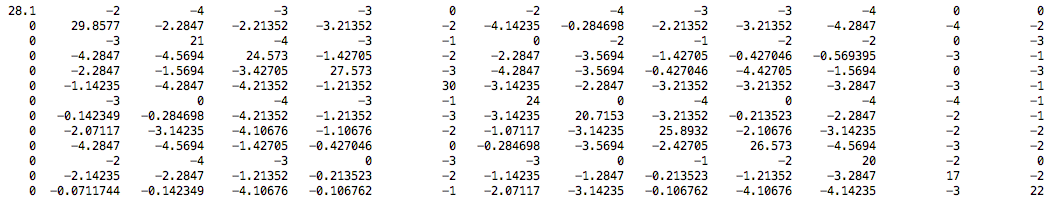
1. **Выходные данные**
2. Матрица А после первого шага метода Гаусса:



= ( 4.00011 5.00012 6.00012 7.00012 8.00012 9.00012 10.0001 11.0001 12.0001 13.0001 14.0001 15.0001 16.0001 )

Error = 7.39098e-06

1. Матрица А после первого шага метода Гаусса

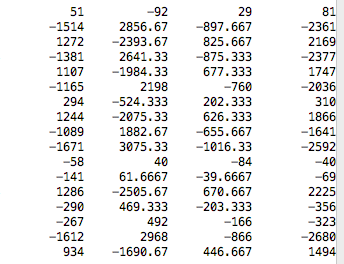
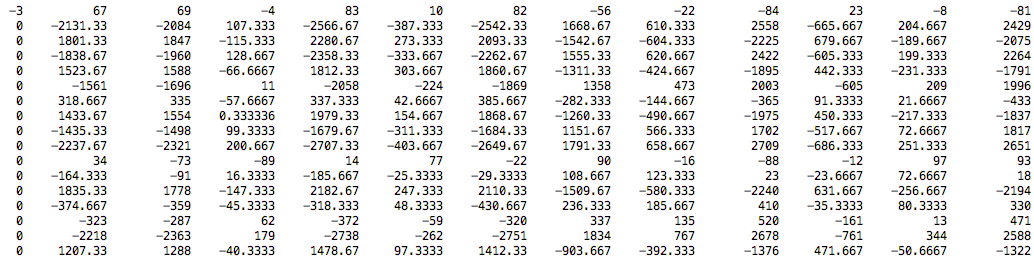


= ( 4.00015 5.00014 6.00014 7.00015 8.00015 9.00015 10.0001 11.0001 12.0001 13.0001 14.0001 15.0001 16.0001 )

Error = 9.23872e-06

с)

* Метод Гаусса без выбора ведущего элемента

Матрица А после первого шага метода Гаусса

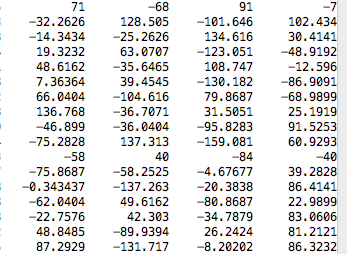
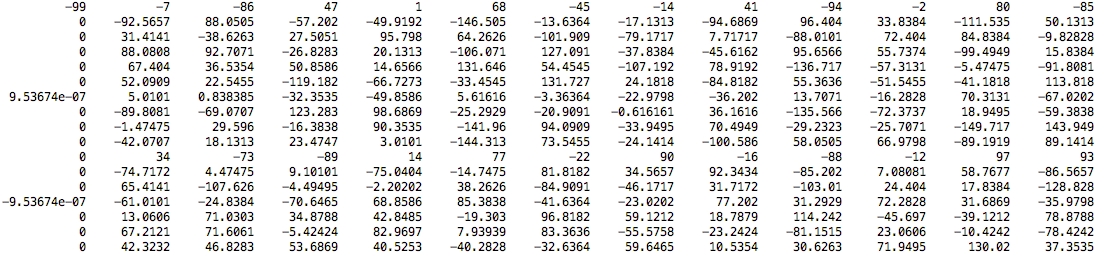
= ( 4.00002 4.99994 6.00002 7.00007 7.99996 9.00009 10 11 12 13 14.0001 14.9999 16 17 18 19 20 )

Error = 5.67436e-06

* Метод Гаусса с выбором ведущего элемента

Матрица А и номер ведущего элемента в столбце после первого шага метода Гаусса:

Номер ведущего элемента в столбце = 15



= ( 4.00001 5 5.99999 7 7.99999 8.99999 10 11 12 13 14 15 16 17 18 19 20 )

Error = 7.39098e-07

1. **Выводы**

В пункте а) ошибка небольшая, значит накопление погрешностей округления мало, а значит матрица хорошо обусловлена

В пункте b) ошибка выросла незначительно, значит или матрица А хорошо обусловлена, и приращение 0.1 не достаточно мало, чтобы изменить обусловленность матрицы

В пункте c) метод Гаусса с выбором ведущего элемента дал большую точность, что совпадает с теорией, т.к. при методе Гаусса без выбора ведущего элемента накопление погрешностей при округлении больше. Так же точность при методе Гаусса без выбора ведущего элемента высокая, значит матрица А хорошо обусловлен