



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ

«ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА

«ТЕОРЕТИЧЕСКАЯ ИНФОРМАТИКА И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ»

**Отчёт по лабораторной работе № 1**  
**«Решение СЛАУ с трёхдиагональной матрицей**  
**методом прогонки»**  
  
*по курсу*  
**«Численные методы»**

Студент группы ИУ9-61Б

Бакланова А.Д.

Преподаватель

Домрачева А.Б.

*Москва, 2020 г.*

## Цель работы

Целью данной работы является реализация программы для решения разреженных СЛАУ методом прогонки, в которых ненулевые элементов ограниченное количество.

## Задание

В ряде случаев практическая задача сводится к решению разреженных СЛАУ, в которых ненулевые элементов ограниченное количество.

Как правило, речь идет о ленточных матрицах, где ненулевые элементы находятся на главной диагонали и смежных с ней.

Ширина ленты  $S$  вычисляется по формуле (1).

$$S = 2l + 1 \quad (1)$$

**Дано:**  $A\bar{x} = \bar{d}, A \in \mathbb{R}^{n \times n}, \bar{d} \in \mathbb{R}^n$

**Найти:**  $\bar{x} \in \mathbb{R}^n$ ?

$$\begin{pmatrix} a_1 & b_1 & 0 & \dots & 0 \\ c_1 & a_2 & b_2 & \dots & 0 \\ 0 & \dots & \dots & \dots & 0 \\ 0 & \dots & \dots & b_{n-1} & \\ 0 & \dots & c_{n-1} & a_n & \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \dots \\ d_{n-1} \\ d_n \end{pmatrix}$$

Тогда имеем систему:

$$\begin{cases} a_1x_1 + b_1x_2 = d_1 \\ c_1x_1 + a_2x_2 + b_2x_3 = d_2 \\ \dots \\ c_{n-1}x_{n-1} + a_nx_n = d_n \end{cases}$$

Далее выражаем  $x_1$  через  $x_2$  и так далее, пока не останется строчка с  $x_n$

$$x_1 = \frac{d_1}{a_1} - \frac{b_1}{a_1} \cdot x_2$$

Далее заменяем на вид  $x_1 = \alpha_1 \cdot x_2 + \beta_1$ , где  $\beta_1 = \frac{d_1}{a_1}$ , а  $\alpha_1 = -\frac{b_1}{a_1}$

И так далее индуктивно  $x_i = \alpha_i \cdot x_{i+1} + \beta_i$

Итого, имеется система:

$$\begin{cases} \alpha_i = -\frac{b_i}{\alpha_{i-1} \cdot c_{i-1} + a_i}, i = \overline{2, \dots, n}, & \alpha_i = -\frac{b_1}{a_1}, a_1 \neq 0, \quad \beta_1 = \frac{d_1}{a_1} \\ \beta_i = \frac{d_i - c_{i-1} \cdot \beta_{i-1}}{\alpha_{i-1} \cdot c_{i-1} + a_i} \end{cases}$$

$$x_i = \alpha_i \cdot x_{i+1} + \beta_i$$

$$i = \overline{n-1, \dots, 1}, \quad x_n = \beta_n$$

Решение имеется только при выполнении трёх условий диагонального преобладания:

$$1) \quad |a_i| \geq |c_{i-1} + b_i|$$

$$2) \quad \frac{|b_i|}{|a_i|} \leq 1$$

$$3) \frac{|c_{i-1}|}{|b_i|} \leq 1$$

## Реализация:

```

9  #include <cmath>
10 #include <iostream>
11
12 int main(int argc, const char * argv[]) {
13     int n = 0;
14
15     std::cout << "Enter size of matrix \n";
16     std::cin >> n;
17
18     float *diag_a = new float[n];
19     float *diag_b = new float[n-1];
20     float *diag_c = new float[n-1];
21     float *free_d = new float[n];
22
23     std::cout << "Enter main diagonal \n";
24     for (int i=0; i<n; i++) {
25         std::cin >> diag_a[i];
26     }
27
28     std::cout << "Enter upper diagonal \n";
29     for (int i=0; i<n-1; i++) {
30         std::cin >> diag_b[i];
31     }
32
33     std::cout << "Enter lower diagonal \n";
34     for (int i=0; i<n-1; i++) {
35         std::cin >> diag_c[i];
36     }
37
38     std::cout << "Enter free numbers d \n";
39     for (int i=0; i<n; i++) {
40         std::cin >> free_d[i];
41     }
42
43     float *alpha = new float[n-1];
44     float *beta = new float[n-1];
45     float *x = new float[n];
46
47     //-----
48     for (int i=1; i<n-1; i++) {
49         if (std::abs(diag_a[i]) < std::abs(diag_c[i-1] + diag_b[i])) {
50             std::cout << "Input is incorrect 1 stat ";
51         }
52         if (std::abs(diag_b[i])/(std::abs(diag_a[i])) > 1) {
53             std::cout << "Input is incorrect 2 stat";
54         }

```

```

55     if (std::abs(diag_c[i-1]) / (std::abs(diag_b[i])) > 1) {
56         std::cout << "Input is incorrect 3 stat";
57     }
58 }
59
60 if (diag_a[0] != 0) {
61     alpha[0] = - diag_b[0]/diag_a[0];
62 } else {
63     std::cout << "a[0] = 0 error";
64     return 1;
65 }
66 beta[0] = free_d[0] / diag_a[0];
67
68 for (int i=1; i<n-1; i++) {
69     alpha[i] = -(diag_b[i])/(alpha[i-1]*diag_c[i-1] + diag_a[i]);
70     beta[i] = (free_d[i]-diag_c[i-1]*beta[i-1])/(alpha[i-1]*diag_c[i-1] + diag_a[i]);
71     //std::cout << "alpha[" << i <<"]= " << alpha[i];
72     //std::cout << "beta[" << i <<"]= " << beta[i];
73 }
74 beta[n-1] = (free_d[n-1]-diag_c[n-2]*beta[n-2])/(alpha[n-2]*diag_c[n-2] + diag_a[n-1]);
75 //std::cout << "beta[" << n-1 <<"]= " << beta[n-1];
76 x[n-1] = beta[n-1];
77
78 for (int i=n-2; i>=0; i--) {
79     x[i] = alpha[i]*x[i+1] + beta[i];
80 }
81
82 for (int i=0; i<n; i++) {
83     std::cout << "x[" << i << "]= " << x[i] << std::endl;
84 }
85
86 return 0;
87 }
88

```

## Тестирование:

$$\begin{pmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \\ 6 \\ 5 \end{pmatrix}$$

```
Enter size of matrix
4
Enter main diagonal
4 4 4 4
Enter upper diagonal
1 1 1
Enter lower diagonal
1 1 1
Enter free numbers d
5 6 6 5
x[0]= 1
x[1]= 1
x[2]= 1
x[3]= 1
Program ended with exit code: 0|
```

### **Вывод:**

Программа реализована и правильно считает разреженные СЛАУ методом прогонки, однако имеет место быть математическая погрешность.