

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ  
ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**  
**(ФГБОУ ВО «КубГУ»)**

Факультет математики и компьютерных наук  
Кафедра математических и компьютерных методов

**СЕМЕСТРОВАЯ РАБОТА № 4**  
**по дисциплине**  
**ОСНОВЫ КОМПЬЮТЕРНЫХ НАУК**  
**Тема: «РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ МЕТОДОМ**  
**ПРОСТЫХ ИТЕРАЦИЙ»**

Работу выполнил  
студент 1 курса  
группы 13А  
Акимов Вадим

Краснодар 2019

## Задача

Используя метод простых итераций, решить систему линейных уравнений согласно варианту с точностью 0,001. Систему привести к виду с диагональным преобладанием. Можно менять уравнения местами, складывать и вычитать уравнения, при необходимости предварительно умножив уравнения на константу.

$$\text{№ 1. } \begin{cases} 2,7x_1 + 3,3x_2 + 1,3x_3 = 2,1; \\ 3,5x_1 - 1,7x_2 + 2,8x_3 = 1,7; \\ 4,1x_1 + 5,8x_2 - 1,7x_3 = 0,8. \end{cases}$$

## Решение

### Краткое описание используемого метода

Постепенно выражая из начального приближения последующие, получать все более уточненные результаты

### Список идентификаторов

mat – матрица коэффициентов при неизвестных, double

b - матрица свободных членов, double

size – порядок матрицы коэффициентов, int

xn – ответ, double

xр – прошлый ответ, double

eps — погрешность, const double

beta – начальное приближение, double

### Описание алгоритма

Преобразование системы к нормальному виду, формирование начального приближения и начало итерационного метода

**Προπαμμα**

```
#include <iostream>
```

```
#include <iomanip>
```

```
#include <cmath>
```

```
using namespace std;
```

```
const double eps = 0.001;
```

```
void print_vector(double * x, int size) {  
    for (unsigned i = 0; i < size; ++i)  
        cout << x[i] << '\n';  
}
```

```
void print_mat(double ** mat, double * b, int size) {  
    cout << "\nYour matrix: \n";  
    for (unsigned i = 0; i < size; ++i) {  
        for (unsigned j = 0; j < size; ++j)  
            cout << mat[i][j] << ' ';  
        cout << " | " << b[i] << '\n';  
    }  
}
```

```
void read_mat(double ** mat, double * b, int size) {  
    cout << "Enter matrix: \n";  
    for (unsigned i = 0; i < size; i++) {  
        for (unsigned j = 0; j < size; j++)  
            cin >> mat[i][j];  
        cin >> b[i];  
    }  
}
```

```
void read_vec(double * vec, int size) {  
    for (unsigned i = 0; i < size; ++i)
```

```
    cin >> vec[i];  
}
```

```
double * copy_vector(double * xp, double * xn, int size) {  
    for (unsigned i = 0; i < size; i++)  
        xp[i] = xn[i];  
}
```

```
double norm(double * xp, double * xn, unsigned size) {  
    double n = 0;  
    for (unsigned i = 0; i < size; ++i)  
        n += (xp[i] - xn[i]) * (xp[i] - xn[i]);  
    return sqrt(n);  
}
```

```
double ** create_mat(unsigned size) {  
    double **mat = new double*[size];  
    for (unsigned i = 0; i < size; ++i)  
        mat[i] = new double[size];  
    return mat;  
}
```

```
double * create_vector(int size) {  
    double * vec = new double[size];  
    for (unsigned i = 0; i < size; i++)  
        vec[i] = 0;  
    return vec;  
}
```

```
void create_diagonal_dominance(double ** mat, double * b, double size) {  
    for (unsigned i = 0; i < size; i++)  
        for (unsigned k = i + 1; k < size; k++)  
            if (abs(mat[i][i]) < abs(mat[k][i])) {  
                for (unsigned j = 0; j <= size; j++) {
```

```

        double tmp = mat[i][j];
        mat[i][j] = mat[k][j];
        mat[k][j] = tmp;

    }

    double tmp = b[i];
    b[i] = b[k];
    b[k] = tmp;
}

}

void create_first_approximation(double ** mat, double * b, double * beta, int
size) {
    for (unsigned i = 0; i < size; ++i)
        beta[i] = b[i] / mat[i][i];
}

void new_x(double ** mat, double * b, double * beta, double * xn, double *
xp, int size) {
    copy_vector(xn, beta, size);
    for (unsigned i = 0; i < size; i++) {
        xn[i] += b[i] / mat[i][i];
        for (unsigned j = 0; j < size; j++)
            if (i != j)
                xn[i] += ( -mat[i][j] / mat[i][i] * xp[j] );
    }
}

int main() {
    cout << "Enter size matrix: ";
    int size;
    cin >> size;
    cout << '\n';

    double ** mat = create_mat(size);

```

```

double * beta = create_vector(size);
double * b    = create_vector(size);
double * xn   = create_vector(size);
double * xp   = create_vector(size);

read_mat(mat, b, size);
// print_mat(mat, b, size);

create_first_approximation(mat, b, beta, size);
create_diagonal_dominance(mat, b, size);
print_mat(mat, b, size);
new_x(mat, b, beta, xn, xp, size);
while (norm(xp, xn, size) > eps) {
    copy_vector(xp, xn, size);
    new_x(mat, b, beta, xn, xp, size);
}
cout << "\nAnswer vector: \n";
print_vector(xn, size);
return 0;
}

```

## Результат работы программы

```
Enter size matrix: 3
```

```
Enter matrix:
```

```
2.7 3.3 1.3 2.1
```

```
3.5 -1.7 2.8 1.7
```

```
4.1 5.8 -1.7 0.8
```

```
Your matrix:
```

```
4.1 5.8 -1.7 | 0.8
```

```
2.7 3.3 1.3 | 2.1
```

```
3.5 -1.7 2.8 | 1.7
```

```
Answer vector:
```

```
0.0619
```

```
0.3043
```

```
0.7146
```