Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы численного анализа

**ОТЧЁТ**

к лабораторной работе

на тему

Численное решение систем линейных уравнений методом простых итераций и методом Зейделя

Выполнил: студент группы 153503

Кахновский Евгений Сергеевич

Проверил: Анисимов Владимир Яковлевич

Минск 2022

**CОДЕРЖАНИЕ**

1. ЦЕЛЬ3

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ4

3. ЗАДАНИЕ8

4. АЛГОРИТМ РЕШЕНИЯ СЛАУ9

5. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ11

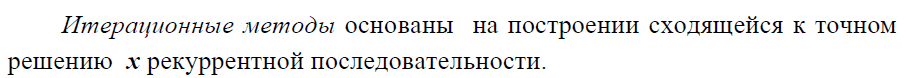
6. РЕЗУЛЬТАТЫ13

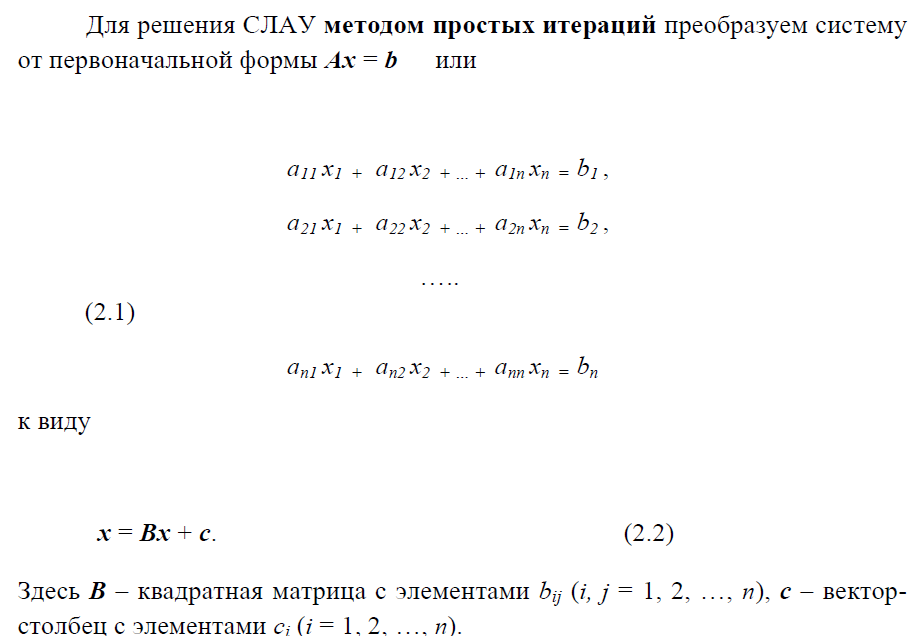
7. ВЫВОД17

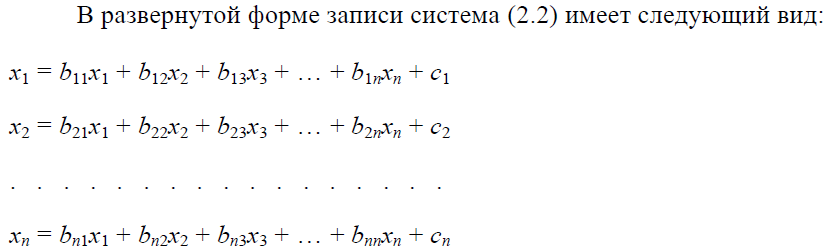
# **ЦЕЛЬ**

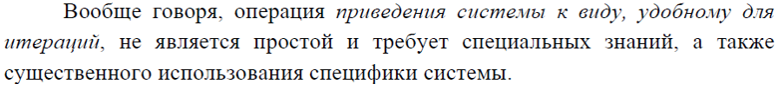
1. Изучить итерационные методы решения СЛАУ (метод простых итераций, метод Зейделя).
2. Составить алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ.
3. Составить программу решения СЛАУ по разработанному алгоритму.
4. Численно решить тестовые примеры и проверить правильность работы программы. Сравнить трудоемкость решения методом простых итерацией и методом Зейделя.

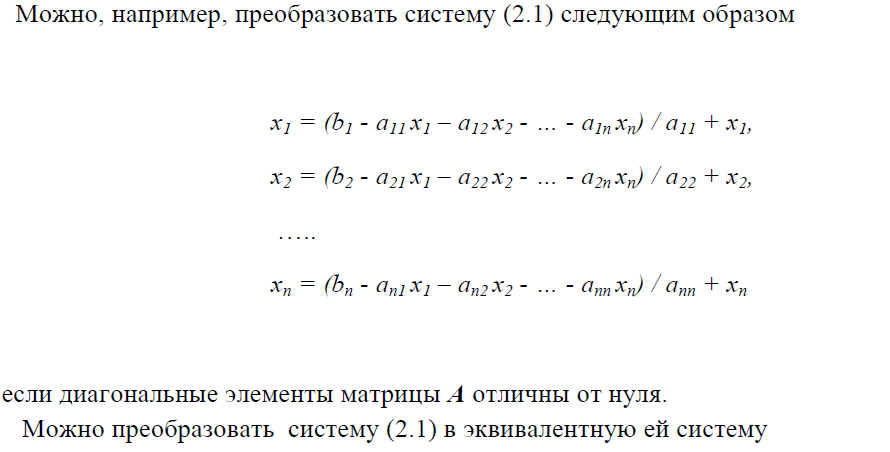
# **КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ**

****

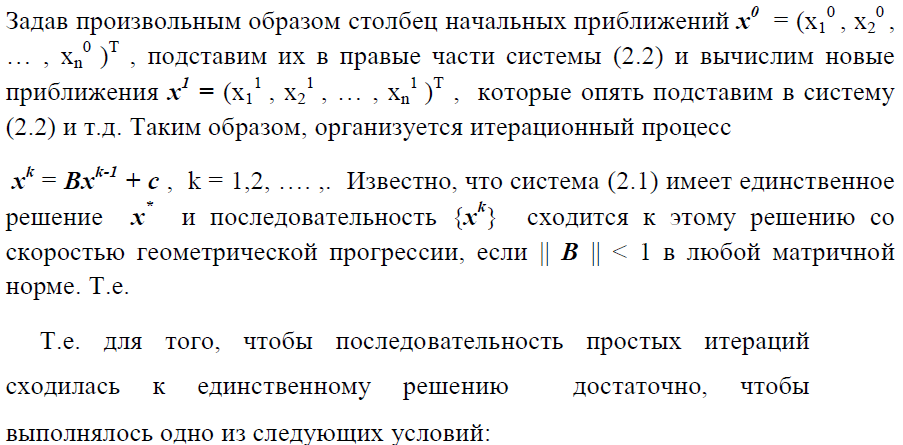
****

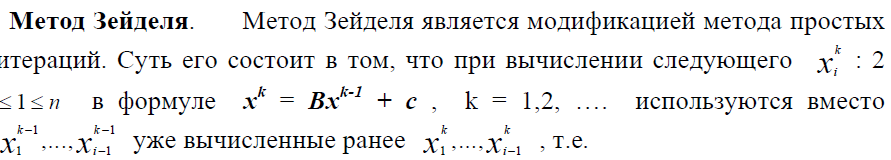
****

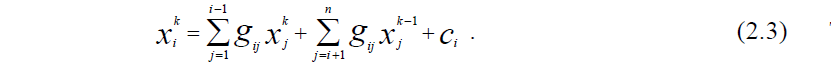
****

****

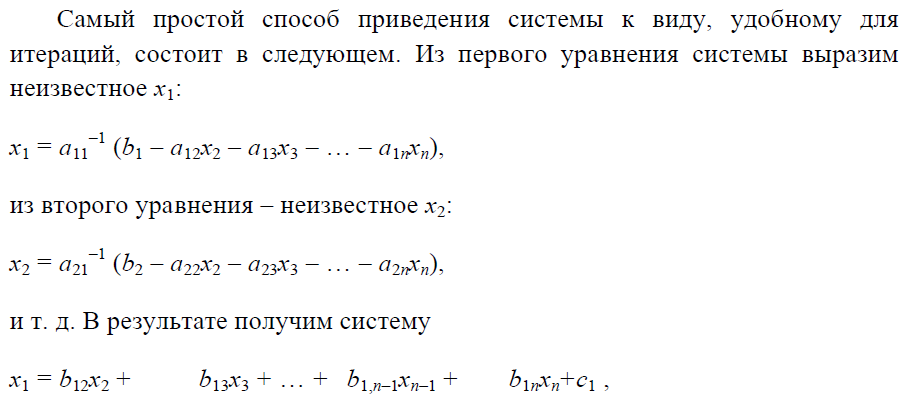
****

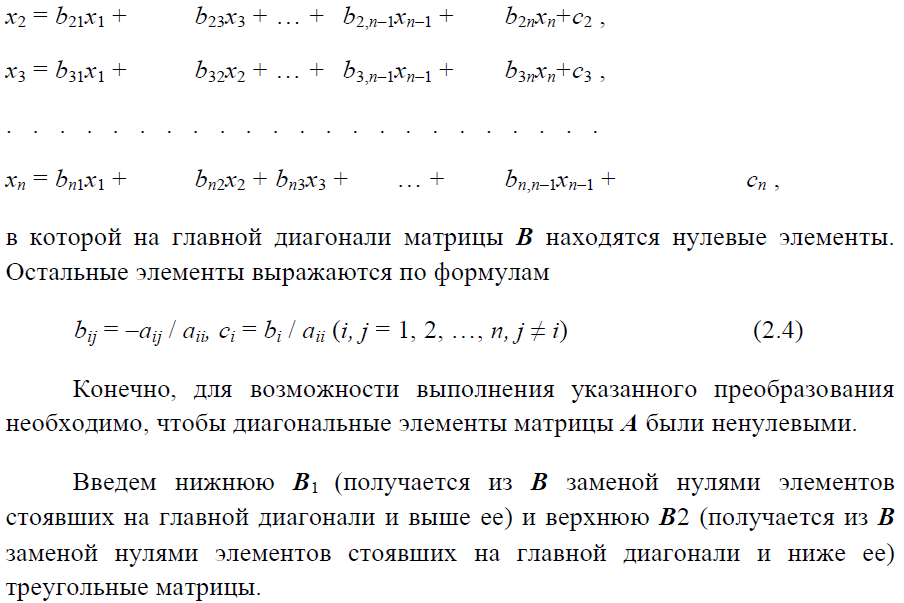
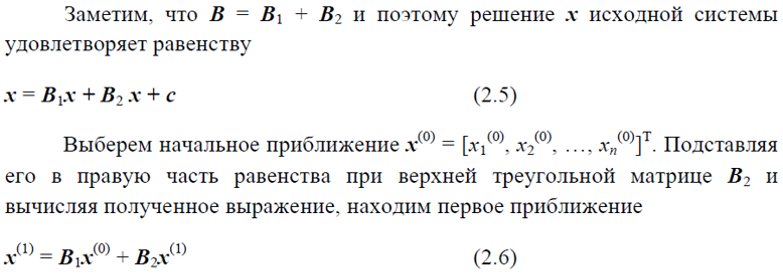
****

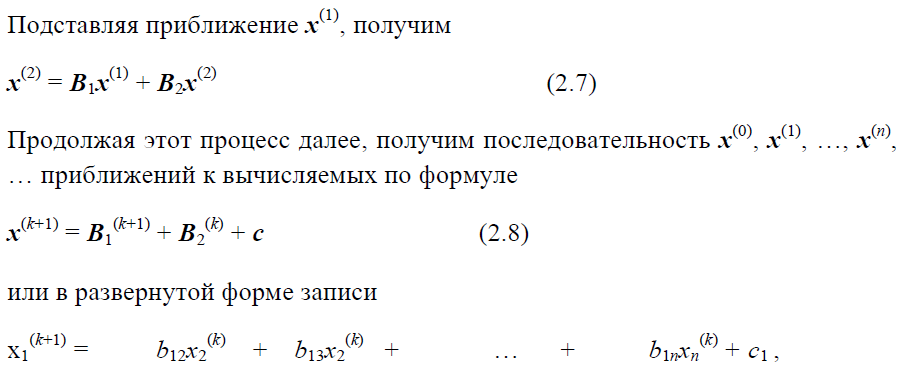
****

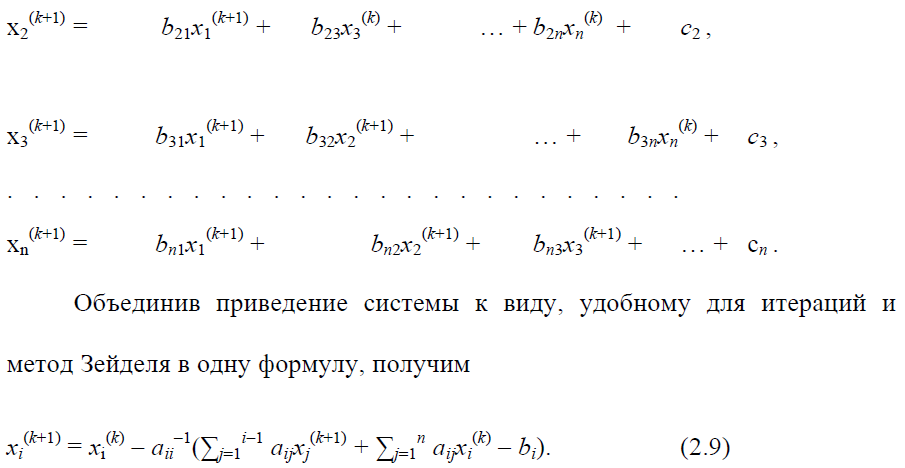
****

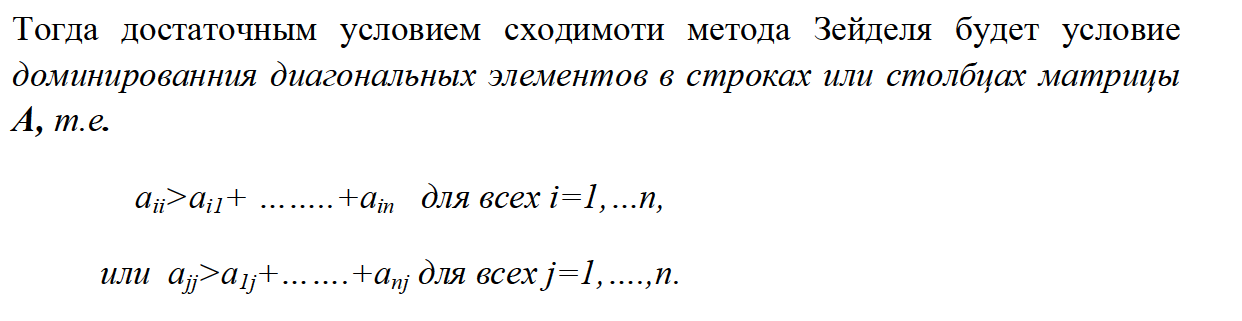
****

****

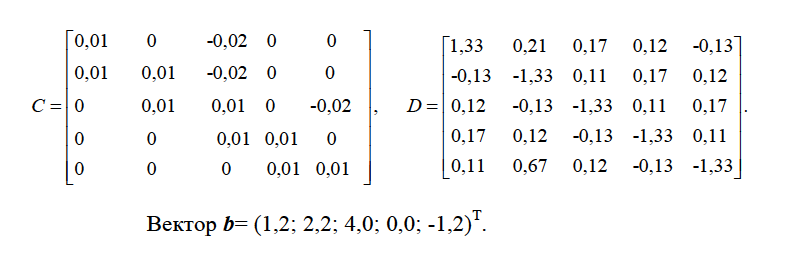
****

****

****



# **ЗАДАНИЕ**

 Вариант 11 (k = 11)

# **АЛГОРИТМ РЕШЕНИЯ СЛАУ**

**1. Метод простых итераций**



**2. Метод Зейделя**

# **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

1. **Метод простых итераций**

static public void Simple\_Iters\_Method()

{

data.iterations = 0;

int size = Matrix\_A.Count;

// Неизвестные на предыдущей итерации

List<double> previousVariable = new List<double> { 0, 0, 0, 0, 0 };

// Пока не будет достигнута точность

bool stop = false;

while (!stop)

{

data.iterations++;

// Неизвестные на текущей итерации

List<double> currentVariable = new List<double> { 0, 0, 0, 0, 0 };

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

{

// x\_i = b\_i

currentVariable[i] = Matrix\_A\_With\_Vector\_B[i][size];

// Вычитаем сумму по всем отличным от i-ой неизвестным

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

{

// С прошой итерации

if (j != i)

currentVariable[i] -= Matrix\_A\_With\_Vector\_B[i][j] \* previousVariable[j];

}

// x\_i /= b\_i

currentVariable[i] /= Matrix\_A\_With\_Vector\_B[i][i];

}

// Максимальная погрешность

double max\_error = 0.0;

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

{

double new\_max\_error = Math.Abs(currentVariable[i] - previousVariable[i]);

max\_error = new\_max\_error > max\_error ? new\_max\_error : max\_error;

}

// Дотигнута ли точность

if (max\_error < data.eps)

{

stop = true;

}

// Переход к следующей итерации

previousVariable = currentVariable;

}

results = previousVariable;

}

**2. Метод Зейделя**

#region Zeidel/SimpleIter\_Method

static public void Zeidel\_Method()

{

data.iterations = 0;

int size = Matrix\_A.Count;

// Неизвестные на предыдущей итерации

List<double> previousVariable = new List<double> {0,0,0,0,0};

// Пока не будет достигнута точность

bool stop = false;

while (!stop)

{

data.iterations++;

// Неизвестные на текущей итерации

List<double> currentVariable = new List<double> { 0, 0, 0, 0, 0 };

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

{

// x\_i = b\_i

currentVariable[i] = Matrix\_A\_With\_Vector\_B[i][size];

// Вычитаем сумму по всем отличным от i-ой неизвестным

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

{

// С этой итерации

if (j < i)

{

currentVariable[i] -= Matrix\_A\_With\_Vector\_B[i][j] \* currentVariable[j];

}

// С прошой итерации

if (j > i)

{

currentVariable[i] -= Matrix\_A\_With\_Vector\_B[i][j] \* previousVariable[j];

}

}

// x\_i /= b\_i

currentVariable[i] /= Matrix\_A\_With\_Vector\_B[i][i];

}

// Максимальная погрешность

double max\_error = 0.0;

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

{

double new\_max\_error = Math.Abs(currentVariable[i] - previousVariable[i]);

max\_error = new\_max\_error > max\_error ? new\_max\_error : max\_error;

}

// Дотигнута ли точность

if (max\_error < data.eps)

{

stop = true;

}

// Переход к следующей итерации

previousVariable = currentVariable;

}

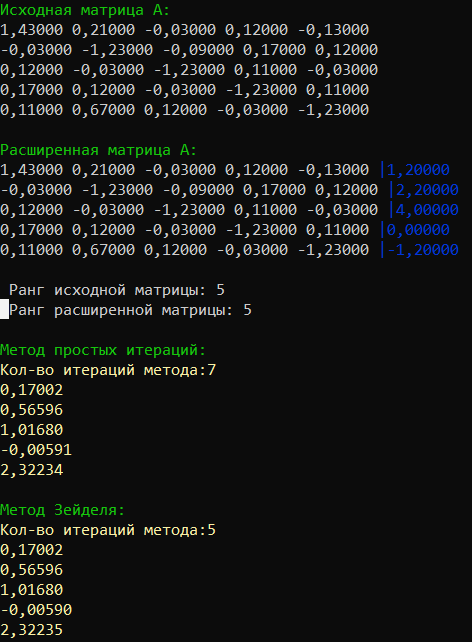
results = previousVariable;

}

# **ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ**

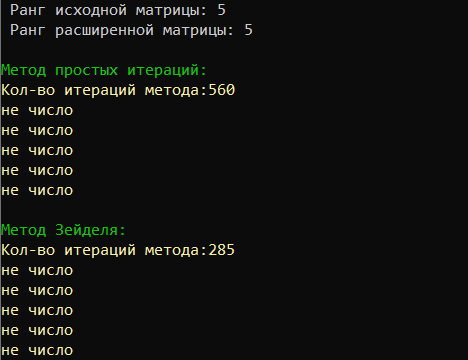
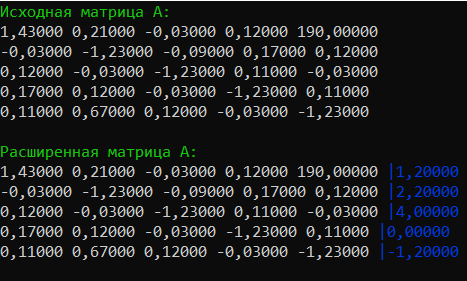
**Результаты задания по варинту.**

Матрица А, полученная в результате вычисления A=10C+D и вектор свободных членов (по условию) b=(1.2; 2.2; 4.0; 0.0; -1.2)



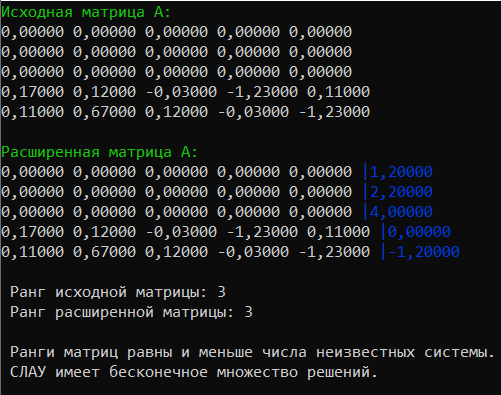
**Тестовый пример 1.**

Исходная матрица не обладает диагональным преобладанием.



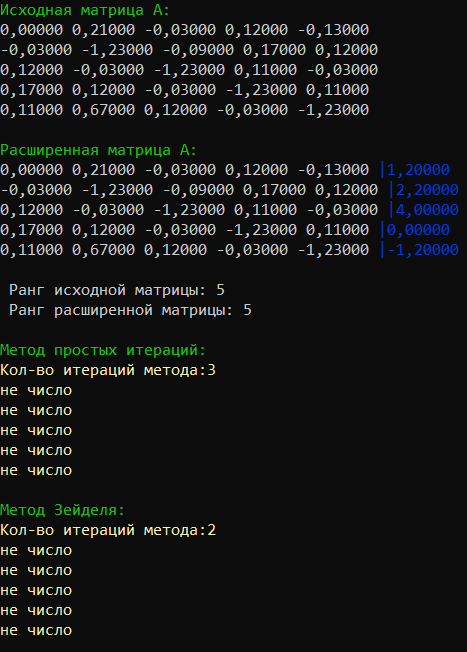
**Тестовый пример 2.**

Ранги матриц равны и меньше числа неизвестных, следовательно СЛАУ имеет бесконечно много решений.



**Тестовый пример 3.**

Сделаем элемент исходной матрица A1,1 = 0.



# **ВЫВОД**

В ходе лабораторной работы был применены итерационные методы решения СЛАУ в двух вариантах: метод простых итераций и метод Зейделя. Были разработаны алгоритмы решения СЛАУ указанными методами, составлена программа по разработанным алгоритмам, решены тестовые примеры.

На основании тестовых примеров можно сделать следующие выводы:

1. Метод простых итераций более ресурснозатратный (исходя из примеров, приведенных в моих результатах) из-за большего количества проводимых итераций.
2. Метод Зейделя является более быстрой модификацией метода простых итерация.
3. Оба метода позволяют получить решение с заданной точностью, причем корни в обоих методах могут отличаться в пределах заданной погрешности.
4. Если не выполняется необходимое и достаточное условие сходимости, то следует прибегнуть к преобразованиям, чтобы попытаться решить заданое СЛАУ.