МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №2

по дисциплине: Исследование операций и теория игр

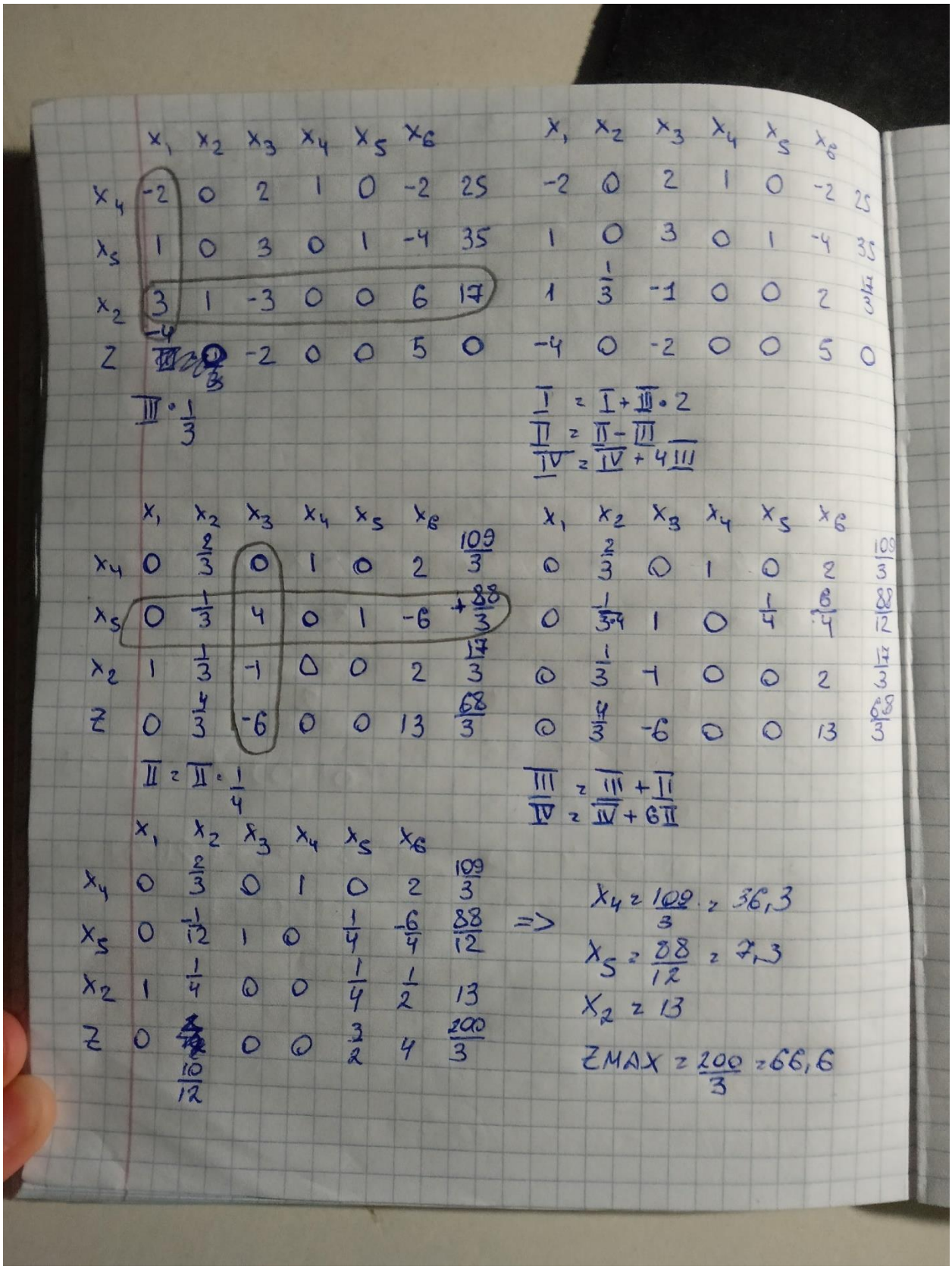
тема: «Симплекс-метод в чистом виде»

Выполнил: ст. группы Вт-211
Руденко Ксения Ильинична

Проверили:
Куртова Лилиана Николаевна

Белгород 2023 г.

Вариант 17



	x_1	x_2	x_3	x_4	x_5	x_6	
x_4	-2	0	2	1	0	-2	25
x_5	1	0	3	0	1	-4	35
x_2	3	1	-3	0	0	6	17
Z	-4	-2	0	0	5	0	

$III \cdot \frac{1}{3}$

	x_1	x_2	x_3	x_4	x_5	x_6	
	-2	0	2	1	0	-2	25
	1	0	3	0	1	-4	35
	1	$\frac{1}{3}$	-1	0	0	2	$\frac{17}{3}$
	-4	0	-2	0	0	5	0

$I = I + III \cdot 2$
$II = II - III$
$IV = IV + 4III$

	x_1	x_2	x_3	x_4	x_5	x_6	
x_4	0	$\frac{2}{3}$	0	1	0	2	$\frac{109}{3}$
x_5	0	$\frac{1}{3}$	4	0	1	-6	$\frac{88}{3}$
x_2	1	$\frac{1}{3}$	-1	0	0	2	$\frac{17}{3}$
Z	0	$\frac{4}{3}$	-6	0	0	13	$\frac{68}{3}$

$II = II \cdot \frac{1}{4}$

	x_1	x_2	x_3	x_4	x_5	x_6	
	0	$\frac{2}{3}$	0	1	0	2	$\frac{109}{3}$
	0	$\frac{1}{3} \cdot 4$	1	0	$\frac{1}{4}$	$\frac{6}{4}$	$\frac{88}{12}$
	1	$\frac{1}{3}$	-1	0	0	2	$\frac{17}{3}$
	0	$\frac{4}{3}$	-6	0	0	13	$\frac{68}{3}$

$III = III + II$
$IV = IV + 6II$

	x_1	x_2	x_3	x_4	x_5	x_6	
x_4	0	$\frac{2}{3}$	0	1	0	2	$\frac{109}{3}$
x_5	0	$\frac{-1}{12}$	1	0	$\frac{1}{4}$	$\frac{-6}{4}$	$\frac{88}{12}$
x_2	1	$\frac{1}{4}$	0	0	$\frac{1}{4}$	$\frac{1}{2}$	13
Z	0	$\frac{10}{12}$	0	0	$\frac{3}{2}$	4	$\frac{200}{3}$

$\Rightarrow$

$x_4 = \frac{109}{3} = 36,3$

$x_5 = \frac{88}{12} = 7,3$

$x_2 = 13$

$Z_{MAX} = \frac{200}{3} = 66,6$

Код программы:

```
#include <iostream>
#include <vector>
#include <windows.h>

using namespace std;

//вывод матрицы
void OutputMatrix(int numberOfRows, int numberOfCollumns,
vector<vector<double>> &a) {
    for (int i = 0; i < numberOfRows; i++) {
        for (int j = 0; j < numberOfCollumns; j++) {
            if (a[i][j] < 0) {
                cout << fixed << a[i][j] << '\t';
            } else {
                cout << " " << fixed << a[i][j] << '\t';
            }
        }
        cout << "\n";
    }
}

//проверка на базисы
void isEnoughtBasis(vector<vector<double>> &matrix, int numberOfRows, int
numberOfCols) {
    int numberOfBasis = 0;
    for (int i = 0; i < numberOfCols - 1; i++) {
        int counterOfZero = 0;
        for (int j = 0; j < numberOfRows; j++) {
            if (matrix[j][i] == 0) {
                counterOfZero++;
            }
        }
        if (counterOfZero == (numberOfRows - 1)) {
            numberOfBasis++;
        }
    }
    if (numberOfBasis == numberOfRows) {
        cout << "Базисов хватает";
    }
}

//функция возвращает значение true, если члены Z функции в симплекс таблице
больше нуля
bool isZfunction(vector<vector<double>> &matrix, int numberOfRows, int
numberOfCols) {
    for (int i = 0; i < numberOfCols - 1; i++) {
        if (matrix[numberOfRows][i] < 0) {
            return false;
        }
    }
    return true;
}

//Симплекс метод
void SymplecsMetod(vector<vector<double>> &matrix, int numberOfRows, int
numberOfCols) {
    //выбор разрешающего столбца
    int indexOfResolutioncolumn;
    double minForCollumn = matrix[numberOfRows][0];
    for (int i = 0; i < numberOfCols - 1; i++) {
        if (matrix[numberOfRows][i] > 0) {
            continue;
```

```
        } else {
            double min1 = matrix[numberOfRows][i];
            if (min1 < minForCollumn) {
                minForCollumn = min1;
                indexOfResolutioncolumn = i;
            }
        }
    }
    cout << "Индекс разрешающего столбца: " << indexOfResolutioncolumn <<
"\n";
    //выбор разрешающей строки
    int indexOfResolutionRow;
    double minimumRatio;
    for (int i = 0; i < numberOfRows - 1; i++) {
        if (matrix[i][indexOfResolutioncolumn] > 0) {
            minimumRatio = matrix[i][numberOfCols - 1] /
matrix[i][indexOfResolutioncolumn];
            indexOfResolutionRow = i;
        }
    }
    double minimumRatio1;
    for (int i = 0; i < numberOfRows; i++) {
        minimumRatio1 = matrix[i][numberOfCols - 1] /
matrix[i][indexOfResolutioncolumn];
        if (minimumRatio1 < minimumRatio && minimumRatio1 > 0) {
            minimumRatio = minimumRatio1;
            indexOfResolutionRow = i;
        }
    }
    cout << "Индекс разрешающей строки: " << indexOfResolutionRow << "\n";
    //выбрав разрешающий стоблец и разрешающую строку начинаем работать с
матрицей
    //преобразовали разрешающую строку
    double FirstCoef = matrix[indexOfResolutionRow][indexOfResolutioncolumn];
    for (int i = 0; i < numberOfCols; i++) {
        matrix[indexOfResolutionRow][i] /= FirstCoef;
    }
    //работаем с остальными строками
    double coef;
    for (int i = 0; i < numberOfRows + 1; i++) {
        coef = matrix[i][indexOfResolutioncolumn] /
matrix[indexOfResolutionRow][indexOfResolutioncolumn];
        for (int j = 0; j < numberOfCols; j++) {
            if (i != indexOfResolutionRow) {
                matrix[i][j] -= matrix[indexOfResolutionRow][j] * coef;
            }
        }
    }
}

//Вектор, который хранит ответ для x;
void indexes(vector<vector<double>> &matrix, int numberOfRows, int
numberOfCollumns,
             vector<double> &indexes) {
    for (int i = 0; i < numberOfCollumns - 1; i++) {
        int counterOfZero = 0;
        for (int j = 0; j < numberOfRows; j++) {
            if (matrix[j][i] == 0) {
                counterOfZero++;
            }
        }
        if (counterOfZero == (numberOfRows - 1)) {
            indexes.push_back(i);
        }
```

```
    }
}

void newIndex(vector<vector<double>> &matrix, int numberOfRows,
vector<double> &indexes) {
    vector<vector<double>> newMatrix(numberOfRows,
vector<double>(numberOfRows));
    for (int i = 0; i < numberOfRows; i++) {
        for (int j = 0; j < numberOfRows; j++) {
            newMatrix[i][j] = matrix[i][indexes[j]];
        }
    }
    vector<double> newIndexes;
    for (int i = 0; i < numberOfRows; i++) {
        for (int j = 0; j < numberOfRows; j++) {
            if (newMatrix[i][j] == 1) {
                newIndexes.push_back(indexes[j] + 1);
            }
        }
    }
    indexes = newIndexes;
}

int main() {
    SetConsoleOutputCP(CP_UTF8);
    int numberOfRows = 3, numberOfCols = 7;
    cout << "Функция Z:" << "\n";
    vector<double> zFunction(numberOfCols - 1);
    zFunction = {4, 0, 2, 0, 0, -5};
    //вывод функции Z
    cout << "Z = ";
    for (int i = 0; i < numberOfCols - 1; i++) {
        cout << zFunction[i] << "x[" << i + 1 << "] ";
    }
    //СЛАУ
    cout << "\n" << "Изначальная систему СЛАУ:" << "\n";
    vector<vector<double>> matrix(numberOfRows,
vector<double>(numberOfCols));
    matrix = {{-2, 0, 2,  1, 0, -2, 25},
              {1,  0, 3,  0, 1, -4, 35},
              {3,  1, -3, 0, 0, 6,  17}};
    vector<double> index;
    indexes(matrix, numberOfRows, numberOfCols, index);
    newIndex(matrix, numberOfRows, index);
    OutputMatrix(numberOfRows, numberOfCols, matrix);
    cout << "Проверка на кол-во базисов: ";
    isEnoughtBasis(matrix, numberOfRows, numberOfCols);
    cout << "\n" << "Изначальная таблица Симплекс метода:" << "\n";
    //первая таблица Симплекс метода
    vector<vector<double>> FirstSimplexMatrix(numberOfRows + 1,
vector<double>(numberOfCols));
    for (int i = 0; i < numberOfRows; i++) {
        for (int j = 0; j < numberOfCols; j++) {
            FirstSimplexMatrix[i][j] = matrix[i][j];
        }
    }
    for (int i = 0; i < numberOfCols; i++) {
        if (zFunction[i] != 0) {
            FirstSimplexMatrix[numberOfRows][i] = -zFunction[i];
        } else {
            FirstSimplexMatrix[numberOfRows][i] = zFunction[i];
        }
    }
    FirstSimplexMatrix[numberOfRows][numberOfCols - 1] = 0;
```

```
    OutputMatrix(numberOfRows + 1, numberOfCols, FirstSimplexMatrix);

    int k = 1;
    while (!isZfunction(FirstSimplexMatrix, numberOfRows, numberOfCols)) {
        SymplecsMetod(FirstSimplexMatrix, numberOfRows, numberOfCols);
        cout << "----------------------------------------------------------------
---" << "\n";
        cout << k << " таблица Симплекс метода" << "\n";
        OutputMatrix(numberOfRows + 1, numberOfCols, FirstSimplexMatrix);
        k++;
    }
    //вывод Zmax и иксов
    for (int i = 0; i < index.size(); i++) {
        cout << "x[" << index[i] << "]= " <<
FirstSimplexMatrix[i][numberOfCols - 1] << "\n";
    }
    cout << "Zmax= " << FirstSimplexMatrix[numberOfRows][numberOfCols - 1];
    return 0;
}
```

Результат: