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 №5

по дисциплине: исследование операций и теория игр
тема: «Двойственный симплекс метод»

Выполнил: ст. группы ПВ-211
Стародубов Алексей Геннадьевич
Проверили:
Куртова Лилиана Николаевна
Вирченко Юрий Петрович

Белгород 2023 г.

Цель работы: изучить элементы теории двойственности, двойственный симплекс метод для пары симметрично двойственных задач, а также метод последовательного уточнения оценок.

Вариант- 19

1. Изучить правило составления двойственных задач, а также формулировки и применения первой, второй и третьей теорем двойственности.
2. Изучить двойственный симплекс-метод для симметрично двойственных задач. Составить и отладить программу решения пары симметрично двойственных задач двойственным симплекс-методом.
3. Изучить понятие псевдоплана, построение симплекс-таблицы, отвечающей псевдоплану. Освоить метод последовательного уточнения оценок. Составить и отладить программу решения задачи ЛП методом последовательного уточнения оценок.
4. Для подготовки тестовых данных решить вручную одну из следующих ниже задач двойственным симплекс-методом для пары симметрично двойственных задач, а также методом последовательного уточнения оценок.

2.

```
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>

using SimplexTable = std::vector<std::pair<std::string,
std::vector<double>>>;

void outputSimplexTable(const SimplexTable &simplexTable)
{
    std::cout << "BV\tFV\t";

    for (size_t i{1}; i < simplexTable.at(0).second.size(); ++i)
    {
        std::cout << "x" << i << '\t';
    }
```

```
    std::cout << '\n';
    for (size_t i{}; i < simplexTable.size(); ++i)
    {
        std::cout << simplexTable.at(i).first << '\t';
        for (size_t j{}; j < simplexTable.at(i).second.size();
++j)
        {
            std::cout << simplexTable.at(i).second.at(j) << '\t';
        }
        std::cout << '\n';
    }
    std::cout << '\n';
}

bool objFunctionHasNegative(const SimplexTable &simplexTable,
size_t &minNegativeIndex)
{
    size_t rowIndex{simplexTable.size() - 1};
    size_t minIndex{};
    bool findNegative{false};
    for (size_t i{1}; i < simplexTable.at(0).second.size(); ++i)
    {
        if (simplexTable.at(rowIndex).second.at(i) < 0)
        {
            findNegative = true;
            if (simplexTable.at(rowIndex).second.at(i) <
simplexTable.at(rowIndex).second.at(minIndex))
            {
                minIndex = i;
            }
        }
    }
    minNegativeIndex = minIndex;
    return findNegative;
}

bool exHasPositiveCoeff(const SimplexTable &simplexTable, const
size_t &colIndex, size_t &minCoeffIndex)
{
    size_t minCoeffIn{};
    double minCoeff{static_cast<double>(LONG_LONG_MAX)};
```

```
    bool findPositive{false};
    for (size_t i{}; i < simplexTable.size() - 1; ++i)
    {
        if (simplexTable.at(i).second.at(colIndex) > 0)
        {
            findPositive = true;
            const double coeff{simplexTable.at(i).second.at(0) /
simplexTable.at(i).second.at(colIndex)};
            if (coeff < minCoeff)
            {
                minCoeff = coeff;
                minCoeffIn = i;
            }
        }
    }
    minCoeffIndex = minCoeffIn;
    return findPositive;
}

double maxValueOfTheObjFunctionWithTableDisplay(SimplexTable
&simplexTable)
{
    size_t minIndex{};
    size_t minCoeffIndex{};
    unsigned simplexTableIndex{1};
    while (objFunctionHasNegative(simplexTable, minIndex))
    {
        std::cout << simplexTableIndex << " simplex table :\n";
        outputSimplexTable(simplexTable);
        if (exHasPositiveCoeff(simplexTable, minIndex,
minCoeffIndex))
        {
            const double
divider{simplexTable.at(minCoeffIndex).second.at(minIndex)};
            for (size_t i{}; i <
simplexTable.at(minCoeffIndex).second.size(); ++i)
            {
                simplexTable.at(minCoeffIndex).second.at(i) /=
divider;
            }
            for (size_t i{}; i < simplexTable.size(); ++i)
```

```
            {
                if (i != minCoeffIndex)
                {
                    const double divide{-
simplexTable.at(i).second.at(minIndex) /
simplexTable.at(minCoeffIndex).second.at(minIndex)};
                    for (size_t j{}; j <
simplexTable.at(i).second.size(); ++j)
                    {
                        simplexTable.at(i).second.at(j) +=
(divide * simplexTable.at(minCoeffIndex).second.at(j));
                    }
                }
            }
            simplexTable.at(minCoeffIndex).first = "x" +
std::to_string(minIndex);
        }
        else
        {
            std::cout << "The problem does not have solution(The
objective function is unbounded on the range of admissible values
of solutions)";
            std::exit(0);
        }
        ++simplexTableIndex;
    }
    std::cout << "Final simplex table :\n";
    outputSimplexTable(simplexTable);
    return simplexTable.at(simplexTable.size() - 1).second.at(0);
}

int main(int argc, char **argv)
{
    size_t numberOfRows{};
    size_t numberOfCols{};
    std::cout << "Number of rows in simplex table =  ";
    std::cin >> numberOfRows;
    std::cout << "Number of cols in simplex table = ";
    std::cin >> numberOfCols;
    std::cout << "Enter simplex table(with the names of basic
variables) : \n";
```

```
    SimplexTable simplexTable(numberOfRows);
    for (size_t i{}; i < numberOfRows; ++i)
    {
        std::string basisVarName{};
        std::cin >> basisVarName;
        simplexTable.at(i).first = basisVarName;
        for (size_t j{}; j < numberOfCols; ++j)
        {
            double value{};
            std::cin >> value;
            simplexTable.at(i).second.push_back(value);
        }
    }

    std::cout << std::setprecision(2);
    double
maxFunctValue{maxValueOfTheObjFunctionWithTableDisplay(simplexTab
le)};
    std::cout << "Max function value = " << maxFunctValue;

    return 0;
}
```

3.

```
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>

using SimplexTable = std::vector<std::pair<std::string,
std::vector<double>>>;

void outputSimplexTable(const SimplexTable &simplexTable)
{
    std::cout << "BV\tFV\t";

    for (size_t i{1}; i < simplexTable.at(0).second.size(); ++i)
    {
```

```
        std::cout << "x" << i << '\t';
    }
    std::cout << '\n';
    for (size_t i{}; i < simplexTable.size(); ++i)
    {
        std::cout << simplexTable.at(i).first << '\t';
        for (size_t j{}; j < simplexTable.at(i).second.size();
++j)
        {
            std::cout << simplexTable.at(i).second.at(j) << '\t';
        }
        std::cout << '\n';
    }
    std::cout << '\n';
}

bool objFunctionHasNegative(const SimplexTable &simplexTable,
size_t &minNegativeIndex)
{
    size_t rowIndex{simplexTable.size() - 1};
    size_t minIndex{};
    bool findNegative{false};
    for (size_t i{1}; i < simplexTable.at(0).second.size(); ++i)
    {
        if (simplexTable.at(rowIndex).second.at(i) < 0)
        {
            findNegative = true;
            if (simplexTable.at(rowIndex).second.at(i) <
simplexTable.at(rowIndex).second.at(minIndex))
            {
                minIndex = i;
            }
        }
    }
    minNegativeIndex = minIndex;
    return findNegative;
}

bool exHasPositiveCoeff(const SimplexTable &simplexTable, const
size_t &colIndex, size_t &minCoeffIndex)
{
```

```
    size_t minCoeffIn{};
    double minCoeff{static_cast<double>(LONG_LONG_MAX)};
    bool findPositive{false};
    for (size_t i{}; i < simplexTable.size() - 1; ++i)
    {
        if (simplexTable.at(i).second.at(0) /
simplexTable.at(i).second.at(colIndex) > 0)
        {
            findPositive = true;
            const double coeff{simplexTable.at(i).second.at(0) /
simplexTable.at(i).second.at(colIndex)};
            if (coeff < minCoeff)
            {
                minCoeff = coeff;
                minCoeffIn = i;
            }
        }
    }
    minCoeffIndex = minCoeffIn;
    return findPositive;
}

void maxValueOfTheObjFunctionWithTableDisplay(SimplexTable
&simplexTable)
{
    size_t minIndex{};
    size_t minCoeffIndex{};
    while (objFunctionHasNegative(simplexTable, minIndex))
    {
        std::cout << "Simplex table :\n";
        outputSimplexTable(simplexTable);
        if (exHasPositiveCoeff(simplexTable, minIndex,
minCoeffIndex))
        {
            const double
divider{simplexTable.at(minCoeffIndex).second.at(minIndex)};
            for (size_t i{}; i <
simplexTable.at(minCoeffIndex).second.size(); ++i)
            {
                simplexTable.at(minCoeffIndex).second.at(i) /=
divider;
```

```
            }
            for (size_t i{}; i < simplexTable.size(); ++i)
            {
                if (i != minCoeffIndex)
                {
                    const double divide{-
simplexTable.at(i).second.at(minIndex) /
simplexTable.at(minCoeffIndex).second.at(minIndex)};
                    for (size_t j{}; j <
simplexTable.at(i).second.size(); ++j)
                    {
                        simplexTable.at(i).second.at(j) +=
(divide * simplexTable.at(minCoeffIndex).second.at(j));
                    }
                }
            }
            simplexTable.at(minCoeffIndex).first = "x" +
std::to_string(minIndex);
        }
        else
        {
            std::cout << "The problem does not have solution(The
objective function is unbounded on the range of admissible values
of solutions)";
            std::exit(1);
        }
    }
    std::cout << "Simplex table :\n";
    outputSimplexTable(simplexTable);
}

bool hasNegativeFree(const SimplexTable &simplexTable, size_t
&minFreeIndex)
{
    size_t minFreeIn{};
    double minFree{static_cast<double>(LONG_LONG_MAX)};
    bool findNegative{false};
    for (size_t i{}; i < simplexTable.size() - 1; ++i)
    {
        if (simplexTable.at(i).second.at(0) < 0)
        {
```

```
            findNegative = true;
            if (simplexTable.at(i).second.at(0) < minFree)
            {
                minFree = simplexTable.at(i).second.at(0);
                minFreeIn = i;
            }
        }
    }
    minFreeIndex = minFreeIn;
    return findNegative;
}

bool hasNegativeCoeff(const SimplexTable &simplexTable, const
size_t &rowIndex, size_t &minColIndex)
{
    bool hasNegative{false};
    double minimum{static_cast<double>(LONG_LONG_MAX)};
    size_t minIndex{0};
    for (size_t i{1}; i <
simplexTable.at(rowIndex).second.size(); ++i)
    {
        if (simplexTable.at(rowIndex).second.at(i) < 0)
        {
            hasNegative = true;
            const double coeff{-
simplexTable.at(simplexTable.size() - 1).second.at(i) /
simplexTable.at(rowIndex).second.at(i)};
            if (coeff < minimum)
            {
                minimum = coeff;
                minIndex = i;
            }
        }
    }
    minColIndex = minIndex;
    return hasNegative;
}

double generalizedSimplexMethod(SimplexTable &simplexTable)
{
    maxValueOfTheObjFunctionWithTableDisplay(simplexTable);
```

```
    size_t minFreeIndex{};
    while (hasNegativeFree(simplexTable, minFreeIndex))
    {
        size_t minColIndex{};
        if (hasNegativeCoeff(simplexTable, minFreeIndex,
minColIndex))
        {
            const double
divide{simplexTable.at(minFreeIndex).second.at(minColIndex)};
            for (size_t i{}; i <
simplexTable.at(minFreeIndex).second.size(); ++i)
            {
                simplexTable.at(minFreeIndex).second.at(i) /=
divide;
            }

            for (size_t i{}; i < simplexTable.size(); ++i)
            {
                if (i != minFreeIndex)
                {
                    const double divider{-
simplexTable.at(i).second.at(minColIndex) /
simplexTable.at(minFreeIndex).second.at(minColIndex)};
                    for (size_t j{}; j <
simplexTable.at(i).second.size(); ++j)
                    {
                        simplexTable.at(i).second.at(j) +=
(divider * simplexTable.at(minFreeIndex).second.at(j));
                    }
                }
            }
            simplexTable.at(minFreeIndex).first = "x" +
std::to_string(minColIndex);
        }
        else
        {
            std::cout << " the problem has no solution due to the
absence of admissible solutions to the system of constraints\n";
            std::exit(1);
        }
        std::cout << "Simplex table :\n";
```

```
            outputSimplexTable(simplexTable);
        }
        return simplexTable.at(simplexTable.size() - 1).second.at(0);
}

int main(int argc, char **argv)
{
    size_t numberOfRows{};
    size_t numberOfCols{};
    std::cout << "Number of rows in simplex table =  ";
    std::cin >> numberOfRows;
    std::cout << "Number of cols in simplex table = ";
    std::cin >> numberOfCols;
    std::cout << "Enter simplex table(with the names of basic
variables) : \n";
    SimplexTable simplexTable(numberOfRows);
    for (size_t i{}; i < numberOfRows; ++i)
    {
        std::string basisVarName{};
        std::cin >> basisVarName;
        simplexTable.at(i).first = basisVarName;
        for (size_t j{}; j < numberOfCols; ++j)
        {
            double value{};
            std::cin >> value;
            simplexTable.at(i).second.push_back(value);
        }
    }

    std::cout << std::setprecision(2);
    double maxFunctValue{generalizedSimplexMethod(simplexTable)};
    std::cout << "Max function value = " << maxFunctValue;

    return 0;
}
```

4.

$$z = -2x_1 + 9x_2 + x_3 + 5x_4 \rightarrow \min$$

$$\begin{cases} -3x_1 + x_2 - 4x_3 + x_4 = 14 \\ -x_1 + 2x_2 + 5x_3 + 2x_4 \geq 30 \\ 6x_1 + 5x_2 + 2x_3 - 3x_4 = 28 \end{cases}$$

$$x_i \geq 0 \left(i = \overline{1,4}\right)$$

$$z_1 = 2x_1 - 9x_2 - x_3 - 5x_4 \rightarrow \max$$

$$\begin{cases} -3x_1 + x_2 - 4x_3 + x_4 \leq 14 \\ 3x_1 - x_2 + 4x_3 - x_4 \leq -14 \\ x_1 - 2x_2 - 5x_3 - 2x_4 \leq -30 \\ 6x_1 + 5x_2 + 2x_3 - 3x_4 \leq 28 \\ -6x_1 - 5x_2 - 2x_3 + 3x_4 \leq -28 \end{cases}$$

$$x_i \geq 0 \left(i = \overline{1,4}\right)$$

$$z_1 = 2x_1 - 9x_2 - x_3 - 5x_4 \rightarrow \max$$

$$\begin{cases} -3x_1 + x_2 - 4x_3 + x_4 + x_5 = 14 \\ 3x_1 - x_2 + 4x_3 - x_4 + x_6 = -14 \\ x_1 - 2x_2 - 5x_3 - 2x_4 + x_7 = -30 \\ 6x_1 + 5x_2 + 2x_3 - 3x_4 + x_8 = 28 \\ -6x_1 - 5x_2 - 2x_3 + 3x_4 + x_9 = -28 \end{cases}$$

$$x_i \geq 0 \left(i = \overline{1,4}\right)$$

Решаем полученную задачу обобщенным двойственным симплекс методом:

Таблица 1

Б	С	x1↓	x2	x3	x4	x5	x6	x7	x8	x9	Отн коэфф
x5	14	-3	1	-4	1	1	0	0	0	0	
x6	-14	3	-1	4	-1	0	1	0	0	0	
x7	-30	1	-2	-5	-2	0	0	1	0	0	
←x8	28	6	5	2	-3	0	0	0	1	0	4 2/3
x9	-28	-6	-5	-2	3	0	0	0	0	1	4 2/3
z	0	-2	9	1	5	0	0	0	0	0	

Таблица 2

Б	С	x1	x2	x3↓	x4	x5	x6	x7	x8	x9
x5	28	0	3 1/2	-3	- 1/2	1	0	0	1/2	0
x6	-28	0	-3 1/2	3	1/2	0	1	0	- 1/2	0
←x7	-34 2/3	0	-2 5/6	-5 1/3	-1 1/2	0	0	1	- 1/6	0
x1	4 2/3	1	5/6	1/3	- 1/2	0	0	0	1/6	0
x9	0	0	0	0	0	0	0	0	1	1
z	9 1/3	0	10 2/3	1 2/3	4	0	0	0	1/3	0
Отн			3 13/17	5/16	2 2/3				2	

Таблица 3

Б	С	x1	x2	x3	x4	x5	x6	x7	x8↓	x9
x5	47 1/2	0	5 3/32	0	11/32	1	0	- 9/16	19/32	0
←x6	-47 1/2	0	-5 3/32	0	- 11/32	0	1	9/16	- 19/32	0
x3	6 1/2	0	17/32	1	9/32	0	0	- 3/16	1/32	0
x1	2 1/2	1	21/32	0	- 19/32	0	0	1/16	5/32	0
x9	0	0	0	0	0	0	0	0	1	1
z	-1 1/2	0	9 25/32	0	3 17/32	0	0	5/16	9/32	0
Отн			1 150/163		10 3/11				9/19	

Таблица 4

Б	С	x1	x2↓	x3	x4	x5	x6	x7	x8	x9
x5	0	0	0	0	0	1	1	0	0	0
x8	80	0	8 11/19	0	11/19	0	-1 13/19	- 18/19	1	0
x3	4	0	5/19	1	5/19	0	1/19	- 3/19	0	0
x1	-10	1	- 13/19	0	- 13/19	0	5/19	4/19	0	0
←x9	-80	0	-8 11/19	0	- 11/19	0	1 13/19	18/19	0	1
z	-24	0	7 7/19	0	3 7/19	0	9/19	11/19	0	0
Отн			140/163		5 9/11					0

Таблица 5

Б	С	x1	x2	x3	x4↓	x5	x6	x7	x8	x9
x5	0	0	0	0	0	1	1	0	0	0
x8	0	0	0	0	0	0	0	0	1	1
x3	1 89/163	0	0	1	40/163	0	17/163	- 21/163	0	5/163
←x1	-3 101/163	1	0	0	- 104/163	0	21/163	22/163	0	- 13/163
x2	9 53/163	0	1	0	11/163	0	- 32/163	- 18/163	0	- 19/163
z	-92 116/163	0	0	0	2 142/163	0	1 150/163	1 64/163	0	140/163
Отн					4 1/2					10 10/13

Таблица 6

Б	С	x1	x2	x3	x4	x5	x6	x7	x8	x9
x5	0	0	0	0	0	1	1	0	0	0
x8	0	0	0	0	0	0	0	0	1	1
x3	2/13	5/13	0	1	0	0	2/13	- 1/13	0	0
x4	5 35/52	-1 59/104	0	0	1	0	- 21/104	- 11/52	0	1/8
x2	8 49/52	11/104	1	0	0	0	- 19/104	- 5/52	0	- 1/8
z	-109	4 1/2	0	0	0	0	2 1/2	2	0	1/2

В последней симплекс таблице псевдоплан является допустимым планом, а, следовательно, оптимальным планом задачи. $z_{1max} = -109$

Решение исходной задачи:

$z_1 = -z \rightarrow z_{min} = 109$

Тестовые данные:

```
PS D:\VS CODE CPlusPlus> .\rooster.exe
Number of rows in simplex table =  6
Number of cols in simplex table = 10
Enter simplex table(with the names of basic variables) :
x5 14 -3 1 -4 1 1 0 0 0 0
x6 -14 3 -1 4 -1 0 1 0 0 0
x7 -30 1 -2 -5 -2 0 0 1 0 0
x8 28 6 5 2 -3 0 0 0 1 0
x9 -28 -6 -5 -2 3 0 0 0 0 1
z 0 -2 9 1 5 0 0 0 0 0
Simplex table :
BV      FV      x1      x2      x3      x4      x5      x6      x7      x8      x9
x5      14      -3      1       -4      1       1       0       0       0       0
x6      -14     3       -1      4       -1      0       1       0       0       0
x7      -30     1       -2      -5      -2      0       0       1       0       0
x8      28      6       5       2       -3      0       0       0       1       0
x9      -28     -6      -5      -2      3       0       0       0       0       1
z       0       -2      9       1       5       0       0       0       0       0

Simplex table :
BV      FV      x1      x2      x3      x4      x5      x6      x7      x8      x9
x5      28      0       3.5     -3      -0.5    1       0       0       0.5     0
x6      -28     0       -3.5    3       0.5     0       1       0       -0.5    0
x7      -34.7   0       -2.83   -5.33   -1.5    0       0       1       -0.167  0
x1      4.67    1       0.833   0.333   -0.5    0       0       0       0.167   0
x9      0       0       0       0       0       0       0       0       1       1
z       9.33    0       10.7    1.67    4       0       0       0       0.333   0

Simplex table :
BV      FV      x1      x2      x3      x4      x5      x6      x7      x8      x9
x5      47.5    0       5.09    0       0.344   1       0       -0.562  0.594   0
x6      -47.5   0       -5.09   0       -0.344  0       1       0.562   -0.594  0
x3      6.5     -0      0.531   1       0.281   -0      -0      -0.188  0.0312  -0
x1      2.5     1       0.656   0       -0.594  0       0       0.0625  0.156   0
x9      0       0       0       0       0       0       0       0       1       1
z       -1.5    0       9.78    0       3.53    0       0       0.312   0.281   0

Simplex table :
BV      FV      x1      x2      x3      x4      x5      x6      x7      x8      x9
x5      0       0       0       0       0       1       1       0       0       0
x8      80      -0      8.58    -0      0.579   -0      -1.68   -0.947  1       -0
x3      4       0       0.263   1       0.263   0       0.0526  -0.158  0       0
x1      -10     1       -0.684  0       -0.684  0       0.263   0.211   0       0
x9      -80     0       -8.58   0       -0.579  0       1.68    0.947   0       1
z       -24     0       7.37    0       3.37    0       0.474   0.579   0       0
```

```
Simplex table :
BV      FV      x1      x2      x3      x4      x5      x6      x7      x8      x9
x5      0       0       0       0       0       1       1       0       0       0
x8      0       0       0       0       0       0       0       0       1       1
x3      1.55    0       0       1       0.245   0       0.104   -0.129  0       0.0307
x1      -3.62   1       0       0       -0.638  0       0.129   0.135   0       -0.0798
x2      9.33    -0      1       -0      0.0675  -0      -0.196  -0.11   -0      -0.117
z       -92.7   0       0       0       2.87    0       1.92    1.39    0       0.859

Simplex table :
BV      FV      x1      x2      x3      x4      x5      x6      x7      x8      x9
x5      0       0       0       0       0       1       1       0       0       0
x8      0       0       0       0       0       0       0       0       1       1
x3      0.154   0.385   0       1       0       0       0.154   -0.0769 0       6.94e-18
x4      5.67    -1.57   -0      -0      1       -0      -0.202  -0.212  -0      0.125
x2      8.94    0.106   1       0       0       0       -0.183  -0.0962 0       -0.125
z       -109    4.5     0       0       0       0       2.5     2       0       0.5

Max function value = -109
PS D:\VS CODE CPlusPlus>
```

Результат, полученный при решении задачи «вручную», совпал с результатом, полученным программой.

Вывод: изучил элементы теории двойственности, двойственный симплекс-метод для пары симметрично двойственных задач, а также метод последовательного уточнения оценок.