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)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №5

по дисциплине: Исследование операций
тема: «Двойственный симплекс метод»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
проф. Вирченко Юрий Петрович

Белгород 2024 г.

Лабораторная работа №5
Двойственный симплекс метод

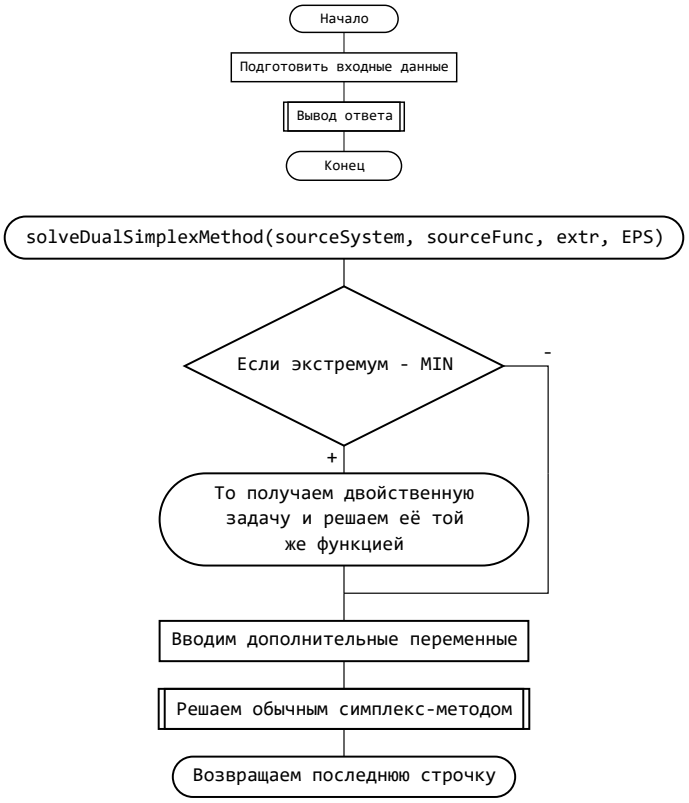
Цель работы: изучить элементы теории двойственности, двойственный симплекс-метод для пары симметрично двойственных задач, а также метод последовательного уточнения оценок.

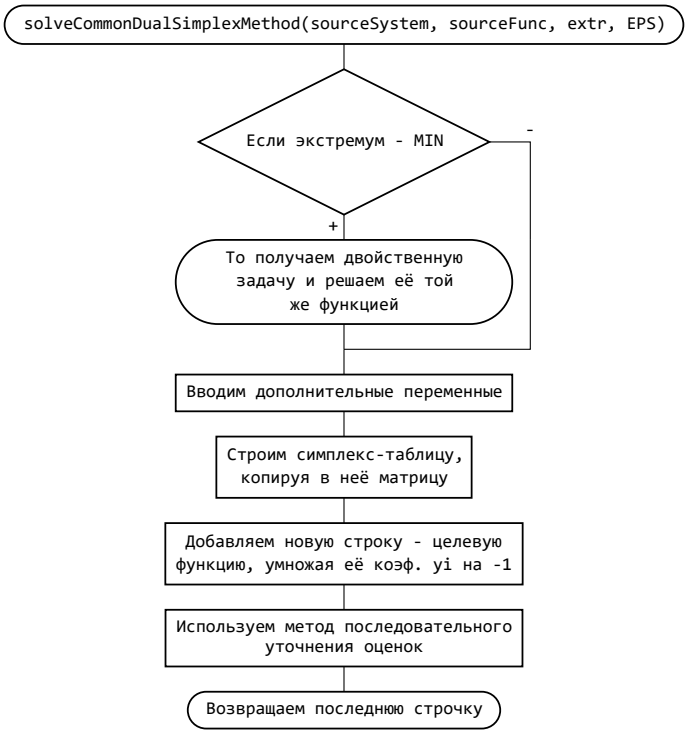
Задание: составить и отладить программы решения транспортной задачи распределительным методом и методом потенциалов. В рамках подготовки тестовых данных решить задачу вручную.

Вариант 10

$$z = -3x_1 + 9x_2 + x_3 + 4x_4 \to min;$$
$$\begin{cases} -4x_1 + x_2 - 4x_3 + x_4 \geq 15, \\ 2x_1 + 3x_2 + 4x_3 + 2x_4 \geq 30, \\ 3x_1 + 7x_2 + 2x_3 - 3x_4 = 25, \end{cases}$$
$$x_i \geq 0 (i = \overline{1,4}).$$

Блок-схемы:





Листинг программы:

```
#include <vector>
#include <array>

#include "../../libs/alg/alg.h"

int main() {
    // Подготовить входные данные
    std::vector<std::array<Fraction, 5>> matrix;
    matrix.push_back({{{-4}, {1}, {-4}, {1}, {15}}});
    matrix.push_back({{{2}, {3}, {4}, {2}, {30}}});
    matrix.push_back({{{3}, {7}, {2}, {-3}, {25}}});
    std::array<Fraction, 5> function{{{-3}, {9}, {1}, {4}, {0}}};

    // Вывод ответа
    auto g = solveDualSimplexMethod<5, 3, Fraction>(matrix, function, MIN, Fraction());
    std::cout << std::get<0>(g) << std::endl;
    auto l = std::get<1>(g);
    for (int i = 0; i < l.size(); i++) {
        std::cout << l[i] << " ";
    }
}
```

```
#pragma once

#include "../alg.h"

enum Extremum {
    MIN,
    MAX
};
```

```cpp
template <std::size_t T, std::size_t MatrixLines, typename CountType>
std::tuple<std::vector<std::array<CountType, MatrixLines + 1>>, std::array<CountType, MatrixLines + 1>, Extremum>
getDualProblem(std::vector<std::array<CountType, T>> sourceSystem, std::array<CountType, T> sourceFunc, Extremum extr) {
    std::vector<std::array<CountType, MatrixLines + 1>> resMatrix;
    for (int i = 0; i < T; i++) {
        resMatrix.push_back({});
    }
    std::array<CountType, MatrixLines + 1> resFunc;
    resFunc.back() = sourceFunc.back();

    for (int i = 0; i < MatrixLines; i++) {
        resFunc[i] = sourceSystem[i].back();

        for (int j = 0; j < T - 1; j++) {
            resMatrix[j][i] = sourceSystem[i][j];
        }
    }

    for (int j = 0; j < T - 1; j++) {
        resMatrix[j].back() = sourceFunc[j];
    }

    return {resMatrix, resFunc, extr == MIN ? MAX : MIN};
}

template <std::size_t T, std::size_t MatrixLines, typename CountType, std::size_t ExtendedMatrixSize = T + MatrixLines>
std::tuple<std::array<CountType, ExtendedMatrixSize>, std::vector<std::array<CountType, ExtendedMatrixSize>>>
↪  introduceNewVariables(std::vector<std::array<CountType, T>> sourceSystem, std::array<CountType, T> sourceFunc) {
std::vector<std::array<CountType, ExtendedMatrixSize>> newSystem;
    for (int i = 0; i < MatrixLines; i++) {
        newSystem.push_back({});
    }
    std::array<CountType, ExtendedMatrixSize> newFunc;
    for (int i = 0; i < MatrixLines; i++) {
        newSystem[i].back() = sourceSystem[i].back();
        int j;
        for (j = 0; j < T - 1; j++) {
            newSystem[i][j] = sourceSystem[i][j];
        }

        newSystem[i][j + i] = {1};
    }
    newFunc.back() = sourceFunc.back();
    for (int i = 0; i < T - 1; i++) {
        newFunc[i] = sourceFunc[i];
    }

    return {newFunc, newSystem};
}

template <std::size_t T, std::size_t MatrixLines, typename CountType, std::size_t ExtendedMatrixSize = T + MatrixLines>
std::tuple<CountType, std::vector<CountType>> solveDualSimplexMethod(std::vector<std::array<CountType, T>> sourceSystem,
↪  std::array<CountType, T> sourceFunc, Extremum extr, CountType EPS) {
```

```
    // Если экстремум - минимум
    if (extr == MIN) {
        // То получаем двойственную задачу и решаем её той же функцией
        auto reversed = getDualProblem<T, MatrixLines, CountType>(sourceSystem, sourceFunc, extr);
        auto res = solveDualSimplexMethod<MatrixLines + 1, T - 1, CountType>(std::get<0>(reversed),
        ↪  std::get<1>(reversed), std::get<2>(reversed), EPS);
        auto newF = std::get<1>(res);
        std::rotate(newF.begin(), newF.begin() + T - 2, newF.end());

        return {std::get<0>(res), newF};
    }

    // Вводим дополнительные переменные
    auto newVars = introduceNewVariables<T, MatrixLines, CountType>(sourceSystem, sourceFunc);
    auto newFunc = std::get<0>(newVars);
    auto newSystem = std::get<1>(newVars);

    // Решаем обычным симплекс-методом
    CountType ans = solveSimplexMethodMax(newSystem, newFunc, EPS);
    // Возвращаем последнюю строчку
    return {ans, std::vector<CountType>(newFunc.begin(), newFunc.end() - 1)};
}
```

```
#pragma once

#include "../alg.h"

template <std::size_t T, std::size_t MatrixLines, typename CountType, std::size_t ExtendedMatrixSize = T + MatrixLines>
std::tuple<CountType, std::vector<CountType>> solveCommonDualSimplexMethod(std::vector<std::array<CountType, T>>
↪  sourceSystem, std::array<CountType, T> sourceFunc, Extremum extr, CountType EPS) {
    // Если экстремум - минимум
    if (extr == MIN) {
        // То получаем двойственную задачу и решаем её той же функцией
        auto reversed = getDualProblem<T, MatrixLines, CountType>(sourceSystem, sourceFunc, extr);
        auto res = solveCommonDualSimplexMethod<MatrixLines + 1, T - 1, CountType>(std::get<0>(reversed),
        ↪  std::get<1>(reversed), std::get<2>(reversed), EPS);
        auto newF = std::get<1>(res);
        std::rotate(newF.begin(), newF.begin() + T - 2, newF.end());

        return {std::get<0>(res), newF};
    }

    // Вводим дополнительные переменные
    auto newVars = introduceNewVariables<T, MatrixLines, CountType>(sourceSystem, sourceFunc);
    auto newFunc = std::get<0>(newVars);
    auto newSystem = std::get<1>(newVars);

    // Строим симплекс-таблицу, копируя в неё матрицу newSystem
    std::vector<std::array<CountType, ExtendedMatrixSize>> simplexMatrix(newSystem);

    // Добавляем новую строку - целевую функцию, умножая её коэф. yi на -1
```

```
    simplexMatrix.push_back(newFunc);
    for (int i = 0; i < T; i++)
        simplexMatrix.back()[i] *= -1;

    CountType minusOne = {-1};
    CountType zero = EPS;
    // Используем метод последовательного уточнения оценок
    while (true) {
        int minRowIndex = -1;
        for (int i = 0; i < MatrixLines; i++)
            if (simplexMatrix[i].back() < zero && (minRowIndex == -1 || simplexMatrix[i].back() <
            ↪  simplexMatrix[minRowIndex].back()))
                minRowIndex = i;

        if (minRowIndex == -1) break;

        int minColumnIndex = -1;
        for (int i = 0; i < ExtendedMatrixSize - 1; i++)
            if (simplexMatrix[minRowIndex][i] < zero && (minColumnIndex == -1 ||
            (minusOne * simplexMatrix.back()[i] / simplexMatrix[minRowIndex][i]) < (minusOne *
            ↪  simplexMatrix.back()[minColumnIndex] / simplexMatrix[minRowIndex][minColumnIndex]))
                minColumnIndex = i;

        if (minColumnIndex == -1) throw std::invalid_argument("No solution");

        subtractLineFromOther(simplexMatrix, minRowIndex, minColumnIndex, EPS);
    }

    // Возвращам последнюю строчку
    return {simplexMatrix.back().back(), std::vector<CountType>(simplexMatrix.back().begin(), simplexMatrix.back().end()
    ↪  - 1)};
}
```

Результат выполнения программы:

```
95
0 7 0 8 0 7 0
```

Результаты вычислений:

$$z = -3x_1 + 9x_2 + x_3 + 4x_4 \to min;$$

$$\begin{cases} -4x_1 + x_2 - 4x_3 + x_4 \geq 15, \\ 2x_1 + 3x_2 + 4x_3 + 2x_4 \geq 30, \\ 3x_1 + 7x_2 + 2x_3 - 3x_4 = 25, \\ x_i \geq 0 (i = \overline{1,4}). \end{cases}$$

Получим двойственную задачу

$$z' = 15y_1 + 30y_2 + 25y_3 \to max;$$
$$\begin{cases} -4y_1 + 2y_2 + 3y_3 \le -3, \\ y_1 + 3y_2 + 7y_3 \le 9, \\ -4y_1 + 4y_2 + 2y_3 \le 1, \\ y_1 + 2y_2 - 3y_3 \le 4, \end{cases}$$
$$y_i \ge 0 (i = \overline{1,3}).$$

Получим симплекс-таблицу

Баз. пер.	Св. чл.	y_1	y_2	y_3	y_4	y_5	y_6	y_7
y_4	-3	2	-1	1	**3**	1	1	0
y_5	9	1	4	1	1	-2	0	1
y_6	1	-1	4	6	3	-8	0	0
y_7	4	-1	4	6	3	-8	0	0
z'	0	-2	-7	-8	-7	9	0	0

Баз. пер.	Св. чл.	y_1	y_2	y_3	y_4	y_5	y_6	y_7
y_3	-1	$-4/3$	**2/3**	1	$1/3$	0	0	0
y_5	16	$31/3$	$-5/3$	0	$-7/3$	1	0	0
y_6	3	$-4/3$	$8/3$	0	1	0	1	0
y_7	1	-3	4	0	1	0	0	1
z'	-25	$-145/3$	$-40/3$	0	$25/3$	0	0	0

Баз. пер.	Св. чл.	y_1	y_2	y_3	y_4	y_5	y_6	y_7
y_2	$-3/2$	**-2**	1	$3/2$	$1/2$	0	0	0
y_5	$27/2$	7	0	$5/2$	$-3/2$	1	0	0
y_6	7	4	0	-4	-2	0	1	0
y_7	7	5	0	-6	-1	0	0	1
z'	-45	-75	0	20	15	0	0	0

Баз. пер.	Св. чл.	y_1	y_2	y_3	y_4	y_5	y_6	y_7
y_1	$3/4$	1	$-1/2$	$-3/4$	$-1/4$	0	0	0
y_5	$33/4$	0	$7/2$	$31/4$	$1/4$	1	0	0
y_6	4	0	2	-1	-1	0	1	0
y_7	$13/4$	0	**5/2**	$-9/4$	$1/4$	0	0	1
z'	$45/4$	0	$-75/2$	$-145/4$	$-15/4$	0	0	0

Баз. пер.	Св. чл.	y_1	y_2	y_3	y_4	y_5	y_6	y_7
y_1	$7/5$	1	0	$-6/5$	$-1/5$	0	0	$1/5$
y_5	$37/10$	0	0	**109/10**	$-1/10$	1	0	$-7/5$

y_6	$7/5$	0	0	$4/5$	$-6/5$	0	1	$-4/5$
y_2	$13/10$	0	1	$-9/10$	$1/10$	0	0	$2/5$
z'	60	0	0	-70	0	0	0	15

Баз. пер.	Св. чл.	y_1	y_2	y_3	y_4	y_5	y_6	y_7
y_1	$197/109$	1	0	0	$-23/109$	$12/109$	0	$5/109$
y_3	$37/109$	0	0	1	$-1/109$	$10/109$	0	$-14/109$
y_6	$123/109$	0	0	0	$-130/109$	$-8/109$	1	$-76/109$
y_2	$175/109$	0	1	0	**10/109**	$9/109$	0	$31/109$
z'	$9130/109$	0	0	0	$-70/109$	$700/109$	0	$655/109$

Баз. пер.	Св. чл.	y_1	y_2	y_3	y_4	y_5	y_6	y_7
y_1	$11/2$	1	$23/10$	0	0	$3/10$	0	$7/10$
y_3	$1/2$	0	$1/10$	1	0	$1/10$	0	$-1/10$
y_6	22	0	13	0	0	1	1	3
y_4	$35/2$	0	$109/10$	0	1	$9/10$	0	$31/10$
z'	95	0	7	0	0	7	0	8

$x_1 = 0$, $x_2 = 7$, $x_3 = 0$, $x_4 = 8$, $z_{min} = 95$

Вывод: в ходе лабораторной работы изучили элементы теории двойственности, двойственный симплекс-метод для пары симметрично двойственных задач, а также метод последовательного уточнения оценок.