МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Системное программирование»

Лабораторная работа №2

**«Симплекс метод»**

по дисциплине

Теория систем и системный анализ

Выполнил: студент гр. БЭИ2202

Кулешов А. С.

Вариант 16

Проверил: Полищук Ю.В.

Москва, 2024 г

Содержание

[**1.** **Лабораторная работа №1** 3](#_Toc177284701)

[**1.1.** **Цель работы** 3](#_Toc177284702)

[**1.2.** **Задание** 3](#_Toc177284703)

[**1.3.** **Ход работы** 3](#_Toc177284704)

[**1.4.** **Заключение** 17](#_Toc177284705)

1. **Лабораторная работа №2**
   1. **Цель работы**

Изучить методы решения задачи линейного программирования алгоритмом симплекс-методом

* 1. **Задание**

Решить 7 заданий линейного программирования. Показать результаты решения, сверить ответы

* 1. **Ход работы**

**Теоретическая информация**

Решить задания линейного программирования можно при помощи симплекс-метода. Основа алгоритма заключается в перемещении между вершинами некой выпуклой фигуры в n-мерном пространстве, для этого в программной реализации заменяют «базисы» - переменные которые отвечают за вершину которую мы выбрали. Меняя вершины алгоритм может закончится одним из трёх способов: заменить вершину на новую невозможно, т.к. текущая оптимальная – в таком случае ответ найден и оптимум достигнут, если каждая оценка ниже нуля – план не ограничен, если достигнуто невозможное для разрешения уравнение – решения не существует.

**Практическая реализация**

Напишу программу на языке C++, использовав лишь 3 библитеки: «iomanip» и «iostream» – для вывода данных в консоль, «vector» – для упрощения работы с динамическими массивами. Перепроверю результаты программы с результатами, полученными с онлайн-решателя.

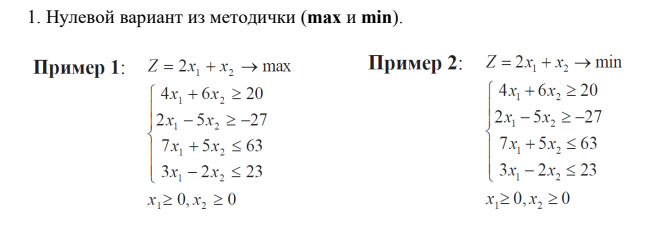


Рисунок 1 – условие первых двух заданий

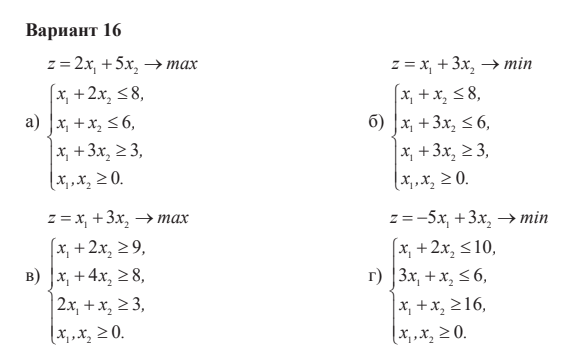


Рисунок 2 – условие четырёх индивидуальных заданий

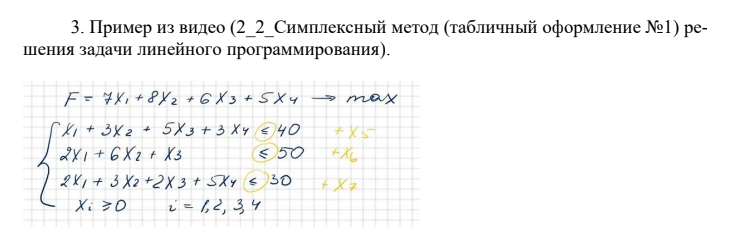


Рисунок 3 – условие последнего задания

Первое задание решено корректно, алгоритм реализованный в данной работе оперирует числами с плавающей точкой, а не дробями, как в выводе с онлайн-решателя, однако нетрудно заметить, что ответы совпадают.

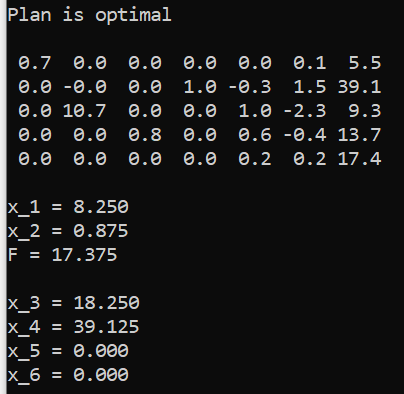


Рисунок 4 – результат работы программы на примере 1

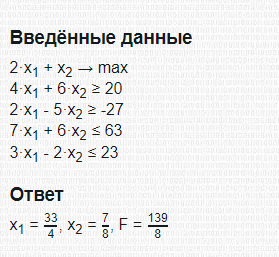


Рисунок 5 – проверка ответа

Аналогично произведу расчёт остальных заданий

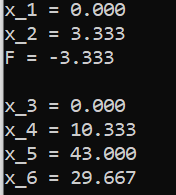


Рисунок 6 – результат работы программы на примере 2

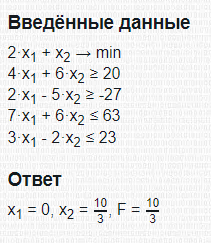


Рисунок 7 – проверка ответа

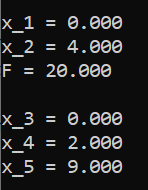


Рисунок 8 – результат работы программы на примере 1 (Вариант 16)

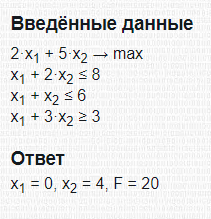


Рисунок 9 – проверка ответа

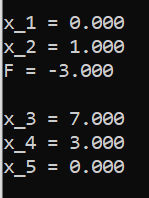


Рисунок 10 – результат работы программы на примере 2 (Вариант 16)

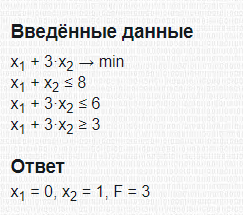


Рисунок 11 – проверка ответа

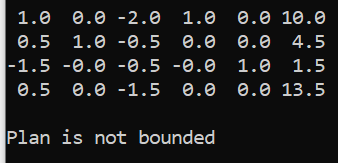


Рисунок 12 – результат работы программы на примере 3 (Вариант 16)

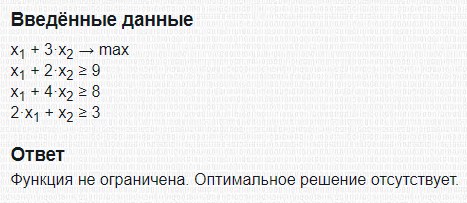


Рисунок 13 – проверка ответа

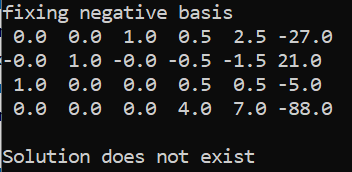


Рисунок 14 – результат работы программы на примере 4 (Вариант 16)

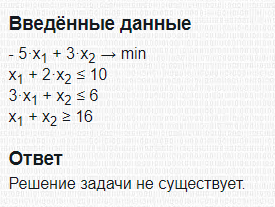


Рисунок 15 – проверка ответа

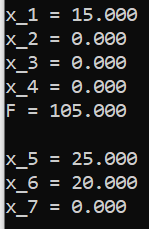


Рисунок 16 – результат работы программы на примере из видео

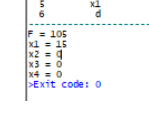


Рисунок 17 – проверка ответа (с ответом в методическом материале)

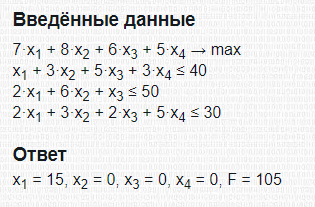


Рисунок 18 – проверка ответа

Программный код представлен ниже.

#include <iostream>

#include <vector>

#include <iomanip>

**using** **namespace** std;

**template** <**typename** T>

std::ostream& **operator**<<(std::ostream &in, **const** std::vector<T> &vect) {

**int** n = vect.size();

**for**(**int** i =**0**;i<n;i++){

in << fixed << setprecision(**1**) <<setw(**4**) << vect[i] << " " ;

}

**return** in;

}

**template** <**typename** T>

vector<T> **operator**\*(**const** vector<T>& vc, T value) {

vector<T> new\_vector = vc;

**for**(**int** i =**0**;i<vc.size();++i){

new\_vector[i] \*= value;

}

**return** new\_vector;

}

**template** <**typename** T>

vector<T> **operator**/(**const** vector<T>& vc, T value) {

vector<T> new\_vector = vc;

**for**(**int** i =**0**;i<vc.size();++i){

new\_vector[i] /= value;

}

**return** new\_vector;

}

**template** <**typename** T>

vector<T>& **operator**-=(vector<T>& vc, **const** vector<T>& sub) {

**for**(**int** i =**0**;i<vc.size();++i){

vc[i] -= sub[i];

}

**return** vc;

}

**template** <**typename** T>

vector<T>& **operator**/=(vector<T>& vc, T value) {

**for**(**int** i =**0**;i<vc.size();++i){

vc[i] /= value;

}

**return** vc;

}

**void** print(**const** vector< vector<**long** **double**> >& equations, **const** vector<**long** **double**>& F){

**for**(**auto**& vc : equations){

cout << vc << endl;

}

cout << F << endl;

cout << endl;

}

**const** **long** **double** eps = **0.001**;

**int** **get\_collumn**(**const** vector<**long** **double**>& F){

**int** index = -**1**;

**long** **double** min\_ = **0**;

**for**(**int** i = **0**; i < F.size() - **1**; ++i){

**if** (F[i] < min\_){

min\_ = F[i];

index = i;

}

}

**return** index;

}

**int** **get\_row**(**int** collumn\_id, **const** vector< vector<**long** **double**> >& equations){

**long** **double** min\_ = **1e100**;

**int** index = -**1**;

**for** (**int** i = **0**; i < equations.size(); ++i){

**long** **double** estimate = \*equations[i].rbegin() / equations[i][collumn\_id];

**if** (estimate <= **0**) **continue**;

**if** (estimate < min\_){

min\_ = estimate;

index = i;

}

}

**return** index;

}

**int** **fix\_negative\_basis**(vector< vector<**long** **double**> >& equations, vector<**long** **double**>& F){

cout << "fixing negative basis" << endl;

print(equations, F);

**int** row\_id = -**1**;

**for** (**int** i = **0**; i < equations.size(); ++i){

**if** (\*equations[i].rbegin() < **0**) row\_id = i;

}

**if** (row\_id == -**1**) **return** **0**;

**int** col\_id = -**1**;

**long** **double** min\_ = **0**;

**for** (**int** i = **0**; i < equations[row\_id].size() - **1**; ++i){

**if** (equations[row\_id][i] < min\_ ){

min\_ = equations[row\_id][i];

col\_id = i;

}

}

**if** (col\_id == -**1**){

cout << "Solution does not exist" << endl;

exit(-**2**);

}

equations[row\_id] /= equations[row\_id][col\_id];

**for**(**int** row = **0**; row < equations.size(); ++row){

**if** (row == row\_id) **continue**;

equations[row] -= equations[row\_id] \* equations[row][col\_id] / equations[row\_id][col\_id];

}

F -= equations[row\_id] \* F[col\_id] / equations[row\_id][col\_id];

**return** **1**;

}

**int** **iterate**(vector< vector<**long** **double**> >& equations, vector<**long** **double**>& F){

**while**(fix\_negative\_basis(equations, F)){}

print(equations, F);

**int** col\_id = get\_collumn(F);

**if** (col\_id == -**1**){

/// план оптимален, выходим

cout << "Plan is optimal" << endl;

**return** -**1**;

}

**int** row\_id = get\_row(col\_id, equations);

**if** (row\_id == -**1**){

cout << "Plan is not bounded" << endl;

exit(-**1**);

}

cout << row\_id << " " << col\_id << endl;

**for**(**int** row = **0**; row < equations.size(); ++row){

**if** (row == row\_id) **continue**;

equations[row] -= equations[row\_id] \* equations[row][col\_id] / equations[row\_id][col\_id];

}

F -= equations[row\_id] \* F[col\_id] / equations[row\_id][col\_id];

**return** **0**;

}

**void** **solve**(vector< vector<**long** **double**> >& equations, vector<**long** **double**>& F){

print(equations, F);

**while**(iterate(equations, F) == **0**){

print(equations, F);

}

cout << endl;

print(equations, F);

**for**(**int** col = **0**; col < equations[**0**].size(); ++col){

**int** index = -**1**;

**for**(**int** i = **0**; i < equations.size(); ++i){

**auto** elem = equations[i][col];

**if** (abs(elem) > eps){

**if** (index != -**1**){index = -**2**; **break**;}

index = i;

}

}

**if** (col == equations.size()) {

cout << fixed << setprecision(**3**) << "F = " << \*F.rbegin() << endl << endl;

}

**if** (index >= **0** )

cout << fixed << setprecision(**3**) <<"x\_" << col+**1** << " = " << \*equations[index].rbegin() / equations[index][col] << endl;

**else**

cout << fixed << setprecision(**3**) <<"x\_" << col+**1** << " = " << (**long** **double**) (**0**) << endl;

}

}

**int** **main**()

{

/\*\*

Программа решает поставленную задачу для канонического вида, т.е.

для случая с F = ... -> max,

x1 + x2 + ... <= ...

\*/

vector< vector<**long** **double**> > equations\_input;

vector<**long** **double**> F\_input;

/\*\* Пример из видео на ютубе (НЕ ТОГО КОТОРОГО ПОКАЗЫВАЛИ НА ПАРЕ!!!)

F\_input = {-3,-4}; /// F = 3\*x1 + 4\*x2 -> max

int n\_input = 2; /// число переменных

int m\_input = 2; /// число уравнений

equations\_input.push\_back({4,1,8}); /// 4x1 + x2 <= 8

equations\_input.push\_back({-1,1,3}); /// -x1 + x2 <= 3

\*/

/\*\*

///Пример 1, Нулевой вариант

F\_input = {-2,-1};

int m\_input = 4; /// число уравнений

equations\_input.push\_back({-4,-6,-20});

equations\_input.push\_back({-2,5,27});

equations\_input.push\_back({7,6,63});

equations\_input.push\_back({3,-2,23});

\*/

/\*\*

/// Нулевый вариант, задание 2

F\_input = {2,1};

int m\_input = 4; /// число уравнений

equations\_input.push\_back({-4,-6,-20});

equations\_input.push\_back({-2,5,27});

equations\_input.push\_back({7,6,63});

equations\_input.push\_back({3,-2,23});

\*/

/\*\*

/// Вариант 16, а

F\_input = {-2,-5};

int m\_input = 3; /// число уравнений

equations\_input.push\_back({1,2,8});

equations\_input.push\_back({1,1,6});

equations\_input.push\_back({-1,-3,-3});

\*/

/\*\*

/// Вариант 16, б

F\_input = {1,3};

int m\_input = 3; /// число уравнений

equations\_input.push\_back({1,1,8});

equations\_input.push\_back({1,3,6});

equations\_input.push\_back({-1,-3,-3});

\*/

/\*\*

/// Вариант 16, в

F\_input = {-1,-3};

int m\_input = 3; /// число уравнений

equations\_input.push\_back({-1,-2,-9});

equations\_input.push\_back({-1,-4,-8});

equations\_input.push\_back({-2,-1,-3});

\*/

/\*\*

/// Вариант 16, г

F\_input = {-5,3};

int m\_input = 3; /// число уравнений

equations\_input.push\_back({1,2,10});

equations\_input.push\_back({3,1,6});

equations\_input.push\_back({-1,-1,-16});

\*/

/\*\*

/// Задание из видео (2.2)

F\_input = {-7,-8,-6,-5};

int m\_input = 3; /// число уравнений

equations\_input.push\_back({1,3,5,3,40});

equations\_input.push\_back({2,6,1,0,50});

equations\_input.push\_back({2,3,2,5,30});

\*/

vector<**long** **double**> F = F\_input;

**for**(**int** i = **0**; i < m\_input + **1**; ++i){

F.push\_back(**0**);

}

vector< vector<**long** **double**> > equations = equations\_input;

**int** counter = **0**;

**for**(**auto**& vc : equations){

**auto** last\_elem = vc[vc.size() - **1**];

vc.pop\_back();

**for**(**int** i = **0**; i< m\_input; ++i){

**if** (counter == i){

vc.push\_back(**1**);

}**else**{

vc.push\_back(**0**);

}

}

counter++;

vc.push\_back(last\_elem);

}

cout << "input matrix" << endl;

print(equations, F);

solve(equations, F);

**return** **0**;

}

* 1. **Заключение**

В ходе проделанной работы мы познакомились с симплекс-методом решения задач линейного программирования