**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 1

**З дисципліни:** *“Дослідження операцій”*

**На тему:** *“Розв’язування задачі лінійного програмування*

*із двома змінними графічно та за симплекс-методом”*

**Лектор:**

доц. каф. ПЗ

Журавчак Л. М.

**Виконав:**

ст. гр. ПЗ-23

Михалевич П.-І.В.

**Прийняла:**

асист. каф. ПЗ

Івасько Т. М.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_\_ .

Львів – 2022

**Тема роботи:** Розв’язування задачі лінійного програмування із двома змінними графічно та за симплекс-методом.

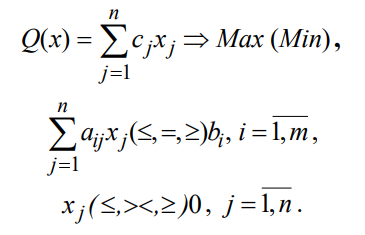
**Мета роботи:** Ознайомитись на практиці із основними поняттями теорії лінійного програмування, навчитись знаходити оптимальні плани задач лінійного програмування графічно, за допомогою симплекс методу.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

18. Чи може задача ЛП мати більше, ніж один, оптимальний розв’язок?

Так, наприклад коли при графічному розв’язуванні отримуєм пряму.

1. Сформулюйте задачу лінійного програмування.



4. Який план називається опорним?

Це базовий розв’язок системи, який неможливо виразити у вигляді лінійної комбінації будь-який інших розв’язків.

**ЗАВДАННЯ**

1. Отримати індивідуальний варіант завдання.

2. Записати математичну модель задачі ЛП та зазначити економічний зміст цільової функції і системи обмежень. Написати програму розв’язування та з її допомогою знайти розв’язок (максимальне (мінімальне) значення функції та значення змінних, при якому воно досягається) задачі ЛП згідно з варіантом з Додатка № 1 до лабораторної роботи № 1.

3. Розв’язати задачу ЛП з Додатка № 2 графічним методом (знайти максимальне та мінімальне значення функції та значення змінних, при якому вони досягаються).

4. Оформити звіт про виконану роботу.

5. Продемонструвати викладачеві результати, відповісти на запитання стосовно виконання роботи.

**ХІД ВИКОНАННЯ**

1. Індивідуальне завдання з додатка №1 та додатка №2.

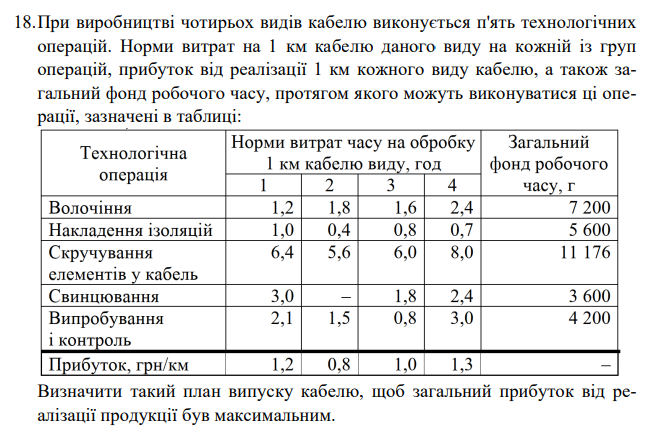


Рис. 1. Задача з додатка №1

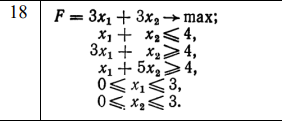


Рис. 2. Задача з додатка №2

1. Математична модель задачі ЛП:

F = 1.2x1 + 0.8x2 + x3 + 1.3x4 🡺 Max;

Обмеження:

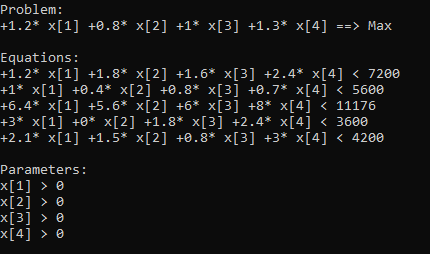
Змінні:

x1 >= 0

x2 >= 0

x3 >= 0

x4 >= 0



1. Код програми:

Приведеня до канонічного вигляду:

void SyntexSolver::ConvertToCanonical() {

for (auto& equation : m\_conditions.m\_parameters) {

switch (equation.GetSign())

{

case EquationSign::LessThan:

{

if (equation.GetSign() == EquationSign::LessThan) {

for (auto& a : equation.GetCoeficients()) {

a.second = -(a.second);

}

equation.SetSign(EquationSign::GreaterThan);

}

break;

}

case EquationSign::NotEqual:

{

m\_conditions.m\_removedParam.push\_back(std::pair<int, double>(equation.GetCoeficients()[0].first, equation.GetB()));

equation.SetSign(EquationSign::None);

equation.SetB(m\_conditions.m\_parameters.size());

Equation y1, y2;

y1.SetCoeficient(m\_conditions.m\_parameters.size(), 1);

y1.SetSign(EquationSign::GreaterThan);

y1.SetB(0);

y2.SetCoeficient(m\_conditions.m\_parameters.size()+1, 1);

y2.SetSign(EquationSign::GreaterThan);

y2.SetB(0);

m\_conditions.m\_parameters.push\_back(y1);

m\_conditions.m\_parameters.push\_back(y2);

break;

}

}

break;

}

for (auto& equation : m\_conditions.m\_equations) {

auto& coeficients = equation.GetCoeficients();

for (auto& coeficient : coeficients) {

auto& parameter = m\_conditions.m\_parameters[coeficient.first - 1];

if (parameter.GetSign() == EquationSign::None) {

auto paramY1 = m\_conditions.m\_parameters[static\_cast<int>(parameter.GetB())];

auto paramY2 = m\_conditions.m\_parameters[static\_cast<int>(parameter.GetB()) + 1];

auto prevCoef = coeficient.second;

coeficient = std::pair<int, double>(static\_cast<int>(parameter.GetB()), prevCoef);

auto coeficient2 = std::pair<int, double>(static\_cast<int>(parameter.GetB()) + 1, -prevCoef);

auto index = &coeficient - &coeficients[0];

coeficients.insert(coeficients.begin() + (index + 1), coeficient2);

}

}

if (equation.GetSign() == EquationSign::GreaterThan) {

Equation y;

y.SetCoeficient(m\_conditions.m\_parameters.size(), 1);

y.SetSign(EquationSign::GreaterThan);

y.SetB(0);

coeficients.push\_back(std::pair<int, double>(m\_conditions.m\_parameters.size() + 1, -1));

m\_conditions.m\_parameters.push\_back(y);

}

else if (equation.GetSign() == EquationSign::LessThan) {

Equation y;

y.SetCoeficient(m\_conditions.m\_parameters.size(), 1);

y.SetSign(EquationSign::GreaterThan);

y.SetB(0);

coeficients.push\_back(std::pair<int, double>(m\_conditions.m\_parameters.size() + 1, 1));

m\_conditions.m\_parameters.push\_back(y);

}

equation.SetSign(EquationSign::Equal);

}

auto& problemCoeficients = m\_conditions.m\_problem.GetCoeficients();

for (auto& coeficient : problemCoeficients) {

auto& parameter = m\_conditions.m\_parameters[coeficient.first];

if (parameter.GetSign() == EquationSign::None) {

auto paramY1 = m\_conditions.m\_parameters[static\_cast<int>(parameter.GetB())];

auto paramY2 = m\_conditions.m\_parameters[static\_cast<int>(parameter.GetB()) + 1];

auto prevCoef = coeficient.second;

coeficient = std::pair<int, double>(static\_cast<int>(parameter.GetB()), prevCoef);

auto coeficient2 = std::pair<int, double>(static\_cast<int>(parameter.GetB()) + 1, -prevCoef);

auto index = &coeficient - &problemCoeficients[0];

problemCoeficients.insert(problemCoeficients.begin() + (index + 1), coeficient2);

}

}

if (m\_conditions.m\_problem.GetB() == 0) {

for (auto& a : m\_conditions.m\_problem.GetCoeficients()) {

a.second = -(a.second);

}

m\_conditions.m\_problem.SetB(1);

}

AddAllParams();

}

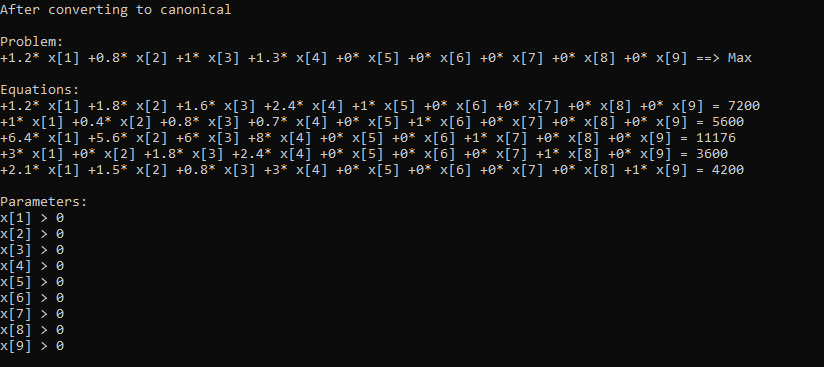


Рис. 3. Приведеня до канонічного вигляду

Обчислення таблиці:

void SyntexSolver::CalculateQ() {

for (int j = 0; j < m\_syntexTable[0].size(); ++j) {

m\_q[j] = 0;

}

double min = 0;

int min\_index = -1;

double min\_row = std::numeric\_limits<double>::max();

int min\_index\_row = -1;

const auto& problemCoef = m\_conditions.m\_problem.GetCoeficients();

for (int j = 0; j < m\_syntexTable.size(); ++j) {

m\_q[0] += m\_syntexTable[j][0] \* problemCoef[basisVectorIndices[j]].second;

}

for (int i = 1; i < m\_syntexTable[0].size(); ++i) {

for (int j = 0; j < m\_syntexTable.size(); ++j) {

m\_q[i] += (m\_syntexTable[j][i] \* problemCoef[basisVectorIndices[j]].second);

}

m\_q[i] -= problemCoef[i - 1].second;

if (m\_q[i] < 0 && m\_q[i] < min) {

min = m\_q[i];

min\_index = i;

}

}

m\_mainColIndex = min\_index;

if (m\_mainColIndex == -1) {

for (int j = 0; j < m\_syntexTable.size(); ++j) {

m\_lastColumn[j] = -1;

}

return;

}

for (int j = 0; j < m\_syntexTable.size(); ++j) {

if (m\_syntexTable[j][m\_mainColIndex] == 0) {

m\_lastColumn[j] = -1;

}

else {

m\_lastColumn[j] = m\_syntexTable[j][0] / m\_syntexTable[j][m\_mainColIndex];

if (m\_lastColumn[j] > 0 && m\_lastColumn[j] < min\_row) {

min\_row = m\_lastColumn[j];

min\_index\_row = j;

}

}

}

m\_lastColumn[m\_lastColumn.size() - 1] = min;

m\_mainRowIndex = min\_index\_row;

}

bool SyntexSolver::FoundOptimal() {

return m\_mainColIndex == -1;

}

void SyntexSolver::RecalculateTable() {

basisVectorIndices[m\_mainRowIndex] = m\_mainColIndex - 1;

// Perpendiculars.

for (int i = 0; i < m\_syntexTable.size(); ++i) {

if (i == m\_mainRowIndex) {

continue;

}

for (int j = 0; j < m\_syntexTable[0].size(); ++j) {

if (j == m\_mainColIndex) {

continue;

}

m\_syntexTable[i][j]

-= ((m\_syntexTable[i][m\_mainColIndex]\* m\_syntexTable[m\_mainRowIndex][j]) / m\_syntexTable[m\_mainRowIndex][m\_mainColIndex]);

}

}

for (int j = 0; j < m\_syntexTable[0].size(); ++j) {

if (j == m\_mainColIndex) {

continue;

}

m\_syntexTable[m\_mainRowIndex][j] /= m\_syntexTable[m\_mainRowIndex][m\_mainColIndex];

}

for (int i = 0; i < m\_syntexTable.size(); ++i) {

if (i == m\_mainRowIndex) {

m\_syntexTable[i][m\_mainColIndex] = 1;

continue;

}

m\_syntexTable[i][m\_mainColIndex] = 0;

}

}

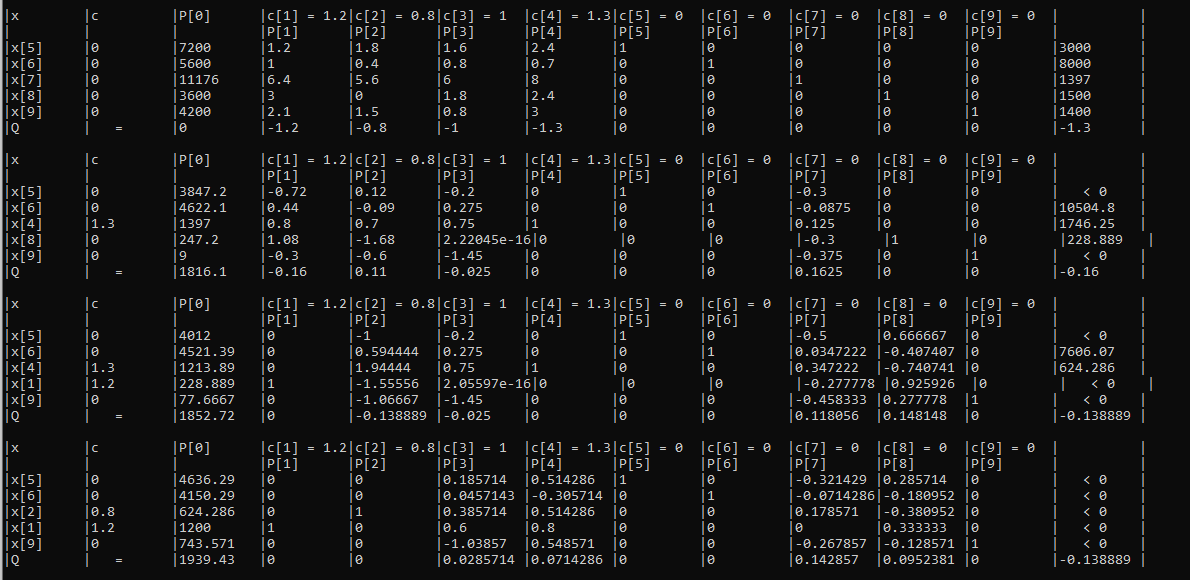


Рис. 4. Обчислення таблиці

Результати:

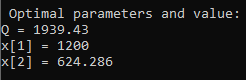


Рис. 5. Результати

1. Розв’язання задачі з додатку №2 .

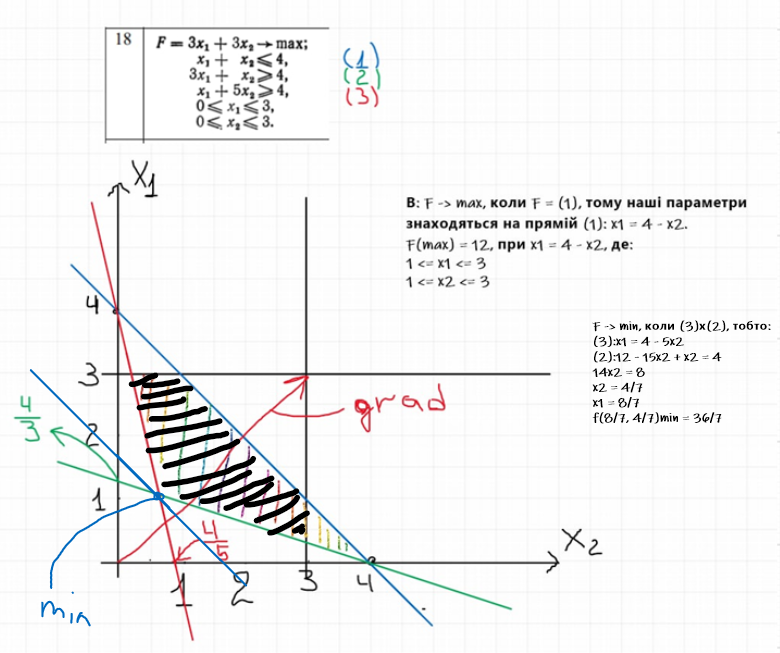


Рис. 6. Розв’язок задачі з додатку №2

**ВИСНОВКИ**

На цій лабораторній роботі я ознайомився з одним із методів вирішення задачі ЛП. Реалізував програму з використанням симплекс-методу, яка зводить дані до канонічного вигляду та шукає оптимальне рішення задачі. Також ознайомився із геометричним методом і знайшов максимальне значення фунції з двома параметрами.