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 №3**по дисциплине: «Исследование операций»

Выполнил: ст. группы ПВ-211

Медведев Дмитрий Сергеевич

Проверили:

Куртова Лилиана Николаевна

Вирченко Юрий

Петрович

Белгород 2023 г.

Модификации симплекс метода. Методы искусственного базиса и больших штрафов

# Вариант 8

**Цель работы:** изучение методов искусственного базиса и больших штрафов решения задач ЛП в канонической форме, не подготовленных к работе симплекс-методом в чистом виде.

**Ход работы**



1. Изучить метод и алгоритм искусственного базиса и составить программу решения задачи ЛП этим методом.
2. Изучить метод и алгоритм больших штрафов и составить программу решения задачи ЛП этим методом.

# Метод искусственного базиса

**Код библиотеки:**

SimplexTable SimplexTable::findArtificialBasis() {  
 SimplexTable processedTable;  
 **this**->copy(processedTable);  
 *//Делаем свободные члены положительными* **for** (**int** row = 0; row < processedTable.matrix.getRows() - 1; row++) {  
 **if** (processedTable.matrix.getData(row, processedTable.matrix.getColumns() - 1) < 0) {  
 processedTable.matrix.multiplyRowBy(row, -1);  
 }  
 }  
  
 *//Определяем значения коэффициентов целевой ф-ции подзадачи* **for** (**int** column = 0; column < processedTable.matrix.getColumns(); column++) {  
 **float** sum = 0;  
 **for** (**int** row = 0; row < processedTable.matrix.getRows() - 1; row++) {  
 sum += processedTable.matrix.getData(row, column);  
 }  
  
 processedTable.matrix.setData(processedTable.matrix.getRows() - 1, column, -sum);  
 }  
  
 *//Вводим искусственные переменные* **for** (**int** row = 0; row < processedTable.matrix.getRows() - 1; row++) {  
 processedTable.matrix.pushBackZeroColumn();  
 processedTable.matrix.setData(row, processedTable.matrix.getColumns() - 1, 1);  
 }  
  
 *//Применяем симплекс метод* **bool** hasSolution = **true**;  
 **bool** isSolution = **false**;  
 **while** (hasSolution && !isSolution) {  
 processedTable = processedTable.simplexMethodStep(hasSolution, isSolution);  
 }  
  
 *//Приводим к подходящему для решения исходной задачи виду* processedTable.matrix.deleteRow(processedTable.matrix.getRows() - 1);  
 **for** (**int** i = 0; i < processedTable.matrix.getRows(); i++) {  
 processedTable.matrix.deleteColumn(processedTable.matrix.getColumns() - 1);  
 }  
 vector<**float**> targetRow(**this**->matrix.getRow(**this**->matrix.getRows() - 1));  
 processedTable.matrix.addNewRow(targetRow);  
  
 **return** processedTable;  
}

**Код основной программы:**

#include **<iostream>**#include **"libs/matrix/matrix.h"**#include **"libs/simplexTable/simplexTable.h"  
  
int** main() {  
 SimplexTable simplexTable;  
 simplexTable.inputTable(2, 6,  
 **{**{1, 5, -3, -4, 2, 1, 14},  
 {2, 9, -5, -7, 4, 2, 32}  
 **}**,  
 **{**1, -3, 4, 5, -1, 8, 0**}**);  
  
  
 SimplexTable newTable = simplexTable.findArtificialBasis();  
  
  
 **float** maxFValue;  
 vector<**float**> solution = newTable.findSolutionBySimplexMethod(maxFValue);  
 **if** (solution.empty())  
 cout << **"Решений нет"**;  
 **else** {  
 cout << **"{"**;  
 **for** (**auto** i: solution) {  
 cout << i << **"; "**;  
 }  
  
  
 cout << **"\b\b}\n"**;  
 cout << **"Fmax = "** << maxFValue;  
 }  
}

**Результат работы программы:**



# Метод больших штрафов

**Код библиотеки:**

vector<**float**> SimplexTable::findSolutionBySimplexBigPenalty(**float** &maxFValue) {  
 SimplexTable processedTable;  
 **this**->copy(processedTable);  
  
 *//Делаем свободные члены положительными* **for** (**int** row = 0; row < processedTable.matrix.getRows() - 1; row++) {  
 **if** (processedTable.matrix.getData(row, processedTable.matrix.getColumns() - 1) < 0) {  
 processedTable.matrix.multiplyRowBy(row, -1);  
 }  
 }  
  
 **float** penalty = 10000;  
 *//Вводим искусственные переменные* **for** (**int** row = 0; row < processedTable.matrix.getRows() - 1; row++) {  
 processedTable.matrix.pushBackZeroColumn();  
 processedTable.matrix.setData(row, processedTable.matrix.getColumns() - 1, 1);  
 processedTable.matrix.setData(processedTable.matrix.getRows() - 1,  
 processedTable.matrix.getColumns() - 1,  
 penalty);  
 processedTable.matrix.subtractMultipliedRow(processedTable.matrix.getRows() - 1,  
 row, penalty);  
 }  
  
 **for** (**int** row = 0; row < processedTable.matrix.getRows() - 1; row++) {  
 processedTable.matrix.deleteColumn(processedTable.matrix.getColumns() - 1);  
 }  
  
 **return** processedTable.findSolutionBySimplexMethod(maxFValue);

**Код основной программы:**

#include **<iostream>**#include **"libs/matrix/matrix.h"**#include **"libs/simplexTable/simplexTable.h"  
  
int** main() {  
 SimplexTable simplexTable;  
 simplexTable.inputTable(2, 6,  
 **{**{1, 5, -3, -4, 2, 1, 14},  
 {2, 9, -5, -7, 4, 2, 32}  
 **}**,  
 **{**1, -3, 4, 5, -1, 8, 0**}**);  
  
  
 **float** maxFValue;  
 vector<**float**> solution = simplexTable.findSolutionBySimplexBigPenalty(maxFValue);  
 **if** (solution.empty())  
 cout << **"Решений нет"**;  
 **else** {  
 cout << **"{"**;  
 **for** (**auto** i: solution) {  
 cout << i << **"; "**;  
 }  
  
  
 cout << **"\b\b}\n"**;  
 cout << **"Fmax = "** << maxFValue;  
 }  
}

**Результат работы программы:**

****

1. Запрограммировать изученные алгоритмы и отладить соответствующие программы. В рамках подготовки тестовых данных решить вручную одну из следующих ниже задач.

**Нахождение искусственного базиса:**

# Метод искусственного базиса



**Решение исходной задачи:**

****

# Метод больших штрафов

****

**Вывод:** в ходе лабораторной работы мы изучили методы искусственного базиса и больших штрафов решения задач ЛП в канонической форме, не подготовленных к работе симплекс-методом в чистом виде.