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 №1

по дисциплине: «Исследование операций»

Вариант 3

Выполнил: ст. группы ПВ-211

Чувилко Илья Романович

Проверил:

Лилиана Куртова

Белгород 2023 г.

Тема: Исследование множества опорных планов системы ограничений задачи линейного программирования (задачи ЛП) в канонической форме

Цель работы: изучить метод Гаусса-Жордана и операцию замещения, а также освоить их применение к отысканию множества допустимых базисных видов системы линейных уравнений, и решению задачи линейного программирования простым перебором опорных решений.

Задания для подготовки к работе

1. Составить программу для отыскания всех базисных видов системы линейных уравнений.

2. Организовать отбор опорных планов среди всех базисных решений, а также нахождение оптимального опорного плана методом прямого перебора. Целевая функция выбирается произвольно.

3. Решить одну из следующих ниже задач вручную (подготовить тестовые данные).

$$\textbf{3.}\begin{cases} 2x_1 - x_2 + 6x_3 - x_4 + 3x_5 = 12 \\ 3x_1 + 5x_2 + x_3 - 12x_4 + 2x_5 = 14 \\ -3x_1 + 6x_2 + 8x_3 + 7x_4 - 4x_5 = 18 \end{cases}$$

Ручное решение:

$$\left(\begin{array}{ccccc|c} 2 & -1 & 6 & -1 & 3 & 12 \\ 3 & 5 & 1 & -12 & 2 & 14 \\ -3 & 6 & 8 & 7 & -4 & 18 \end{array}\right) \sim \left(\begin{array}{ccccc|c} 2 & -1 & 6 & -1 & 3 & 12 \\ 0 & 6\frac{1}{2} & -8 & -10\frac{1}{2} & -2\frac{1}{2} & -4 \\ 0 & 4\frac{1}{3} & 17 & 5\frac{1}{2} & \frac{1}{2} & 36 \end{array}\right) \sim \left(\begin{array}{ccccc|c} 2 & -1 & 6 & -1 & 3 & 12 \\ 0 & 6\frac{1}{2} & -8 & -10\frac{1}{2} & -2\frac{1}{2} & -4 \\ 0 & 0 & 22\frac{8}{39} & 12\frac{20}{21} & 2\frac{3}{13} & 38\frac{10}{13} \end{array}\right)$$

$\text{II} - 1{,}5\,\text{I}$
$\text{III} + 1{,}5\,\text{I}$

$\text{III} - \frac{9}{13}\,\text{II}$

$\operatorname{rang} A = \operatorname{rang} \bar{A} = 3$

Базисные решения:

1) $X_1 X_2 X_3$
2) $X_1 X_2 X_4$
3) $X_1 X_2 X_5$
4) $X_1 X_3 X_4$
5) $X_1 X_3 X_5$
6) $X_1 X_4 X_5$
7) $X_2 X_3 X_4$
8) $X_2 X_3 X_5$
9) $X_2 X_4 X_5$
10) $X_3 X_4 X_5$

1) $\left(\begin{array}{ccccc|c} 2 & -1 & 6 & -1 & 3 & 12 \\ 0 & 6\frac{1}{2} & -8 & -10\frac{1}{2} & -2\frac{1}{2} & -4 \\ 0 & 0 & 22\frac{8}{39} & 12\frac{20}{21} & 2\frac{3}{13} & 38\frac{10}{13} \end{array}\right) \sim \left(\begin{array}{ccccc|c} 2 & -1 & 6 & -1 & 3 & 12 \\ 0 & 6\frac{1}{2} & -8 & -10\frac{1}{2} & -2\frac{1}{2} & -4 \\ 0 & 0 & 1 & \frac{166}{293} & \frac{29}{293} & \frac{504}{293} \end{array}\right) \sim \left(\begin{array}{ccccc|c} 2 & -1 & 0 & -\frac{1289}{293} & \frac{705}{293} & \frac{492}{293} \\ 0 & 1 & 0 & -\frac{269}{293} & -\frac{1001}{586} & -\frac{770}{293} \\ 0 & 0 & 1 & \frac{166}{293} & \frac{29}{293} & \frac{504}{293} \end{array}\right) \sim$

$\text{II} + 8\,\text{IV}$
$\text{I} - 6\,\text{III}$
$\text{IV} : 6\frac{1}{2}$

$\text{I} + \text{II}$

$\sim \left(\begin{array}{ccccc|c} 2 & 0 & 0 & \frac{1558}{293} & -\frac{401}{586} & \frac{52}{293} \\ 0 & 1 & 0 & -\frac{269}{293} & -\frac{1001}{586} & -\frac{770}{293} \\ 0 & 0 & 1 & \frac{166}{293} & \frac{29}{293} & \frac{504}{293} \end{array}\right) \sim \left(\begin{array}{ccccc|c} 1 & 0 & 0 & \frac{779}{293} & -\frac{401}{1172} & \frac{26}{293} \\ 0 & 1 & 0 & -\frac{269}{293} & -\frac{1001}{586} & -\frac{770}{293} \\ 0 & 0 & 1 & \frac{166}{293} & \frac{29}{293} & \frac{504}{293} \end{array}\right)$

Базисные решения: $\left(\frac{26}{293}; -\frac{770}{293}; \frac{504}{293}; 0; 0\right)$

2) $\left(\begin{array}{ccccc|c} 2 & -1 & 6 & -1 & 3 & 12 \\ 0 & 6\frac{1}{2} & -8 & -10\frac{1}{2} & -2\frac{1}{2} & -4 \\ 0 & 0 & 22\frac{8}{39} & 12\frac{20}{21} & 2\frac{3}{13} & 38\frac{10}{13} \end{array}\right) \sim \left(\begin{array}{ccccc|c} 2 & -1 & 6 & -1 & 3 & 12 \\ 0 & 6\frac{1}{2} & -8 & -10\frac{1}{2} & -2\frac{1}{2} & -4 \\ 0 & 0 & \frac{383}{166} & 1 & \frac{29}{166} & \frac{252}{83} \end{array}\right) \sim \left(\begin{array}{ccccc|c} 2 & -1 & \frac{1289}{166} & 0 & \frac{523}{166} & \frac{1298}{83} \\ 0 & 1 & \frac{269}{166} & 0 & -\frac{13}{166} & \frac{5956}{10958} \\ 0 & 0 & \frac{383}{166} & 1 & \frac{29}{166} & \frac{252}{83} \end{array}\right) \sim$

$\text{III} : 12\frac{20}{21}$

$\text{II} + 10\frac{1}{2} \cdot \text{III}$
$\text{I} + \text{III}$
$\text{IV} : 6\frac{1}{2}$

$\text{I} + \text{II}$
$\text{I} : 2$

$\sim \left(\begin{array}{ccccc|c} 1 & 0 & \frac{279}{166} & 0 & \frac{255}{166} & \frac{11090}{10958} \\ 0 & 1 & \frac{269}{166} & 0 & -\frac{13}{166} & \frac{5956}{10958} \\ 0 & 0 & \frac{383}{166} & 1 & \frac{29}{166} & \frac{252}{83} \end{array}\right)$

Базисные решения: $\left(\frac{11090}{10958}; \frac{5956}{10958}; 0; \frac{252}{83}; 0\right)$

Код программы:

Содержимое заголовочного файла:

```
#ifndef CODE_MATRIX_H
#define CODE_MATRIX_H

#include <iostream>
#include <utility>
#include <vector>
#include <iomanip>

#define EPS 0.001

using namespace std;
using matrixRow = vector<double>;
using matrix = vector<matrixRow>;

class Matrix {
  public:
  matrix data;
  int nRows;
  int nColumns;
  int nSwap;

  explicit Matrix(int nRows, int nColumns) {
    this->nRows = nRows;
    this->nColumns = nColumns;
    nSwap = 0;
    for (int i = 0; i < nColumns; i++) {
      matrixRow r(nRows);
      data.push_back(r);
    }
  }

  explicit Matrix(matrix m) {
    this->nColumns = m[0].size();
    this->nRows = m.size();
    nSwap = 0;
    data = std::move(m);
  }

  void Output();
  void forwardGauss();
  double determinant();
  vector<vector<int>> generateCombinations();
  void deleteRow(int rowIndex);
  void reverseGauss(vector<int> &supports);
  bool CheckBasisMatrix(vector<int> &supports);
  void findAllBasis();
};


#endif //CODE_MATRIX_H
```

Содержимое исполняемого файла:

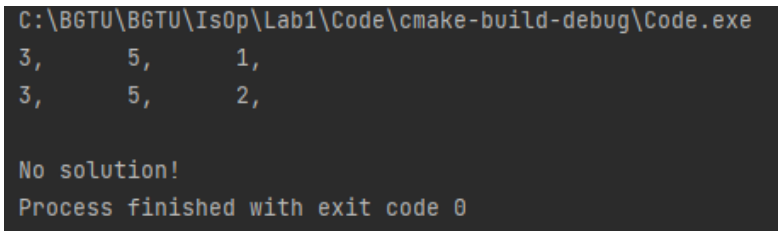
```
void Matrix::forwardGauss() {
  for (int i = 0; i < nRows - 1; i++) {
    for (int j = i + 1; j < nRows; j++)
      if (abs(data[i][i]) >= EPS) {
        double dif = data[j][i] / data[i][i];
        for (int k = i; k < nColumns; k++)
          data[j][k] -= data[i][k] * dif;
      }
  }
}

void Matrix::reverseGauss() {
  for (int i = nRows - 1; i >= 0; i--) {
    if (abs(data[i][i]) >= EPS) {
      double dif = data[i][i];
      for (int j = i; j < nColumns; j++)
        data[i][j] /= dif;
    }

    for (int j = i - 1; j >= 0; j--) {
      for (int k = nRows; k < nColumns; k++)
        data[j][k] -= data[j][i] * data[i][k];
      data[j][i] = 0;
    }
  }
}


bool Matrix::isFindZeroRow() {
  for (int i = 0; i < nRows; i++) {
    bool allZero = true;
    for (int j = 0; j < nRows; j++)
      if (abs(data[i][j]) >= EPS) {
        allZero = false;
        break;
      }
    if (allZero)
      return true;
  }
  return false;
}

void Matrix::deleteZeroRows() {
  for (int i = 0; i < nRows; i++) {
    bool allZero = true;
    for (int j = 0; j < nRows; j++) {
      if (abs(data[i][j]) > EPS) {
        allZero = false;
        break;
      }
    }
    if (allZero) {
      data[i].clear();
      data.erase(data.begin());
      nRows--;
    }
  }
}

double Matrix::determinant() {
```

```
 this->forwardGauss();
 this->deleteZeroRows();
 double det = data[0][0];
 for (int i = 1; i < nRows; i++) {
  det *= data[i][i];
 }
 if (nSwap % 2 == 1)
  det *= -1;
 return det;
}

void _generateCombinations(int n, int k, int i, int b,
                           vector<int> inputSet,
                           vector<int> generatingSet,
                           vector<vector<int>> &combinations) {
 for (int x = b; x <= n - k + i; x++) {
  vector<int> copyGeneratingSet = generatingSet;
  copyGeneratingSet.push_back(inputSet[x]);
  if (i == k) {
   combinations.push_back(generatingSet);
   return;
  } else
   _generateCombinations(n, k, i + 1, x + 1, inputSet,
                         copyGeneratingSet, combinations);
 }
}

vector<vector<int>> Matrix::generateCombinations() {
 vector<int> inputSet;
 for (int i = nColumns - 1 - 1; i >= 0; i--)
  inputSet.push_back(i);
 vector<vector<int>> combinations;
 vector<int> generatingSet;
 _generateCombinations(nColumns - 1, nRows, 0, 0, inputSet, generatingSet, combinations);
 return combinations;
}


void Matrix::reverseGauss(vector<int> &supports) {
 for (int i = nRows - 1; i >= 0; i--) {
  int j = supports[nRows - i - 1];
  if (abs(data[i][j]) >= EPS) {
   double dif = data[i][j];
   for (int k = 0; k < nColumns; k++)
    data[i][k] /= dif;
  }

  for (int k = i - 1; k >= 0; k--) {
   if (abs(data[i][j]) >= EPS) {
    double dif = data[k][j];
    for (int m = 0; m < nColumns; m++)
     data[k][m] -= dif * data[i][m];
   }
  }
 }
}

bool Matrix::CheckBasisMatrix(vector<int> &supports) {
 matrix m(nRows);
 for (int i = 0; i < nRows; i++)
  for (int j = nRows - 1; j >= 0; j--)
   m[i].push_back(data[i][supports[j]]);
 Matrix MMM(m);
```

```
 if (MMM.determinant() > 0)
   return true;
 return false;
}

bool Matrix::isHaveSolution() {
 for (int i = 0; i < nRows; i++) {
   bool isAllZero = true;
   for (int j = 0; j < nColumns - 1; j++) {
     if (abs(data[i][j]) >= EPS) {
       isAllZero = false;
       break;
     }
   }
   if (isAllZero && abs(data[i][nColumns - 1]) >= 0)
     return false;
 }
 return true;
}

void Matrix::findAllBasis() {
 Matrix M(data);
 M.forwardGauss();
 if (!M.isHaveSolution()) {
   cout << "No solution!";
   return;
 }

 vector<vector<int>> res = M.generateCombinations();
 cout << "Basis solutions:\n";
 for (auto &i: res) {
   if (CheckBasisMatrix(i)){
     Matrix tmp(M);
     tmp.reverseGauss(i);
     for (int j = 0; j < i.size(); j++)
       cout << 'x' << i[j] << " = " << tmp.data[j][nColumns - 1] << ",\t";
     cout << "\b\b\n";
   }
 }
}
```

Результат работы программы:

```
C:\BGTU\BGTU\IsOp\Lab1\Code\cmake-build-debug\Code.exe
2,      -1,     6,      -1,     3,      12,
3,      5,      1,      -12,    2,      14,
-3,     6,      8,      7,      -4,     18,

Basis solutions:
x4 = 2.55,      x3 = 10.4,      x2 = 17.4,
x4 = 4.93,      x3 = 10.4,      x1 = 17.4,
x4 = 74,        x3 = 10.4,      x0 = 17.4,
x4 = 1.89,      x2 = 2.29,      x1 = 17.4,
x4 = 7.71,      x2 = 2.29,      x0 = 17.4,
x4 = -17,       x1 = 6.07,      x0 = 17.4,
x3 = -2.76,     x2 = 2.65,      x0 = 3.04,
x3 = 9.66,      x1 = 4.29,      x0 = 3.04,
x2 = 1.59,      x1 = 1.5,       x0 = 1.72,

Process finished with exit code 0
```



Вывод: в ходе лабораторной работы мы изучили метод Гаусса-Жордана и операцию замещения, а также освоили их применение к отысканию множества допустимых базисных видов системы линейных уравнений, и решению задачи линейного программирования простым перебором опорных решени