Министерство науки и высшего

образования Российской Федерации

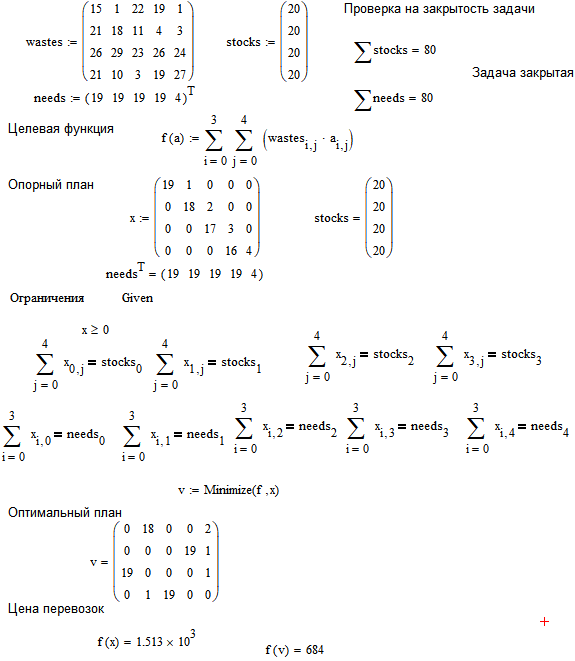
Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Рыбинский государственный авиационный технический университет  
имени П. А. Соловьева»

Кафедра математического и программного обеспеченияэлектронных вычислительных средств  
  
  
ЛАБОРАТОРНАЯ РАБОТА №3  
  
по дисциплине  
Исследование операций  
на тему  
«Решение транспортной задачи методом потенциалов»Вариант №2  
  
Студент группы ИПБ-22\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Новиков А. С.  
Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Задорина Н. А.  
  
  
  
Рыбинск 2024

Задание

Решить транспортную задачу, используя метод потенциалов.

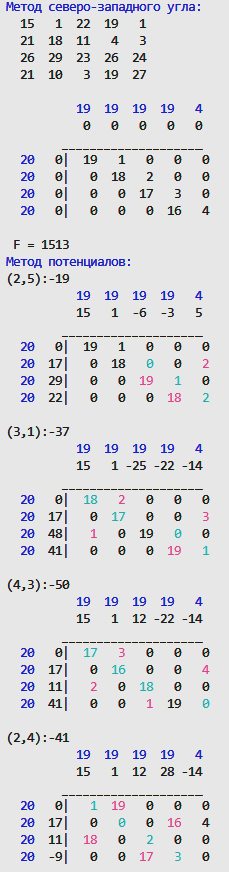
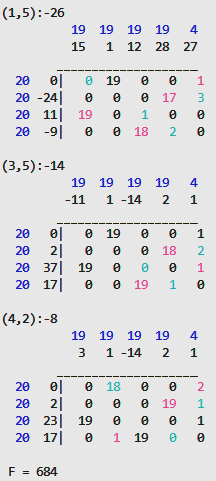
**Решение Mathcad**



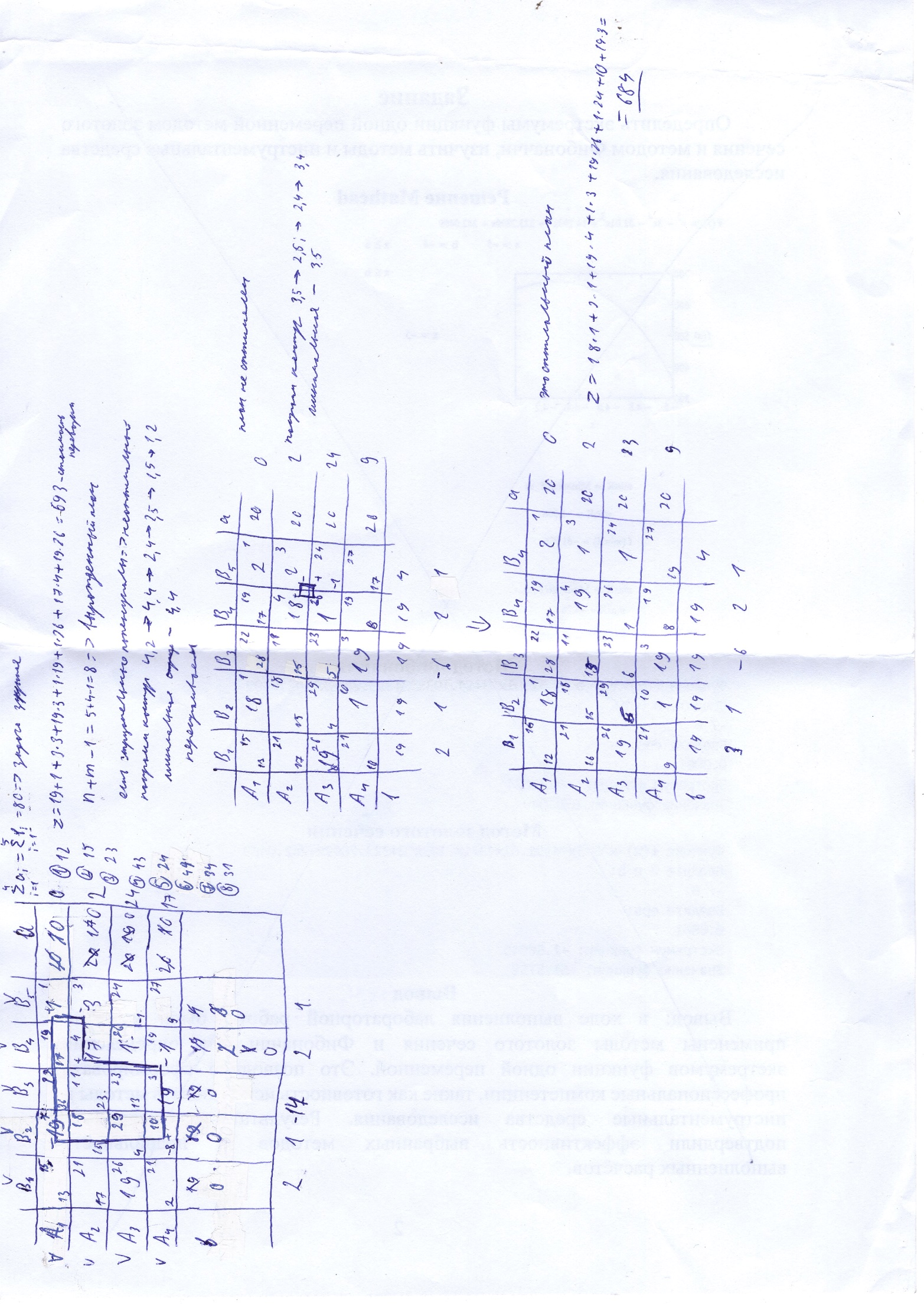
F(x) – траты при неоптимальном плане, построенном по методу северо-западного угла.

F(v) – траты при оптимальном плане

**Результат работы программы**

**Решение вручную**

****

**Код программы**

    void northwestCorner()

    {

        vector<int> newNeeds = needs;

        vector<int> newStocks = stocks;

        for (int i = 0; i < this->size\_row; i++)

        {

            for (int j = 0; j < this->size\_column; j++)

            {

                if (newStocks.at(i) < newNeeds.at(j) && newNeeds.at(j) != 0)

                {

                    matrix[i][j] += newStocks[i];

                    newNeeds[j] -= newStocks[i];

                    newStocks[i] = 0;

                }

                else if (newStocks.at(i) == newNeeds.at(j) && newStocks.at(i) != 0 && newNeeds.at(j) != 0)

                {

                    matrix[i][j] = newNeeds[j];

                    newStocks[i] = 0;

                    newNeeds[i] = 0;

                }

                else if (newStocks.at(i) > newNeeds.at(j) && newStocks.at(i) != 0 && newNeeds.at(j) != 0)

                {

                    matrix[i][j] += newStocks.at(i) - (newStocks.at(i) - newNeeds.at(j));

                    newStocks[i] -= newNeeds[j];

                    newNeeds[j] = 0;

                }

            }

        }

    }

    void calculating()

    {

        minimal\_element = INT\_MAX;

        for (int i = 0; i < size\_row; i++)

            for (int j = 0; j < size\_column; j++)

                if (matrix\_plus\_minus[i][j] == 1)

                    if (matrix[i][j] < minimal\_element)

                        minimal\_element = matrix[i][j];

        for (int i = 0; i < size\_row; i++)

            for (int j = 0; j < size\_column; j++)

                if (matrix\_plus\_minus[i][j] == 1)

                    matrix[i][j] -= minimal\_element;

                else if (matrix\_plus\_minus[i][j] == 0)

                    matrix[i][j] += minimal\_element;

        minimal\_element = 0;

    }

//Функция запуска поиска оптимального плана на основе неоптимального

    void start()

    {

        while (potentials())

        {

            for (int i = 0; i < size\_row; i++)

                for (int j = 0; j < size\_column; j++)

                    matrix\_plus\_minus[i][j] = -1;

            searchLoop(potential\_i, potential\_j);

            calculating();

            printMatrix();

        }

    }

    bool potentials() // метод потенциалов

    {

        vector<bool> boolU(size\_row, false);

        vector<bool> boolV(size\_column, false);

        boolU[0] = true;

        for (auto elem : u\_vec)

            elem = 0;

        for (auto elem : v\_vec)

            elem = 0;

        while (IsItNotTrue(boolU) || IsItNotTrue(boolV))

        {

            for (int i = 0; i < size\_row; i++)

            {

                for (int j = 0; j < size\_column; j++)

                {

                    if (matrix[i][j] != 0)

                    {

                        if (boolU[i])

                        {

                            v\_vec[j] = wastes[i][j] - u\_vec[i];

                            boolV[j] = true;

                        }

                        if (boolV[j])

                        {

                            u\_vec[i] = wastes[i][j] - v\_vec[j];

                            boolU[i] = true;

                        }

                    }

                }

            }

        }

        if (delta(u\_vec, v\_vec))

            return 1;

        return 0;

    }

    bool delta(vector<int> u\_vec, vector<int> v\_vec)

    {

        potential\_i = 0;

        potential\_j = 0;

        for (int i = 0; i < size\_row; i++)

        {

            for (int j = 0; j < size\_column; j++)

                if (matrix[i][j] == 0)

                {

                    this->free\_matrix[i][j] = wastes[i][j] - (u\_vec[i] + v\_vec[j]);

                    // нахождение наибольшего отрицательного значения

                    if (free\_matrix[i][j] < free\_matrix[potential\_i][potential\_j])

                    {

                        potential\_i = i;

                        potential\_j = j;

                    }

                }

        }

        // проверка существование отрицательных значений

        if (free\_matrix[potential\_i][potential\_j] < 0)

        {

            cout << "(" << potential\_i + 1 << ',' << potential\_j + 1 << "):" << free\_matrix[potential\_i][potential\_j] << endl;

            return 1;

        }

        return 0;

    }

    void searchLoop(int i, int j) // поиск контура

    {

        matrix\_plus\_minus[i][j] = 0;

        if (j && !way)

            go\_left(i, j - 1);

        if ((j != size\_column - 1) && !way)

            go\_right(i, j + 1);

        if (i && !way)

            go\_up(i - 1, j);

        if ((i != size\_row - 1) && !way)

            go\_down(i + 1, j);

        way = false;

        sign = 1;

    }

    void go\_down(int i, int j)

    {

        if ((i == potential\_i) && (j == potential\_j))

            way = true;

        if ((i != this->size\_row - 1) && !way)

            go\_down(i + 1, j);

        if (j && matrix[i][j] && !way)

        {

            go\_left(i, j - 1);

            if (way && matrix[i][j])

            {

                matrix\_plus\_minus[i][j] = sign;

                sign = 1 - sign;

            }

        }

        if ((j != this->size\_column - 1) && (matrix[i][j] != 0) && !way)

        {

            go\_right(i, j + 1);

            if (way && matrix[i][j])

            {

                matrix\_plus\_minus[i][j] = sign;

                sign = 1 - sign;

            }

        }

    }

    void go\_up(int i, int j)

    {

        if ((i == potential\_i) && (j == potential\_j))

            way = true;

        if (i && !way)

            go\_up(i - 1, j);

        if (j && matrix[i][j] && !way)

        {

            go\_left(i, j - 1);

            if (way && matrix[i][j])

            {

                matrix\_plus\_minus[i][j] = sign;

                sign = 1 - sign;

            }

        }

        if ((j != this->size\_column - 1) && matrix[i][j] && !way)

        {

            go\_right(i, j + 1);

            if (way && matrix[i][j])

            {

                matrix\_plus\_minus[i][j] = sign;

                sign = 1 - sign;

            }

        }

    }

    void go\_right(int i, int j)

    {

        if ((i == potential\_i) && (j == potential\_j))

            way = true;

        if ((j != this->size\_column - 1) && !way)

            go\_right(i, j + 1);

        if (i && matrix[i][j] && !way)

        {

            go\_up(i - 1, j);

            if (way && matrix[i][j])

            {

                matrix\_plus\_minus[i][j] = sign;

                sign = 1 - sign;

            }

        }

        if ((i != this->size\_row - 1) && matrix[i][j] && !way)

        {

            go\_down(i + 1, j);

            if (way && matrix[i][j])

            {

                matrix\_plus\_minus[i][j] = sign;

                sign = 1 - sign;

            }

        }

    }

    void go\_left(int i, int j)

    {

        if ((i == potential\_i) && (j == potential\_j))

            way = true;

        if (j && !way)

            go\_left(i, j - 1);

        if (i && matrix[i][j] && !way)

        {

            go\_up(i - 1, j);

            if (way && matrix[i][j])

            {

                matrix\_plus\_minus[i][j] = sign;

                sign = 1 - sign;

            }

        }

        if ((i != this->size\_row - 1) && matrix[i][j] && !way)

        {

            go\_down(i + 1, j);

            if (way && matrix[i][j])

            {

                matrix\_plus\_minus[i][j] = sign;

                sign = 1 - sign;

            }

        }

    }

};

int main()

{

    SetConsoleCP(1251);

    SetConsoleOutputCP(1251);

    // Matrix D(4, 4);

    // D.wastes = {{4, 8, 1, 6}, {3, 5, 3, 4}, {2, 6, 4, 3}, {1, 4, 5, 3}};

    // D.stocks = {70, 80, 90, 80};

    // D.needs = {60, 40, 120, 100};

    Matrix D(4, 5);

    D.wastes = {{15, 1, 22, 19, 1}, {21, 18, 11, 4, 3}, {26, 29, 23, 26, 24}, {21, 10, 3, 19, 27}};

    D.stocks = {20, 20, 20, 20};

    D.needs = {19, 19, 19, 19, 4};

    cout << "\x1B[33mМетод северо-западного угла: \x1B[37m" << endl;

    D.printWastes();

    D.northwestCorner();

    D.printMatrix();

    D.target\_func();

    cout << "\x1B[33mМетод потенциалов: \x1B[37m" << endl;

    D.start();

    D.target\_func();

    // system("pause");

}

**Вывод**

Вывод: в ходе выполнения лабораторной работы был успешно применен метод серверо-западного угла для поиска плана для транспортной задачи, был применен метод потенциалов для поиска, на его основе, оптимального плана на его основе. Совпадение результатов вычислений разными способами подтвердило правильность выполненных расчётов.