Звіт

Програмування інтелектуальних інформаційних систем

Лабораторна робота №5

“ Реалізувати симплексний метод для знаходження мінімуму функції заданої у табличному вигляді разом з системою обмежень на будь-якій мові програмування для наступних вхідних даних”

Студентки групи ІП-01

Галько Міли Вячеславівни

Лабораторна робота №5

“ Реалізувати симплексний метод для знаходження мінімуму функції заданої у табличному вигляді разом з системою обмежень на будь-якій мові програмування для наступних вхідних даних”

Варіант №6:

1. Розв’язати задачу лінійного програмування з такою матрицею коефіцієнтів системи обмежень (A), вектором вільних членів обмежень (B) і вектором коефіцієнтів цільової функції (C):

$$A=\begin{bmatrix} -1 & 1 & 1 & 0 & 0 \\ 5 & 0 & 1 & 1 & 1 \\ 3 & 2 & 0 & 0 & 1 \end{bmatrix};\quad B=(2,11,6);\quad C=(-6,1,-2,1,-1)$$

Виконання:

В першу чергу реалізуємо клас для утримання необхідних даних – class Arguments.

```
public class Arguments
{
    public double[,] array;
    public double[] B;
    public double[] C;
    public int[] basisVector;

    public int Height => array.GetLength(0);
    public int Width => array.GetLength(1);

    public double this[int i, int j]
    {
        get => array[i, j];
        set => array[i, j] = value;
    }
}
```

Далі реалізуємо class SimplexMethod.

```
public class SimplexMethod
{
    private Arguments _arguments;
    private double[] _bVector;
    private double[] _cVector;
    private double _zValue;
    private int[] _basisVector;

    public SimplexMethod(Arguments arguments)
    {
        _arguments = arguments;
        _bVector = arguments.B;
```

```
        _cVector = arguments.C.Select(el => el *= -1).ToArray();
        _basisVector = arguments.basisVector;
        DoDiagonalGauss();
        Normalise();
    }

    public void Solve()
    {
        Console.WriteLine(this);
        var i = 1;

        while (!_cVector.All(x => x <= 0.0000001))
        {
            SelectNewBasis();
            DoDiagonalGauss();
            Normalise();
            Console.WriteLine($"{i}) \n{this}");
            i++;
        }
    }

    private void SelectNewBasis()
    {
        for (var i = 0; i < _cVector.Length; i++)
        {
            var maxColumn = GetMaxIndex(_cVector, i);
            var current = Enumerable.Range(0, _arguments.Height)
                .OrderBy(index => _bVector[index] / _arguments[index,
maxColumn])
                .FirstOrDefault(index => _arguments[index, maxColumn] > 0, -
1);

            if (current == -1) continue;
            _basisVector[current] = maxColumn;
            break;
        }
    }

    private static int GetMaxIndex(double[] arr, int skipElement = 0)
    {
        return arr.OrderByDescending(a => a)
            .Select(el => Array.IndexOf(arr, el))
            .Skip(skipElement).FirstOrDefault(-1);
    }

    private void DoDiagonalGauss()
    {
        BotTriangleGauss();
        TopTriangleGauss();
    }

    private void BotTriangleGauss()
    {
        for (var i = 0; i < _basisVector.Length; i++)
        {
            var topRowIndex = -1;
            for (var row = i; row < _arguments.Height; row++)
```

```
            {
                if (_arguments[row, _basisVector[i]] == 0) continue;
                if (topRowIndex != -1)
                {
                    var multiplier = _arguments[row, _basisVector[i]];
                    var divider = 1.0 / _arguments[topRowIndex,
_basisVector[i]];

                    MultiplyRowOnNum(topRowIndex, divider);

                    _bVector[topRowIndex] *= divider;
                    _bVector[row] -= _bVector[topRowIndex] * multiplier;

                    SubtractTwoRows(row, topRowIndex, multiplier);
                }
                else
                {
                    topRowIndex = row;
                }
            }

            if (topRowIndex == i) continue;
            for (var j = 0; j < _arguments.Width; j++)
            {
                (_arguments[topRowIndex, j], _arguments[i, j]) =
(_arguments[i, j], _arguments[topRowIndex, j]);
            }

            (_bVector[topRowIndex], _bVector[i]) = (_bVector[i],
_bVector[topRowIndex]);
        }

        var last = _basisVector.Length - 1;
        var dividerOfLast = 1.0 / _arguments[last, _basisVector[^1]];
        MultiplyRowOnNum(last, dividerOfLast);
        _bVector[last] *= dividerOfLast;
    }

    private void TopTriangleGauss()
    {
        for (var i = _basisVector.Length - 1; i >= 0; i--)
        {
            for (var j = 0; j < i; j++)
            {
                var multiplier = _arguments[j, _basisVector[i]];
                _bVector[j] -= _bVector[i] * multiplier;
                SubtractTwoRows(j, i, multiplier);
            }
        }
    }

    public void SubtractTwoRows(int row1, int row2, double multiplier)
    {
        for (int i = 0; i < _arguments.Width; i++)
        {
            _arguments[row1, i] -= _arguments[row2, i] * multiplier;
        }
```

```
    }

    public void MultiplyRowOnNum(int row, double multiplier)
    {
        for (int i = 0; i < _arguments.Width; i++)
        {
            _arguments[row, i] *= multiplier;
        }
    }

    private void Normalise()
    {
        for (var i = 0; i < _basisVector.Length; i++)
        {
            var multiplier = _cVector[_basisVector[i]];

            if (multiplier == 0) continue;
            Enumerable.Range(0, _cVector.Length).ToList().ForEach(j =>
_cVector[j] -= _arguments[i, j] * multiplier);
            _zValue -= _bVector[i] * multiplier;
        }
    }
}
```

Результати:

```
    X1|    X2|    X3|    X4|    X5|     Z|
  20,0|   0,0|   0,0|   0,0|   4,0|  17,0|
   1,5|   1,0|   0,0|   0,0|   0,5|   3,0|
  -2,5|   0,0|   1,0|   0,0|  -0,5|  -1,0|
   7,5|   0,0|   0,0|   1,0|   1,5|  12,0|

1)
    X1|    X2|    X3|    X4|    X5|     Z|
   0,0|   0,0|   0,0|  -2,7|   0,0| -15,0|
   0,0|   1,0|   0,0|  -0,2|   0,2|   0,6|
   0,0|   0,0|   1,0|   0,3|   0,0|   3,0|
   1,0|   0,0|   0,0|   0,1|   0,2|   1,6|
```

Спершу бачимо наші вхідні дані представлені у вигляді Simplex таблиці. В результаті виконання методу отримали рішення:

x1 = 1,6

x2 = 0,6

x3 = 3

x4 = 0

x5 = 0

F = c1*x1 + c2*x2 + c3*x3 = (-6)*1,6 + 0,6 – 2*3 = -9,6 + 0,6 – 6 = -15