

Міністерство науки і освіти України  
Житомирський державний технологічний університет  
Кафедра програмного забезпечення систем

# Лабораторна робота

## З дисципліни «Математичні методи дослідження операцій»

Студента 3 курсу групи ПІ-48  
Галузь знань 0501 «Інформатика та  
обчислювальна техніка»  
Напрямок підготовки 6.050103  
«Програмна інженерія»

Виконав  
Перевірів

Бліндарук Т.В.  
Яремчук С.І.

**Тема:** Симплекс-метод

**Мета:** Створення програмного забезпечення для реалізації алгоритму знаходження оптимального розв’язку задач лінійного програмування.

## Зміст

Теоретичні відомості.....	3
Керівництво користувача.....	4
Керівництво програміста.....	6
Висновок.....	13

## Теоретичні відомості

**Задача лінійного програмування** - задача знаходження екстремуму лінійної функції цілі на множені припустимих значень, обмежених системою лінійних рівнянь.

**Симплекс-метод** — метод розв'язання задачі лінійного програмування, в якому здійснюється скерований рух по опорних планах до знаходження оптимального розв'язку; симплекс-метод також називають методом поступового покращення плану. Метод був розроблений американським математиком Джорджем Данцігом у 1947 році.

**Основна ідея симплекс-методу:** перебір кутових точок. На кожному кроці методу здійснюється перехід від даної кутової точки в сусідню, що знаходиться на одному ребрі з даною, зі значенням функції цілі меншим(або рівним) ніж в даній.

Для застосування симплекс-методу необхідно щоб задача мала зручний для цього вид.

Зручний вид задачі лінійного програмування для застосування симплекс-методу:

- 1) Функції цілі мінімізується
- 2) Вільні члени обмежень невід'ємні.
- 3) Обмеження складається лише з рівнянь.
- 4) Матриця коефіцієнтів при базисних змінних одинична(в кожному рівнянні міститься одна базисна змінна з коефіцієнтом "+1" і кожна базисна змінна міститься лише в одному рівнянні).
- 5) На всі змінні накладено умови невід'ємності.

## Керівництво користувача

Програма реалізована в вигляді веб сторінки і може бути запущена за допомогою браузера. Після запуску програми, буде відчинена перша форма, в якій необхідно вказати параметри задачі лінійного програмування. А саме, кількість обмежень та кількість керованих змінних.

Ведіть кількість рівнянь

Ведіть кількість змінних

Рисунок 1 Форма параметрів задачі лінійного

Після натиснення кнопки “Далі”, користувач потрапляє на форму заповнення даних, які стосуються задачі лінійного програмування.

5

	x1	x2	x3	знак	b
f1	<input type="text" value="3"/>	<input type="text" value="2"/>	<input type="text" value="-7"/>	<input "="" type="button" value="="/>	<input type="text" value="6"/>
f2	<input type="text" value="1"/>	<input type="text" value="3"/>	<input type="text" value="2"/>	<input "="" type="button" value="="/>	<input type="text" value="4"/>
f3	<input type="text" value="-2"/>	<input type="text" value="4"/>	<input type="text" value="1"/>	<input "="" type="button" value="&lt;="/>	<input type="text" value="5"/>
f4	<input type="text" value="2"/>	<input type="text" value="-3"/>	<input type="text" value="5"/>	<input "="" type="button" value="&lt;="/>	<input type="text" value="7"/>

Рисунок 2 Форма введення даних

Після натиснення на кнопку “Розрахування” відбудеться розв’язання задачі лінійного програмування. Якщо задача має вид не зручний для застосування симплекс-методу, програма виконає відповідні перетворення. В результаті розв’язання буде виведено результат.

localhost:63342/courseraCourses/Університет/ММДО/simplex/index.html

Додатки 404 Not Found | 355446282

Повідомлення від сторінки localhost:63342:

X(1.4285714285714286,0.8571428571428572,0,) F(x)= 9.714285714285715

☐ Заборонити створення додаткових діалогових вікон цією сторінкою.

5

	x1	x2
f1	<input type="text" value="3"/>	<input type="text" value="2"/>
f2	<input type="text" value="1"/>	<input type="text" value="3"/>
f3	<input type="text" value="-2"/>	<input type="text" value="4"/>
f4	<input type="text" value="2"/>	<input type="text" value="-3"/>

### Рисунок 3 Результат

# Керівництво програміста

Програма реалізована за допомогою можливостей мови Java Script.

Точкою входу додатку є HTML документ index.html. Саме в цьому документі відбувається підключення необхідних скриптів а також запуск програми.

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
<script src="node_modules/jquery/dist/jquery.min.js"></script>
<script src="js/Input.js"></script>
<script src="js/Simplex.js"></script>
<script src="js/ElementarySimplex.js"></script>
<script>
  var M = 100000000;
  (new Input()).firstView();
</script>
</body>
</html>
```

В програмі є 3 основні класи:

- Input — служить для вводу інформації
- Simplex — використовується для приведення ЗЛП до виду зручного для застосування симплекс методом.
- ElementarySimplex — Використовується для розв'язування симплекс методом ЗЛП.

## Class Simplex

```
/**
 * Created by t.blindaruk on 17.06.16.
 */
/**
 * @param goalsFunction
 * @param matrix
 * @constructor
 */
function Simplex(goalsFunction, matrix) {
  this.goalsFunction = goalsFunction;
  this.matrix = matrix;
  this.changeGoalFunction = 0;
  this.changeMatrix = [];
}
Simplex.prototype.run = function () {
  var self = this;
```

```

    if (self.goalsFunction[self.goalsFunction.length - 1] === -1) {
        self.changeGoalFunction = 1;
        for (var i = 0; i < self.goalsFunction.length; ++i) {
            self.goalsFunction[i] = -self.goalsFunction[i];
        }
    }
    self.goalsFunction.splice(-1, 1);
    self.base = [];
    this.checkB();
    this.lessEqual();
    if (!this.moreEqualAndEqual()) {
        this.more();
    }
    this.equals();
    for (var y = 0; y < self.matrix.length; ++y) {
        self.matrix[y].splice(-2, 1);
    }
    debugger;
    for (var iy = 0; iy < self.matrix.length; ++iy) {
        for (var ix = self.matrix[iy].length - 2; ix >= 0; --ix) {
            if (self.matrix[iy][ix] === 1) {
                var count = 0;
                for (var iyy = 0; iyy < self.matrix.length; ++iyy) {
                    if (self.matrix[iyy][ix] === 1) {
                        count++;
                    }
                }
                if (count === 1) {
                    self.base.push(ix);
                    break;
                }
            }
        }
    }
    debugger;
    return {
        'matrix': self.matrix,
        'goalsFunction': self.goalsFunction,
        'base': self.base
    }
};
Simplex.prototype.checkB = function () {
    var self = this;
    for (var i = 0; i < self.matrix.length; ++i) {
        if (self.matrix[i][self.matrix[i].length - 1] < 0) {
            for (var ix = 0; ix < self.matrix[i].length; ++ix) {
                self.matrix[i][ix] = -self.matrix[i][ix];
            }
        }
    }
};
Simplex.prototype.lessEqual = function () {
    var self = this;
    for (var y = 0; y < self.matrix.length; ++y) {
        var lengthVectorY = self.matrix[y].length;
        if (self.matrix[y][lengthVectorY - 2] === -1) {
            for (var yy = 0; yy < self.matrix.length; ++yy) {
                self.matrix[yy].splice(-2, 0, ((yy === y) ? 1 : 0));
            }
            self.goalsFunction.push(0);
            self.changeMatrix.push(y);
        }
    }
};
Simplex.prototype.moreEqualAndEqual = function () {

```

```

var self = this;
var bestEqualIndex = -1;
var bestEqualValue = -1;
var vectorMore = [];
var bestMoreIndex = -1;
var bestMoreValue = -1;
for (var y = 0; y < self.matrix.length; ++y) {
    var sign = self.matrix[y][self.matrix[y].length - 2];
    if (sign === 1) {
        vectorMore.push(y);
        var value = self.matrix[y][self.matrix[y].length - 1];
        if (bestMoreValue < value) {
            bestMoreIndex = y;
            bestMoreValue = value;
        }
    } else if (sign === 0) {
        var value = self.matrix[y][self.matrix[y].length - 1];
        if (bestEqualValue < value) {
            bestEqualIndex = y;
            bestEqualValue = value;
        }
    }
}
if (bestEqualIndex !== -1 && bestMoreIndex !== -1) {
    if (bestEqualValue < bestMoreValue) {
        var k = parseInt((bestMoreValue / bestEqualValue) + 1);
        for (var i = 0; i < self.matrix[bestEqualIndex].length; ++i) {
            self.matrix[bestEqualIndex][i] *= k;
        }
    }
    for (var y = 0; y < self.matrix.length; ++y) {
        var lengthVectorY = self.matrix[y].length;
        if (self.matrix[y][lengthVectorY - 2] === 1) {
            for (var yy = 0; yy < self.matrix.length; ++yy) {
                self.matrix[yy].splice(-2, 0, ((yy === y) ? -1 : 0));
            }
            for (var ix = 0; ix < self.matrix[y].length; ++ix) {
                self.matrix[y][ix] = self.matrix[bestEqualIndex][ix] - self.matrix[y][ix];
            }
            self.goalsFunction.push(0);
            self.changeMatrix.push(y);
        }
    }
    return true;
}
return false;
};

Simplex.prototype.more = function () {
    var self = this;
    var bestMoreIndex = -1;
    var bestMoreValue = -1;
    for (var i = 0; i < self.matrix.length; ++i) {
        if (self.matrix[i][self.matrix[i].length - 2] === 1) {
            var value = self.matrix[i][self.matrix[i].length - 1];
            if (bestMoreValue < value) {
                bestMoreIndex = i;
                bestMoreValue = value;
            }
        }
    }
    if (bestMoreIndex !== -1) {
        for (var iy = 0; iy < self.matrix.length; ++iy) {
            self.matrix[iy].splice(-2, 0, ((iy === bestMoreIndex) ? -1 : 0));
        }
        self.goalsFunction.push(bestMoreIndex);
    }
}

```



```

        for (var y = 0; y < self.matrix.length; ++y) {
            if (self.matrix[y][self.matrix[y].length - 2] === 1 && y !== bestMoreIndex) {
                for (var iy = 0; iy < self.matrix.length; ++iy) {
                    self.matrix[iy].splice(-2, 0, ((iy === y) ? -1 : 0));
                }
                for (var ix = 0; ix < self.matrix[y].length; ++ix) {
                    self.matrix[y][ix] = self.matrix[bestMoreIndex][ix] - self.matrix[y][ix];
                }
                self.changeMatrix.push(y);
                self.goalsFunction.push(0);
            }
        }
    }
};

function find(array, value) {
    var result = false;
    for (var i = 0; i < array.length; ++i) {
        if (array[i] === value) {
            result = true;
            break;
        }
    }
    return result;
}

Simplex.prototype.equals = function () {
    var self = this;
    for (var y = 0; y < self.matrix.length; ++y) {
        if (!find(self.changeMatrix, y)) {
            for (var iy = 0; iy < self.matrix.length; ++iy) {
                self.matrix[iy].splice(-2, 0, (iy === y) ? 1 : 0);
            }
            self.goalsFunction.push(M);
        }
    }
}
}

```

Клас який відповідає за перетворення ЗЛП до виду зручного для застосування симплекс методом

### Class ElementarySimplex

Даний клас відповідає за розв'язування симплекс таблиці.

Конструктор містить

- goalFunction,
- matrix,
- basis

Для обрахування симплекс таблиці необхідно визвати функцію run.

```

/**
 * Created by tania on 15.06.16.
 */
/**
 * @param goalFunction
 * @param matrix
 * @param basis
 * @constructor
 */

```

```

function ElementarySimplex(goalFunction, matrix, basis) {
    var self = this;
    self._option = {};
    //self._option.goalFunction = [-2, -1, 0, 0, 0];
    self._option.goalFunction = goalFunction;
    //self._option.matrix = [
    //    [-1, 1, 1, 0, 0, 2],
    //    [3, 0, -2, 1, 0, 3],
    //    [1, 0, 3, 0, 1, 12]
    //];
    self._option.matrix = matrix;
    //self._option.basis = [1, 3, 4];
    self._option.basis = basis;
}
/**
 * @returns {{status: string, goalFunction: (*|Array), matrix: (*|Array), basis: (*|Array)}}
 */
ElementarySimplex.prototype.run = function () {
    var result = {
        'status': 'go',
        'goalFunction': this._option.goalFunction,
        'matrix': this._option.matrix,
        'basis': this._option.basis
    };
    do {
        debugger;
        result = this.runIteration(result);
    } while (result['status'] === 'go');
    return result;
};
/**
 * @param option
 * @returns {{status: string, goalFunction: *, matrix: *, basis: *}}
 */
ElementarySimplex.prototype.runIteration = function (option) {
    var goalFunction = option['goalFunction'];
    var matrix = option['matrix'];
    var basis = option['basis'];
    var evaluationVector = this.findEvaluation(goalFunction, matrix, basis);
    var maxEv = this.findMaxEvaluation(evaluationVector);
    var status = 'go';
    if (maxEv !== -1) {
        var row = this.findMinX(matrix, maxEv);
        if (row !== '-') {
            matrix = this.nextTable(matrix, row, maxEv);
            if (basis[row] === maxEv) {
                status = 'end';
            }
            basis[row] = maxEv;
        } else {
            status = 'unlimited';
        }
    } else {
        status = 'end';
    }
    if (row === maxEv) {
        status = 'end';
    }
    return {
        'status': status,
        'goalFunction': goalFunction,
        'matrix': matrix,
        'basis': basis
    };
};

```

```

/**
 * @param goalFunction
 * @param matrix
 * @param basis
 * @returns {Array}
 */
ElementarySimplex.prototype.findEvaluation = function (goalFunction, matrix, basis) {
    var functionVector = new Array(goalFunction.length);
    for (var y = 0; y < matrix.length; ++y) {
        for (var x = 0; x < matrix[y].length - 1; ++x) {
            if (!functionVector[x]) {
                functionVector[x] = 0;
            }
            functionVector[x] += goalFunction[basis[y]] * matrix[y][x];
        }
    }
    for (var i = 0; i < goalFunction.length; ++i) {
        functionVector[i] -= goalFunction[i];
    }
    return functionVector;
};

/**
 * @param vectorEvaluation
 * @returns {number}
 */
ElementarySimplex.prototype.findMaxEvaluation = function (vectorEvaluation) {
    var j = -1;
    var evaluation = -1;
    for (var i = 0; i < vectorEvaluation.length; ++i) {
        if (evaluation < vectorEvaluation[i]) {
            evaluation = vectorEvaluation[i];
            j = i;
        }
    }
    return j;
};

/**
 * @param matrix
 * @param maxEvaluation
 * @returns {string}
 */
ElementarySimplex.prototype.findMinX = function (matrix, maxEvaluation) {
    var vectorRelations = [];
    var functionLength = 0;
    for (var y = 0; y < matrix.length; ++y) {
        functionLength = matrix[y].length;
        if (matrix[y][maxEvaluation] <= 0) {
            vectorRelations.push('-');
        } else {
            vectorRelations.push(matrix[y][functionLength - 1] / matrix[y][maxEvaluation]);
        }
    }
    var min = '-';
    var j = '-';
    for (var i = 0; i < vectorRelations.length; ++i) {
        if ((!isFinite(min) && isFinite(vectorRelations[i])) ||
            (isFinite(min) && isFinite(vectorRelations[i]) && min > vectorRelations[i])) {
            min = vectorRelations[i];
            j = i;
        }
    }
    return j;
};

```

```

* @param matrix
* @param row
* @param coll
* @returns {*}
*/
ElementarySimplex.prototype.nextTable = function (matrix, row, coll) {
    var pivotNumber = matrix[row][coll];
    for (var i = 0; i < matrix[row].length; ++i) {
        matrix[row][i] /= pivotNumber;
    }
    for (var y = 0; y < matrix.length; ++y) {
        if (y !== row) {
            var pivotRowNumber = -matrix[y][coll];
            for (var x = 0; x < matrix[y].length; ++x) {
                matrix[y][x] += matrix[row][x] * pivotRowNumber;
            }
        }
    }
    return matrix;
};
ElementarySimplex.prototype.getPoint = function(){
    var self = this;
    var point = [];
    for(var i = 0; i < self._option.goalFunction.length; ++i){
        point.push(0);
    }
    for(var i = 0; i < self._option.basis.length; ++i){
        point[self._option.basis[i]] = self._option.matrix[i][self._option.matrix[i].length-1];
    }
    return point;
}

```

## Висновок

Під час виконання лабораторної роботи було створено web-додаток, який надає змогу своїм користувачам отримувати оптимальний розв'язок задач лінійного програмування. Алгоритм програми базується на використанні алгоритму симплекс-методу. Додаток розрахований як на опрацювання задач лінійного програмування, що мають зручний вигляд для застосування симплекс методу, так і задач, для яких потребується приведення до зручного вигляду для розв'язування (окрім обмежень). Дана програма є зручною у використанні.