Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №2

Метод отсекающих плоскостей

Выполнил:

студент гр. 953504

Кондрашов И.Д.

Проверил:

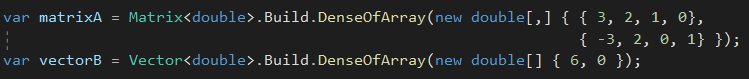
кандидат физико-математических наук

Дугинов О.И.

Минск 2022

1. **Пример работы программы:**

Входные данные:



Ответ:

A screenshot of a computer

Description automatically generated with medium confidence

1. **Исходный код программы (C#)**

Класс Simplex.cs – это методы, используемые в 2 и 3 лабораторных работах дисциплины «Методы оптимизации и управления»

public static class Simplex

{

private static Vector<double> basis = Vector<double>.Build.DenseOfArray(new double[] { 0 });

/\* Методы из ЛР №3 МОиУ 6 семестр \*/

public static Tuple<ArraySegment<double>, List<double>> StartSimplexMethod(Matrix<double> matrixA, Vector<double> vectorB)

{

var rowsCount = matrixA.RowCount;

var columnsCount = matrixA.ColumnCount;

var zeros = Vector<double>.Build.Dense(columnsCount, 0);

var identityMatrix = Matrix<double>.Build.DenseIdentity(2, 2);

var ones = Vector<double>.Build.Dense(rowsCount, 1);

for (int i = 0; i < vectorB.Count; i++)

{

if (vectorB[i] < 0)

{

vectorB[i] \*= -1;

for (int j = 0; j < matrixA.ColumnCount; j++)

{

matrixA[i, j] \*= -1;

}

}

}

var newMatrix = matrixA.Append(identityMatrix);

var vectorJb = Vector<double>.Build.DenseOfEnumerable(ones.Select((z, i) => columnsCount + z + i));

var vectorC = Vector<double>.Build.DenseOfEnumerable(zeros.Concat(ones.Negate()));

var vectorX = Vector<double>.Build.DenseOfEnumerable(zeros.Concat(vectorB));

var (resX, resB) = MainSimplexMethod(newMatrix, vectorC, vectorX, vectorJb);

var subResX = resX.SubVector(columnsCount, resX.Count - columnsCount);

if (subResX.Any(x => x != 0))

{

throw new Exception("Задача не совместима");

}

//Находим индекс положительного элемента

int k = 0;

for (int i = 0; i < resB.Count; i++)

{

if (resB.ElementAt(i) > 0)

{

k = i;

}

}

var array = new double[columnsCount - basis.Count];

for (int i = 0; i < columnsCount - 1; i++)

{

array[i] = i + 1;

}

var vectorJnb = Vector<double>.Build.DenseOfEnumerable(array.Where(x => !basis.Contains(x)));

var test = Matrix<double>.Build.DenseOfColumns(resB.Select(j => newMatrix.Column((int)j - 1).ToArray()));

var invertedAb = test.Inverse();

var vectorL = vectorJnb.Select(j => invertedAb.Multiply(newMatrix.Column((int)j)).ToArray()).ToArray();

if (vectorL.All(x => x[k] == 0))

{

vectorJb.ToList().RemoveAt(k);

}

//выбираем элементы из resX по позициям от 0 до columnsCount

var res = new ArraySegment<double>(resX.ToArray(), 0, columnsCount);

return Tuple.Create(res, vectorJb.ToList());

}

//Методы из ЛР №2 по МОиУ 6 семестр

private static Tuple<Vector<double>, Vector<double>> MainSimplexMethod(Matrix<double> matrixA, Vector<double> vectorC, Vector<double> vectorX, Vector<double> vectorB)

{

int replaceIndex = 0;

//Бесконечный цикл, но с итерациями

for (int i = 0; ; i++)

{

var basisMatrix = Matrix<double>.Build.DenseOfColumnArrays(vectorB.Select(j => matrixA.Column((int)j - 1).ToArray()).ToArray());

var invertMatrix = basisMatrix.Inverse();

var vectorCB = Vector<double>.Build.DenseOfEnumerable(vectorB.Select(j => vectorC[(int)j - 1]));

var potentialVector = vectorCB \* invertMatrix;

var gradeVector = (potentialVector \* matrixA) - vectorC;

int j0 = -1;

var number = gradeVector.ToList().FirstOrDefault(x => !vectorB.Contains(gradeVector.ToList().IndexOf(x) + 1) && x < 0);

if (number != 0)

{

j0 = gradeVector.ToList().IndexOf(number);

}

if (j0 == -1)

{

return Tuple.Create(vectorX, vectorB);

}

var vectorZ = invertMatrix \* matrixA.Column(j0);

var vectorT = vectorZ.Select((z, i) => z > 0 ? vectorX[(int)vectorB[i] - 1] / z : Double.PositiveInfinity).ToArray();

var tetta = vectorT.Min();

if (tetta == Double.PositiveInfinity)

{

throw new Exception("Функция не ограничена");

}

replaceIndex = Array.IndexOf(vectorT, tetta);

for (int j = 0; j < vectorB.Count; j++)

{

double k = vectorB[j];

vectorX[(int)k - 1] -= tetta \* vectorZ[j];

}

vectorB[replaceIndex] = j0 + 1;

vectorX[j0] = tetta;

}

}

}

Класс Program.cs – основной класс программы, где реализованы методы

1. Main(string[] args) – точка выхода в программу, здесь задаются входные данные и вызываются нужные функции для их обработки
2. GetFractionPart(double num) – метод, возвращающий дробную часть числа
3. ClippingPlaneMethod(Matrix<double> matrixA, Vector<double> vectorB) – метод, выполняющий задание лабораторной (clipping plane – отсекающий плоскости)

class Program

{

static void Main(string[] args)

{

var matrixA = Matrix<double>.Build.DenseOfArray(new double[,] { { 3, 2, 1, 0},

{ -3, 2, 0, 1} });

var vectorB = Vector<double>.Build.DenseOfArray(new double[] { 6, 0 });

var result = ClippingPlaneMethod(matrixA, vectorB);

foreach (var res in result)

{

Console.WriteLine($"Элемент итогового списка: {res}");

}

Console.ReadKey();

}

private static double GetFractionPart(double num)

{

return num - Math.Floor(num);

}

private static List<double> ClippingPlaneMethod(Matrix<double> matrixA, Vector<double> vectorB)

{

var res = Simplex.StartSimplexMethod(matrixA, vectorB);

var vectorX = Vector<double>.Build.DenseOfArray(res.Item1.ToArray());

var dirtBasis = res.Item2;

// Проверяем, все ли числа в векторе целые

if (vectorX.All(d => Math.Abs(d % 1) <= (Double.Epsilon \* 100)))

{

return vectorX.ToList();

}

var basis = dirtBasis.Select(x => x - 1);

var bruh = new Dictionary<int, double>();

// Заполняем словарь(ключ = i, значение = vectorX[i])

for (int i = 0; i < vectorX.Count; i++)

{

bruh.Add(i, vectorX[i]);

}

var notBasis = bruh.Keys.Where(i => !basis.Contains(i));

var matrixAb = Matrix<double>.Build.DenseOfColumns(basis.Select(i => matrixA.Column((int)i)));

var matrixAn = Matrix<double>.Build.DenseOfColumns(notBasis.Select(i => matrixA.Column(i)));

var multipleRes = matrixAb.Inverse() \* matrixAn;

// Находим индекс нецелого числа в vectorX

int index = vectorX.ToList().FindIndex(x => !(Math.Abs(x % 1) <= (Double.Epsilon \* 100)));

double k = basis.ElementAt(index);

var fractions = multipleRes.Row((int)k).Select(GetFractionPart);

double[] arr = new double[vectorX.Count];

for (int i = 0; i < arr.Length; i++)

{

int ind = notBasis.ToList().IndexOf(i);

if (ind != -1)

{

arr[i] = fractions.ElementAt(ind);

}

else

{

arr[i] = 0.0;

}

}

double fractionPart = GetFractionPart(vectorX[(int)k]);

var resVector = arr.Append(-1.0).Append(fractionPart).ToList();

return resVector;

}

}