Министерство образования Республики Беларусь  
Учреждения образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе 4

студент гр. 853504  
Лукашёнок В.Д.

Минск 2021

**Лабораторная работа 4. Вариант 19.**

Конспектировал на лабораторной, совершаю действие в действие, как было продемонстрировано на лр. В файле 4\_1.go просто лежат функции считывания задачи и умножения вектора на матрицу (Да, в gonum такой функции нет).

**Код программы**

4.go

package main

import (

"fmt"

"math"

"gonum.org/v1/gonum/mat"

)

func doubleSimplexMethod(scalesVector \*mat.VecDense, conditionsMatrix \*mat.Dense, freeVector, baselineIndexes, yVector \*mat.VecDense) (\*mat.VecDense, \*mat.VecDense) {

fmt.Println("New iteration")

// conditionsNumber - rows, varNumber - columns

conditionsNumber, varNumber := conditionsMatrix.Dims()

nonBaseLineIndexes, j := mat.NewVecDense(varNumber-conditionsNumber, nil), 0

for i := 0; i < varNumber; i++ {

if !Find(RawVector(baselineIndexes), float64(i)) {

nonBaseLineIndexes.SetVec(j, float64(i))

j++

}

}

// BaselineVector and baselinematrix

baselineMatrix := mat.NewDense(conditionsNumber, conditionsNumber, nil)

baselineVector := mat.NewVecDense(conditionsNumber, nil)

for i := 0; i < conditionsNumber; i++ {

baselineMatrix.SetCol(i, RawVector(conditionsMatrix.ColView(int(baselineIndexes.AtVec(i)))))

baselineVector.SetVec(i, scalesVector.AtVec(int(baselineIndexes.AtVec(i))))

}

baselineMatrixInv := mat.NewDense(conditionsNumber, conditionsNumber, nil)

baselineMatrixInv.Inverse(baselineMatrix)

// Vector Kappa

baselineKappa := mat.NewVecDense(conditionsNumber, nil)

baselineKappa.MulVec(baselineMatrixInv, freeVector)

kappa := mat.NewVecDense(varNumber, nil)

for i := 0; i < conditionsNumber; i++ {

kappa.SetVec(int(baselineIndexes.AtVec(i)), baselineKappa.AtVec(i))

}

// Checking if kappa is optimal case

isOptimalCase, negativeBaselineIndex := true, -1

for i := 0; i < conditionsNumber; i++ {

if baselineKappa.AtVec(i) < 0 {

isOptimalCase = false

negativeBaselineIndex = i

break // if break is commented, last negative value will be observed, otherwise first

}

}

if isOptimalCase {

fmt.Println("current kappa is positive everywhere, end.")

matPrint(kappa)

return kappa, baselineIndexes

} else {

fmt.Println("current kappa is not positive everywhere")

matPrint(kappa)

}

if yVector == nil {

yVector = vecMulMat(baselineVector, baselineMatrixInv)

}

// y Deltavector is row with index of negative kappa value

yDeltaVector := mat.NewVecDense(conditionsNumber, baselineMatrixInv.RawRowView(negativeBaselineIndex))

fmt.Println("yDeltaVector")

matPrint(yDeltaVector)

muList := make([]float64, varNumber-conditionsNumber)

// for nonbaseline indexes

for i := 0; i < varNumber-conditionsNumber; i++ {

mu := mat.VecDenseCopyOf(yDeltaVector)

muList[i] = mat.Dot(mu, conditionsMatrix.ColView(int(nonBaseLineIndexes.AtVec(i))))

}

// If there's nothing to change, problem is not consistent

isConsistent := false

for i := 0; i < varNumber-conditionsNumber; i++ {

if muList[i] < 0 {

isConsistent = true

break

}

}

if !isConsistent {

panic("Problem is not consistent")

}

// Finding min sigma and its index

minSigma, minSigmaIndex := math.Inf(1), -1

for i := 0; i < varNumber-conditionsNumber; i++ {

if muList[i] < 0 {

currentNonBaselineIndex := int(nonBaseLineIndexes.AtVec(i))

Cj := scalesVector.AtVec(currentNonBaselineIndex)

Aj := mat.Dot(conditionsMatrix.ColView(currentNonBaselineIndex), yVector)

muj := muList[i]

currentSigma := (Cj - Aj) / muj

if currentSigma < minSigma {

minSigma, minSigmaIndex = currentSigma, i

}

} else {

muList[i] = math.Inf(1)

}

}

// Changing dual plan (baseline indexes)

newBaselineIndexes := mat.VecDenseCopyOf(baselineIndexes)

newBaselineIndexes.SetVec(negativeBaselineIndex, float64(minSigmaIndex))

fmt.Println("newBaselineIndexes")

matPrint(newBaselineIndexes)

yDeltaVector.ScaleVec(minSigma, yDeltaVector)

// Updating y vector by adding y vector and scaled y delta vector

yVector.AddVec(yVector, yDeltaVector)

// Next iteration

return doubleSimplexMethod(scalesVector, conditionsMatrix, freeVector, newBaselineIndexes, yVector)

}

func main() {

// Reading problem

scalesVector, conditionsMatrix, freeVector, baselineIndexes := readDoubleOptimizationProblem("input.txt", 4, 2)

// shifting all values by one cause of test cases

for i := 0; i < baselineIndexes.Len(); i++ {

baselineIndexes.SetVec(i, baselineIndexes.AtVec(i)-1)

}

// Solving problem

optimalPlan, baselineIndexes := doubleSimplexMethod(scalesVector, conditionsMatrix, freeVector, baselineIndexes, nil)

fmt.Printf("Result:\n")

matPrint(optimalPlan)

matPrint(baselineIndexes)

}

4\_1.go

package main

import (

"io/ioutil"

"strings"

"gonum.org/v1/gonum/mat"

)

func vecMulMat(vec \*mat.VecDense, matrix \*mat.Dense) \*mat.VecDense {

vectorPre := mat.NewDense(vec.Len(), vec.Len(), nil)

vectorPre.SetRow(0, RawVector(vec))

vectorPre.Mul(vectorPre, matrix)

return mat.VecDenseCopyOf(vectorPre.ColView(0))

}

func readDoubleOptimizationProblem(input string, varNumber, conditionsNumber int) (\*mat.VecDense, \*mat.Dense, \*mat.VecDense, \*mat.VecDense) {

str, err := ioutil.ReadFile(input)

if err != nil {

panic(err)

}

lines := strings.Split(string(str), "\n")

scalesVector, freeVector, baselineIndexes := mat.NewVecDense(varNumber, nil), mat.NewVecDense(varNumber, nil), mat.NewVecDense(varNumber, nil)

lines, scalesVector = readVector(lines, varNumber)

conditionsMatrix := mat.NewDense(conditionsNumber, varNumber, nil)

for i := 0; i < conditionsNumber; i++ {

condition := mat.NewVecDense(varNumber, nil)

lines, condition = readVector(lines, varNumber)

conditionsMatrix.SetRow(i, RawVector(condition))

}

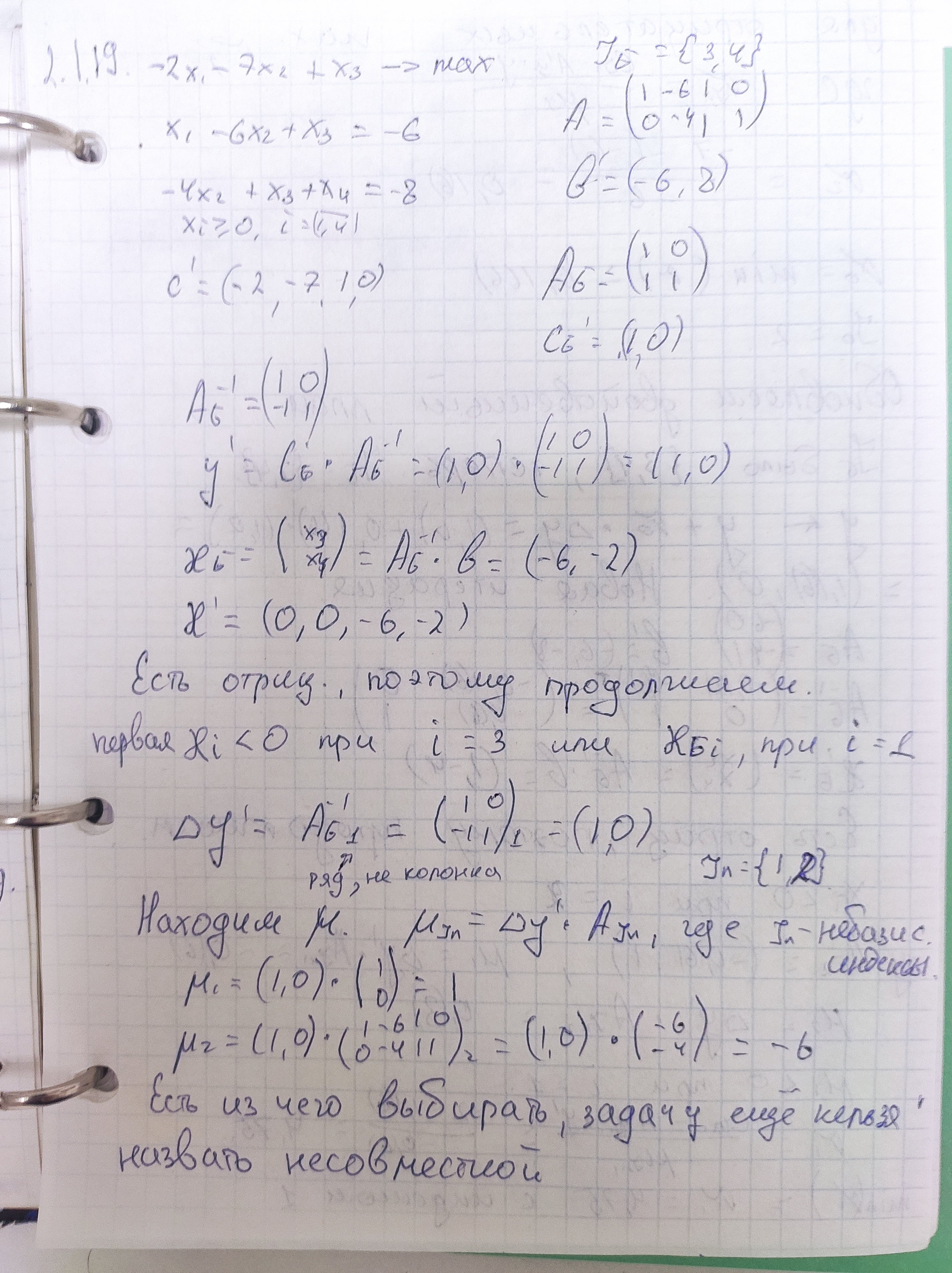
lines, freeVector = readVector(lines, conditionsNumber)

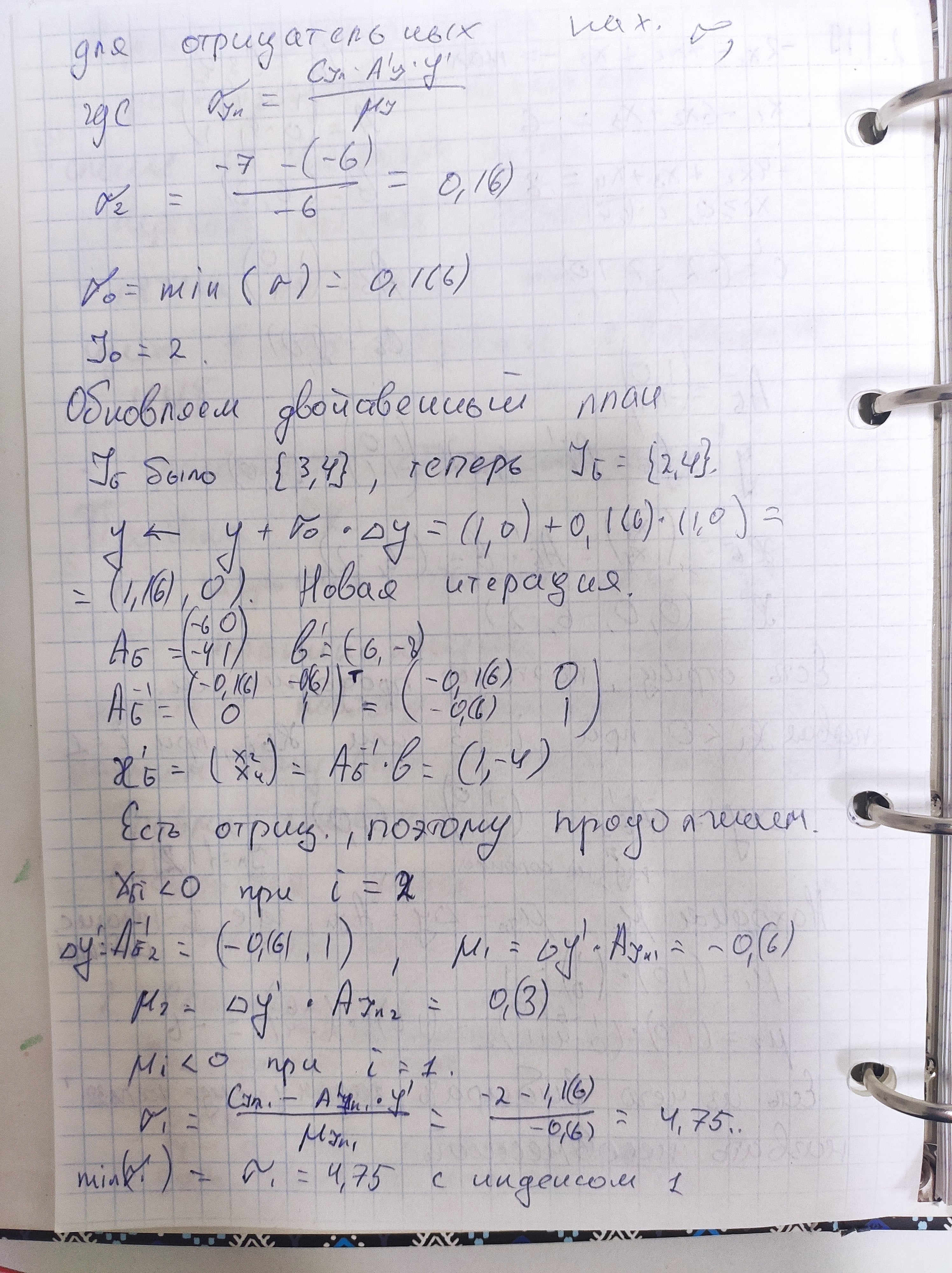
lines, baselineIndexes = readVector(lines, conditionsNumber)

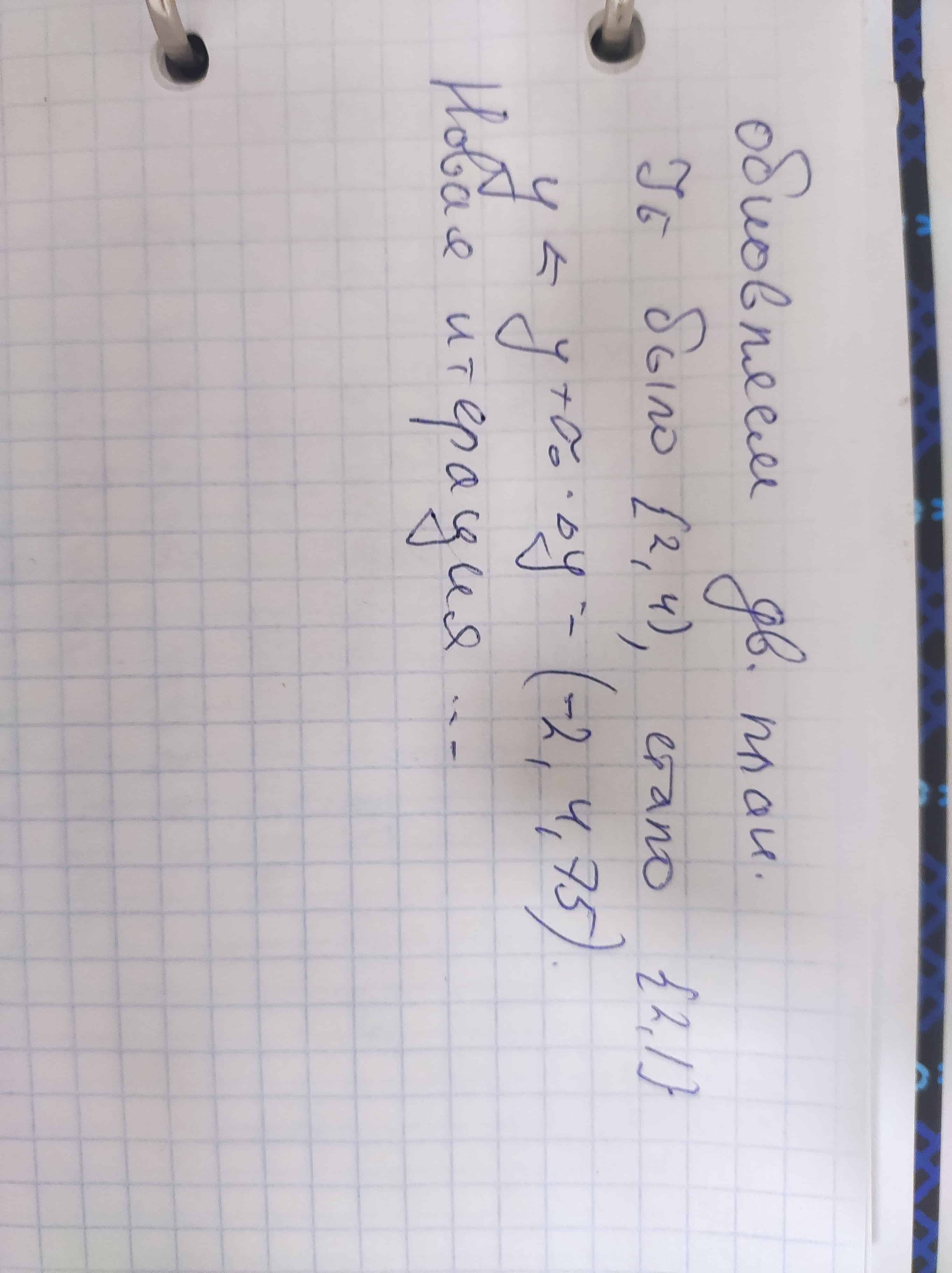
return scalesVector, conditionsMatrix, freeVector, baselineIndexes

}

**Первые итерации**

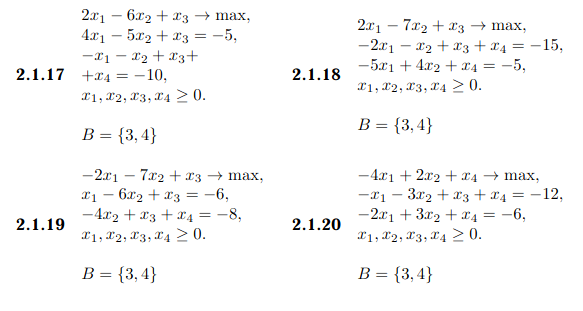
****

****

****

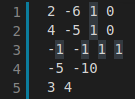
**Демонстрация работы (Вариант 19)**

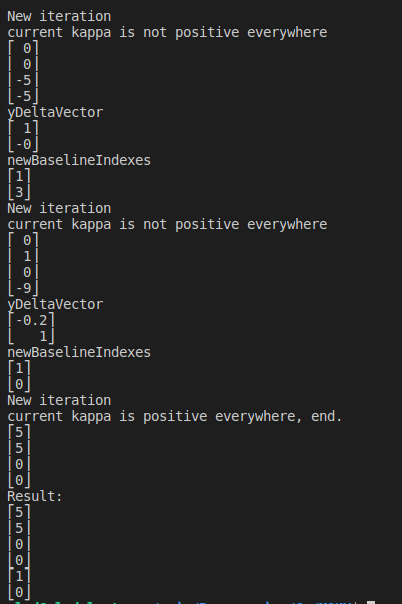
Проверять очень просто, поэтому возьмём несколько примеров (17 18 19 20 варианты к примеру).

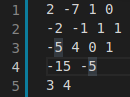
****

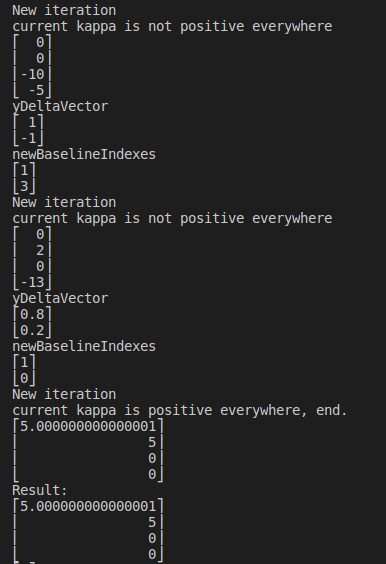
****

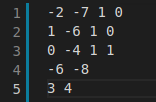
Далее просто пары картинок, где на первой — коэффициенты примера, на второй — консольный вывод.

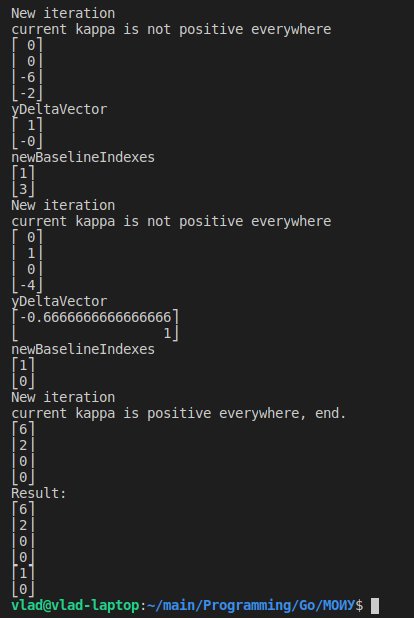
****

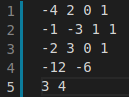
****

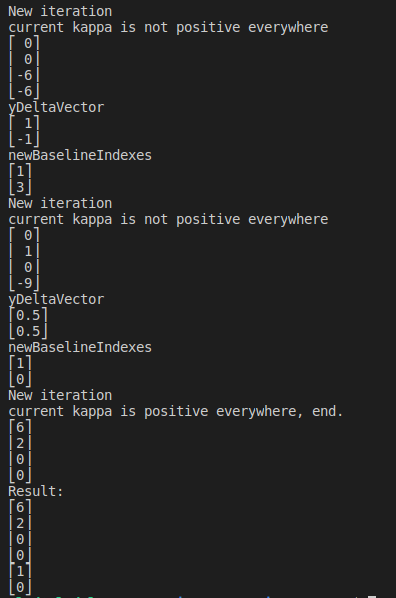
****

****

****

****

****

****