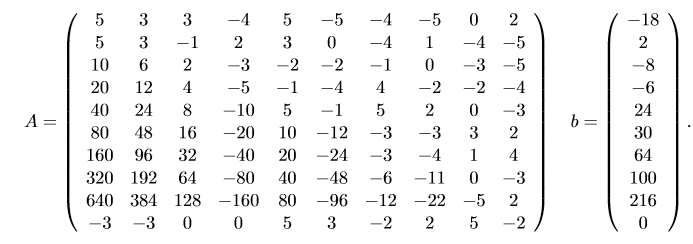
Вариант 11

Задание:

1. Написать программу, которая решает задачи наименьших квадратов вида ||Akx=b||-->min методом нормальных уравнений с использованием разложения Холецкого

Ak – матрица составленная из первых k столбцов матрицы А, k= 1, 2, … n



1. Сравнить полученные результаты с результатами работы функции scipy.linalg.lstsq, сравнить нормы погрешности.
2. Построить график нормы невязки в зависимости от k

Решение

1. Реализовал программу (исходный код привожу позже).

Для решения пользуюсь следующим алгоритмом. Используя исходные значения привожу их к следующему виду .

Где: A – исходная матрица

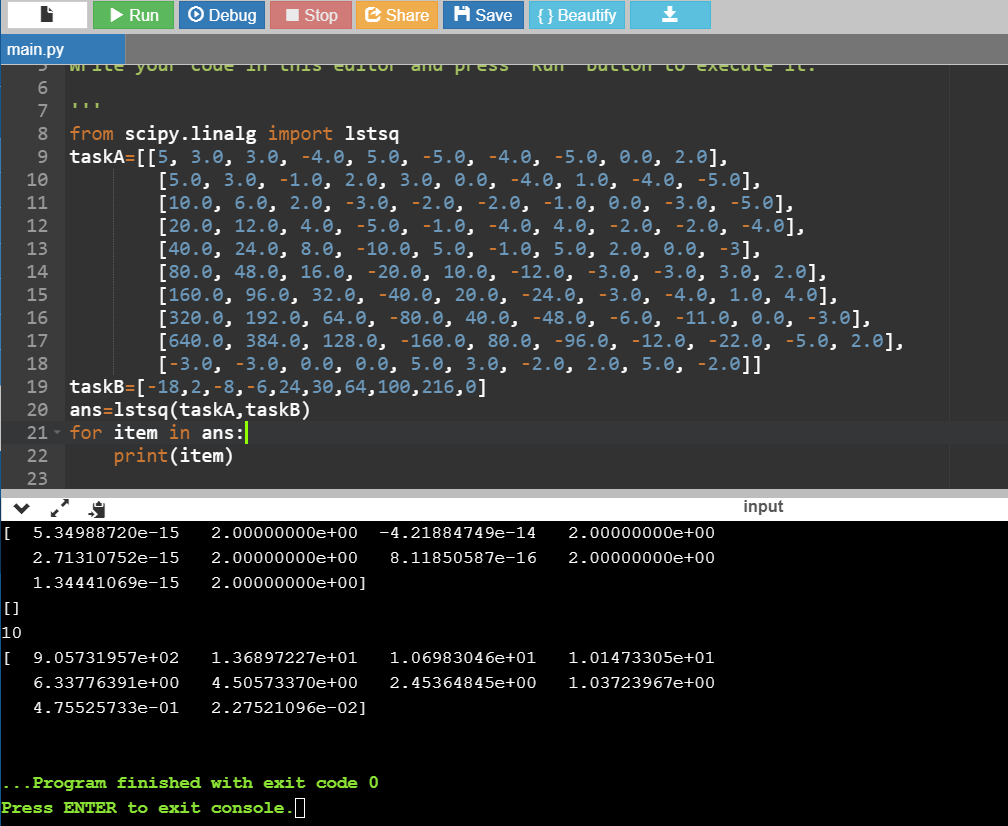
АТ – транспонированная исходная матрица.

x – столбец решений.

b – столбец значений линейных уравнений.

Таким образом сводя задачу к решению СЛАУ, а так как матрица является положительно определенной и симметричной, то для ее подсчета можно пользоваться разложением Холецкого.

1. Воспользовался функцией lstsq из библиотеки scipy.linalg. получил следующий вектор значений (исходный код привожу позже):



Результаты, полученные с помощью моей программы:

[-1.7796032289747514E-10,

1.9999999999251348,

5.2496650551918246E-9,

2.0000000030394776,

-3.502314025054066E-10,

2.0000000001497145,

-2.2582926606028724E-10,

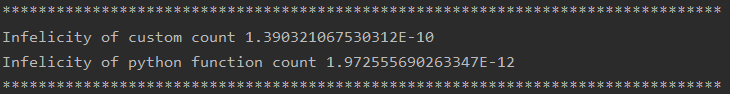
2.0000000001609832,

-5.238636197547992E-11,

1.999999999981236]

Далее, нахожу погрешность вычислений используя формулу

Получаю следующие значения:



На основании чего можно сказать, что результат вычисления с использованием функции lstsq, точнее, чем результат вычисления функции, которую написал я.

3)Вычисляю норму невязку по формуле

Результаты вычисления:

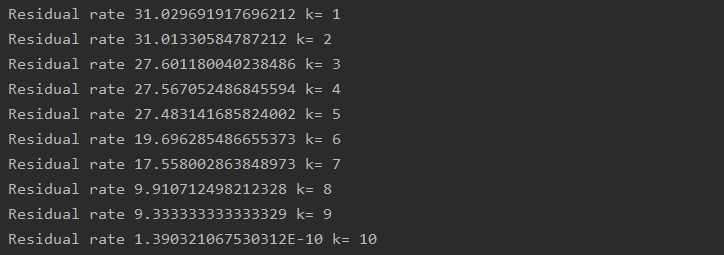
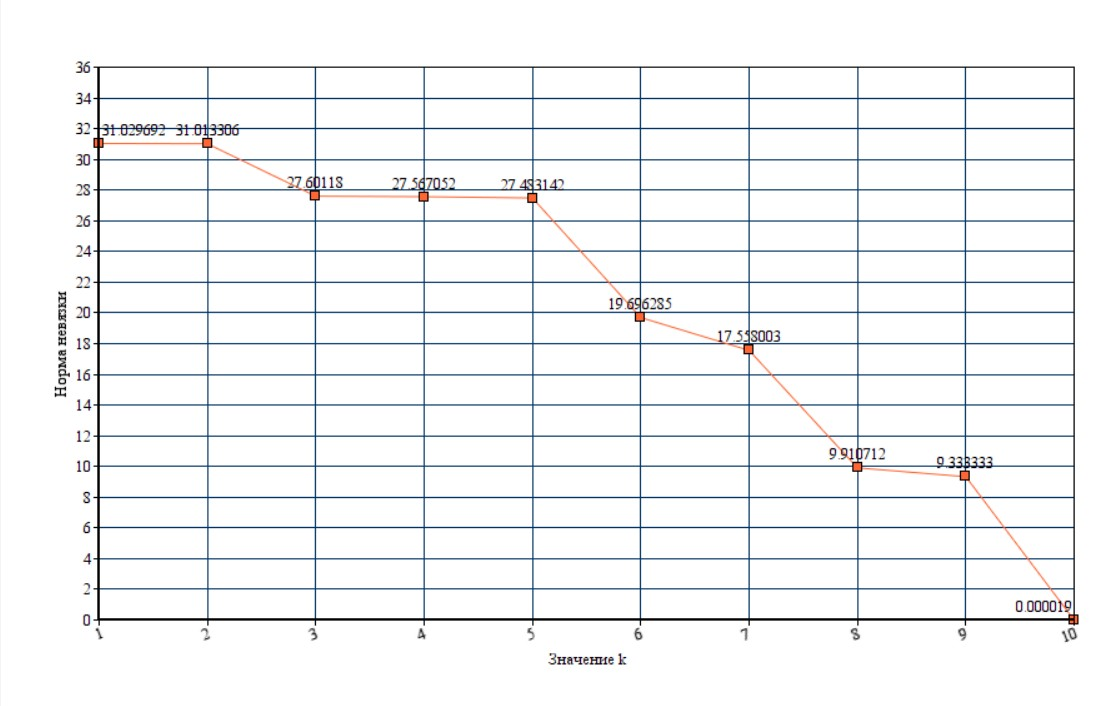


График полученный по результатам работы программы имеет вид:



Код программы на Python:

from scipy.linalg import lstsq

taskA=[[5, 3.0, 3.0, -4.0, 5.0, -5.0, -4.0, -5.0, 0.0, 2.0],

[5.0, 3.0, -1.0, 2.0, 3.0, 0.0, -4.0, 1.0, -4.0, -5.0],

[10.0, 6.0, 2.0, -3.0, -2.0, -2.0, -1.0, 0.0, -3.0, -5.0],

[20.0, 12.0, 4.0, -5.0, -1.0, -4.0, 4.0, -2.0, -2.0, -4.0],

[40.0, 24.0, 8.0, -10.0, 5.0, -1.0, 5.0, 2.0, 0.0, -3],

[80.0, 48.0, 16.0, -20.0, 10.0, -12.0, -3.0, -3.0, 3.0, 2.0],

[160.0, 96.0, 32.0, -40.0, 20.0, -24.0, -3.0, -4.0, 1.0, 4.0],

[320.0, 192.0, 64.0, -80.0, 40.0, -48.0, -6.0, -11.0, 0.0, -3.0],

[640.0, 384.0, 128.0, -160.0, 80.0, -96.0, -12.0, -22.0, -5.0, 2.0],

[-3.0, -3.0, 0.0, 0.0, 5.0, 3.0, -2.0, 2.0, 5.0, -2.0]]

taskB=[-18,2,-8,-6,24,30,64,100,216,0]

ans=lstsq(taskA,taskB)

for item in ans:

print(item)

Код программы на Java (Выборочно):

Метод main:

public static void main(String[] args) {  
 for(int k=1; k<11;k++){  
 Matrix a = *cropColumns*(*TASK\_A*,k);  
 Matrix ta = Matrix.*transposeMatrix*(a);  
 Matrix b = new Matrix(*TASK\_B*);  
  
 var newA = Matrix.*multiplyMatrix*(ta, a);  
 var newB = Matrix.*multiplyMatrix*(ta, b);  
  
 Matrix aDecomposed = Matrix.*choleskyDecomposition*(newA);  
 //System.out.println(aDecomposed);  
 var ans = Matrix.*solveCholesky*(aDecomposed, newB);  
  
  
 System.*out*.println("Ans:");  
 System.*out*.println(Arrays.*toString*(ans));  
  
  
 var customInfelicity = new double[a.getI()];  
 for (int i = 0; i < a.getI(); i++) {  
 double sum = 0;  
 for (int j = 0; j < a.getJ(); j++) {  
 sum += a.matrix[i][j] \* ans[j];  
 }  
 customInfelicity[i] = sum;  
 }  
  
 var inf=Matrix.*countInfelicity*(customInfelicity, *convertMatrixToVector*(b));  
 System.*out*.println("Residual rate "+ inf+" k= "+k);  
   
 if(a.getJ()==10){  
 System.*out*.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 inf=Matrix.*countInfelicity*(customInfelicity, *convertMatrixToVector*(b));  
 System.*out*.println("Infelicity of custom count "+ inf);  
 var pyMatrix = new Matrix(*PYTHON\_ANSWER*);  
 inf= Matrix.*countInfelicity*(*convertMatrixToVector*(Matrix.*multiplyMatrix*(a, pyMatrix)),*convertMatrixToVector*(b));  
 System.*out*.println( "Infelicity of python function count "+ inf);  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  
 }  
 }  
}

Класс Matrix:

package com.company;  
  
import java.util.Arrays;  
  
public class Matrix {  
 double[][] matrix;  
  
 Matrix(double[][] matrix) {  
 this.matrix = matrix;  
 }  
  
 static Matrix transposeMatrix(Matrix a) {  
 var ans = new double[a.matrix[0].length][a.matrix.length];  
  
 for (int i = 0; i < a.getI(); i++) {  
 for (int j = 0; j < a.getJ(); j++) {  
 ans[j][i] = a.matrix[i][j];  
 }  
 }  
 return new Matrix(ans);  
 }  
  
 static Matrix choleskyDecomposition(Matrix a) {  
 var ans = new double[a.getI()][a.getJ()];  
 for (int i = 0; i < a.getI(); i++)  
 {  
 double temp;  
 for (int j = 0; j < i; j++)  
 {  
 temp = 0;  
 for (int k = 0; k < j; k++)  
 {  
 temp += ans[i][k] \* ans[j][k];  
 }  
 ans[i][j] = (a.matrix[i][j] - temp) / ans[j][j];  
 }  
 temp = a.matrix[i][i];  
 for (int k = 0; k < i; k++)  
 {  
 temp -= ans[i][k] \* ans[i][k];  
 }  
 ans[i][i] = Math.*sqrt*(temp);  
 }  
 return new Matrix(ans);  
 }  
  
 static double[] solveCholesky(Matrix u, Matrix b) {  
 var y = *solveStraightCholesky*(u, b);  
 return *solveTransposedCholesky*(*transposeMatrix*(u), y);  
 }  
  
 private static double[] solveStraightCholesky(Matrix u, Matrix b) {  
 var ans = new double[u.getI()];  
 for (int i = 0; i < u.getI(); i++) {  
 double sum = 0.0;  
 for (int j = 0; j < i; j++) {  
 sum += u.matrix[i][j] \* ans[j];  
 }  
 ans[i] = (b.matrix[i][0] - sum) / u.matrix[i][i];  
 }  
 return ans;  
 }  
  
 private static double[] solveTransposedCholesky(Matrix u, double[] y) {  
 var ans = new double[u.getI()];  
 for (int i = u.getI() - 1; i >= 0; i--) {  
 double sum = 0.0;  
 for (int j = u.getJ() - 1; j > i; j--) {  
 sum += u.matrix[i][j] \* ans[j];  
 }  
 ans[i] = (y[i] - sum) / u.matrix[i][i];  
 }  
 return ans;  
 }  
  
  
 static Matrix multiplyMatrix(Matrix a, Matrix b) {  
 if (a.getJ() != b.getI())  
 throw new RuntimeException("Matrix can't be multiply (sizes don't match)");  
  
 double[][] ans = new double[a.getI()][b.getJ()];  
  
  
 for (int i = 0; i < a.getI(); i++) {  
 for (int j = 0; j < b.getJ(); j++) {  
 for (int k = 0; k < b.getI(); k++) {  
 ans[i][j] += a.matrix[i][k] \* b.matrix[k][j];  
 }  
 }  
 }  
  
 return new Matrix(ans);  
 }  
  
 static double countInfelicity(double[] ax, double[] bx ){  
 if(ax.length!=bx.length)  
 throw new RuntimeException("ax.length!= bx.length");  
 double sum=0;  
 for(int i=0;i<ax.length;i++){  
 sum+=(ax[i]-bx[i])\*(ax[i]-bx[i]);  
 }  
 return Math.*sqrt*(sum);  
 }  
  
 public int getI() {  
 return matrix.length;  
 }  
  
 public int getJ() {  
 return matrix[0].length;  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder ans = new StringBuilder();  
 ans.append("Matrix{\n");  
 for (var i : matrix)  
 ans.append("\t").append(Arrays.*toString*(i)).append("\n");  
 ans.append('}');  
 return ans.toString();  
 }  
}

Результаты работы программы:

