МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УКРАЇНСЬКИЙ ДЕРЖАВНИЙ ХІМIКО-ТЕХНОЛОГIЧНИЙ

УНІВЕРСИТЕТ»

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине «Моделирование систем» на тему:

«Регрессионный анализ: парный и множественный»

Виконав:

Студент групи 3-ІС-21, Квасенко Е. О.

Перевірила:

Доцень кафедри ІС, Коротка Л.І.

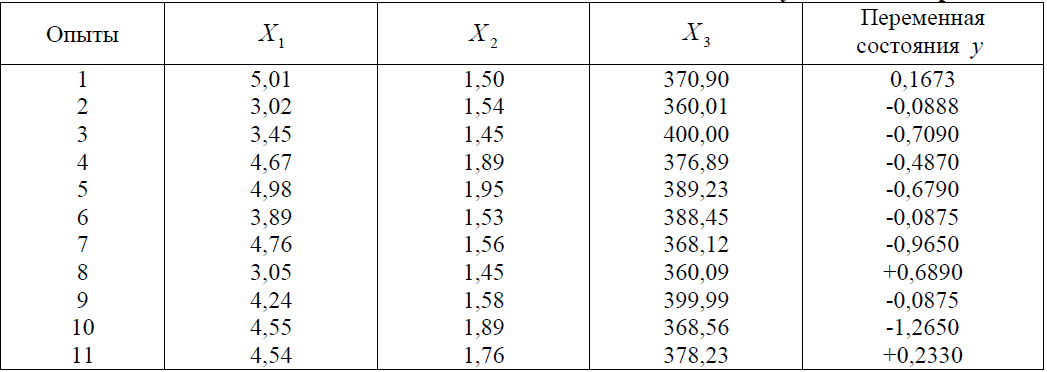
Дніпро, 2018

Цель лабораторной работы: изучить основные понятия корреляционно-

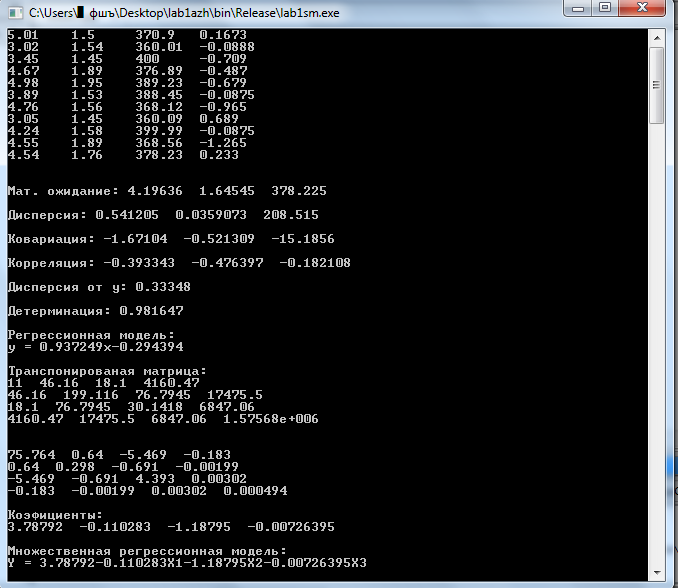
регрессионного анализа. По результатам экспериментов в химической технологии научиться строить математические модели, позволяющие адекватно описывать исследуемые процессы.

Исходные данные:

Вариант 5



Результат работы программы:



Код программы:

#include <iostream>

#include <cmath>

#include <fstream>

using namespace std;

void gaussian(double (\*a)[4],int \*);

void invert(double (\*a)[4]);

int main()

{

double matrix[11][4];

double oneMatrix[11][4] = {0};

double transpMatrix[4][11];

double multMatrix[4][4] = {0};

double multMatrix\_2[4][1] = {0};

double invMatrix[4][4] = {0};

double coefs[4] = {0};

double temp = 0;

double expressY = 0;

double my = 0;

double tmpx[3] = {0},tmpy = 0;

double dispy = 0;

double lastDispY = 0;

double mx[3] = {0};

double corrX[3] = {0};

double dispX[3] = {0};

double coefB = 0,coefA = 0;

double determinate = 0;

ifstream fin;

fin.open("matrix.txt");

if(fin.is\_open())

{

for(int i=0;i<11;i++)

for(int j=0;j<4;j++)

fin >> matrix[i][j];

}

fin.close();

fin.open("oneMatrix.txt");

if(fin.is\_open())

{

for(int i=0;i<11;i++)

{

for(int j=0;j<4;j++)

{

fin >> oneMatrix[i][j];

}

}

}

fin.close();

fin.open("invMatrix.txt");

if(fin.is\_open())

{

for(int i=0;i<4;i++)

{

for(int j=0;j<4;j++)

{

fin >> invMatrix[i][j];

}

}

}

fin.close();

//M(y)

for(int i=0;i<11;i++)

{

my += matrix[i][3];

}

my = my/11;

//Show matrix

for(int i=0;i<11;i++)

{

for(int j=0;j<3;j++)

{

cout << matrix[i][j] << "\t";

}

cout << endl;

}

cout << endl << endl << "Mx: ";

//M(x)

for(int j=0;j<3;j++)

{

for(int i=0;i<11;i++)

{

mx[j] += matrix[i][j];

}

mx[j] = mx[j]/11;

cout << (mx[j]) << " ";

}

cout << endl << endl << "Dx^2: ";

//D(x)

for(int j=0;j<3;j++)

{

for(int i=0;i<11;i++)

{

dispX[j] += pow(matrix[i][j]-mx[j],2);

}

dispX[j] = dispX[j]/10;

cout << dispX[j] << " ";

}

cout << endl << endl << "Cov: ";

//Cov(xy)

for(int j=0;j<3;j++)

{

for(int i=0;i<11;i++)

{

temp += (matrix[i][j]-mx[j]) \* (matrix[i][3]-my);

}

cout << (temp) << " ";

temp = 0;

}

//Expression X

for(int j=0;j<3;j++)

{

for(int i=0;i<11;i++)

{

tmpx[j] += pow(matrix[i][j]-mx[j],2);

}

}

//Expression Y

for(int i=0;i<11;i++)

{

tmpy += pow(matrix[i][3] - my,2);

}

cout << endl << endl << "Corr: ";

//Correlation

for(int j=0;j<3;j++)

{

for(int i=0;i<11;i++)

{

corrX[j] += ((matrix[i][j]-mx[j]) \* (matrix[i][3]-my)/sqrt(tmpx[j]\*tmpy));

}

cout << corrX[j] << " ";

}

//D(y)

for(int i=0;i<11;i++)

{

dispy += pow(matrix[i][3] - my,2);

}

dispy = dispy/10;

cout << endl << endl << "Dy: " << dispy;

for(int i=0;i<11;i++)

{

expressY += (pow(matrix[i][3] - my,2));

}

coefB = corrX[0] \* (sqrt(expressY/12)/sqrt(dispX[0]));

coefA = my - coefB\*mx[0];

for(int i=0;i<11;i++)

{

temp += pow(matrix[i][3] - (coefA\*matrix[i][0]+coefB),2);

}

lastDispY = temp/10;

temp = 0;

cout << endl << endl;

determinate = 1 - (dispy/lastDispY);

cout << determinate << " ";

cout << "y = " << coefA << "x" << coefB << endl << endl;

//Transp

for (int i = 0; i < 11; i++) {

for (int j = 0; j < 4; j++) {

transpMatrix[j][i] = oneMatrix[i][j];

}

}

for (int i = 0; i < 4; i++) {

for (int k = 0; k < 11; k++) {

multMatrix\_2[i][0] += transpMatrix[i][k] \* matrix[k][3];

}

}

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

for (int k = 0; k < 11; k++) {

multMatrix[i][j] += transpMatrix[i][k] \* oneMatrix[k][j];

}

}

}

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

cout << multMatrix[i][j] << " ";

}

cout << endl;

}

invert(multMatrix);

std::cout << endl << endl;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

cout << invMatrix[i][j] << " ";

}

cout << endl;

}

cout << endl;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

coefs[i] += invMatrix[i][j] \* multMatrix\_2[j][0];

}

}

for(int i=0;i<4;i++)

cout << coefs[i] << " ";

cout << endl;

cout << "Y = " << coefs[0] << coefs[1] << "X1" << coefs[2] << "X2" << coefs[3] << "X3";

return 0;

}

void gaussian(double (\*a)[4],int \* index)

{

int n = 4;

double c[4] = {0};

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

index[i] = i;

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

{

double c1 = 0;

for (int j=0; j<n; ++j)

{

double c0 = std::abs(a[i][j]);

if (c0 > c1) c1 = c0;

}

c[i] = c1;

}

int k = 0;

for (int j=0; j<n-1; ++j)

{

double pi1 = 0;

for (int i=j; i<n; ++i)

{

double pi0 = std::abs(a[index[i]][j]);

pi0 /= c[index[i]];

if (pi0 > pi1)

{

pi1 = pi0;

k = i;

}

}

int itmp = index[j];

index[j] = index[k];

index[k] = itmp;

for (int i=j+1; i<n; ++i)

{

double pj = a[index[i]][j]/a[index[j]][j];

a[index[i]][j] = pj;

for (int l=j+1; l<n; ++l)

a[index[i]][l] -= pj\*a[index[j]][l];

}

}

}

void invert(double (\*a)[4])

{

int n = 4;

double x[n][n] = {0};

double b[n][n] = {0};

int index[n] = {0};

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

b[i][i] = 1;

gaussian(a, index);

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

for (int j=i+1; j<n; ++j)

for (int k=0; k<n; ++k)

b[index[j]][k] -= a[index[j]][i]\*b[index[i]][k];

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

{

x[n-1][i] = b[index[n-1]][i]/a[index[n-1]][n-1];

for (int j=n-2; j>=0; --j)

{

x[j][i] = b[index[j]][i];

for (int k=j+1; k<n; ++k)

{

x[j][i] -= a[index[j]][k]\*x[k][i];

}

x[j][i] /= a[index[j]][j];

}

}

}