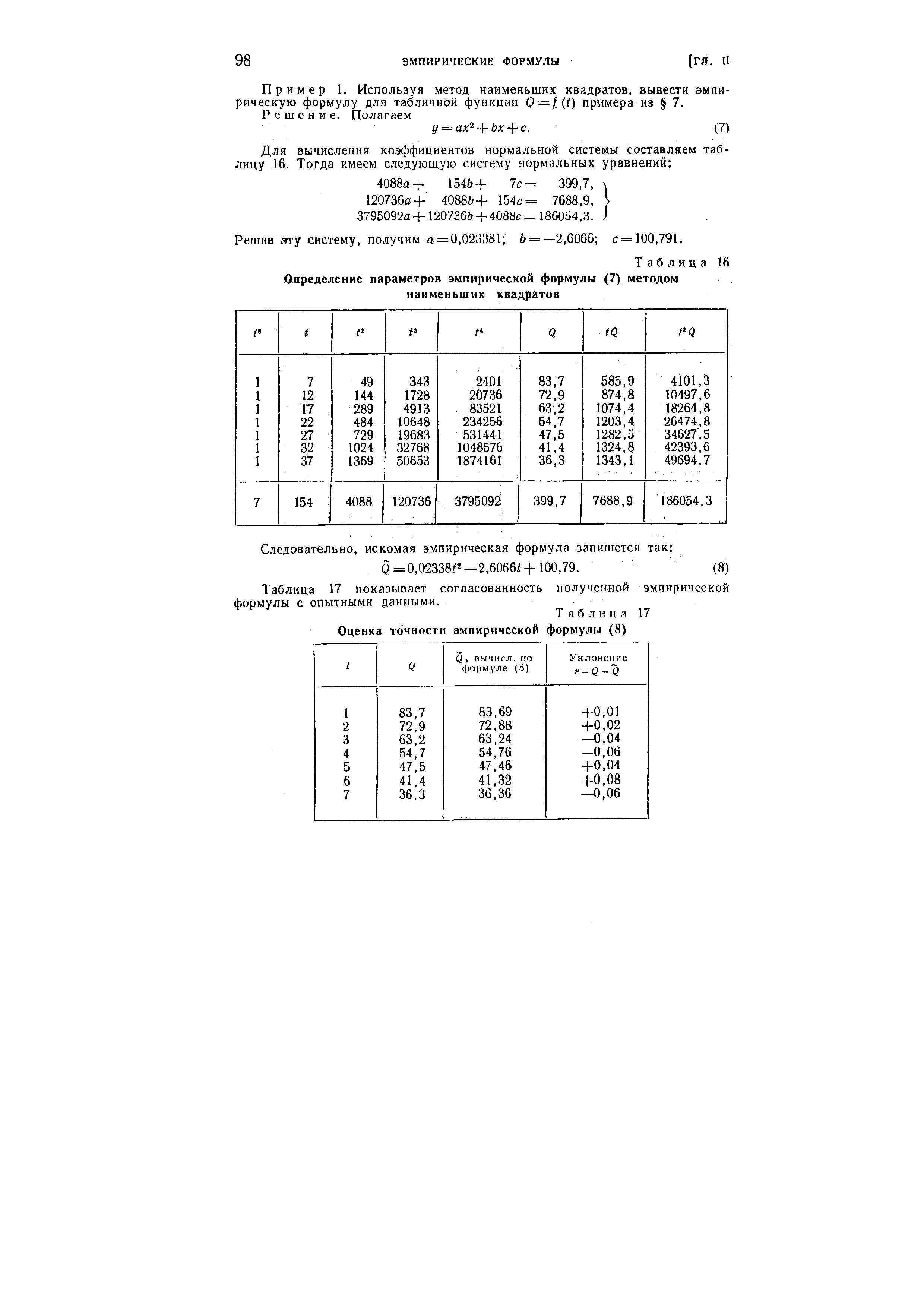
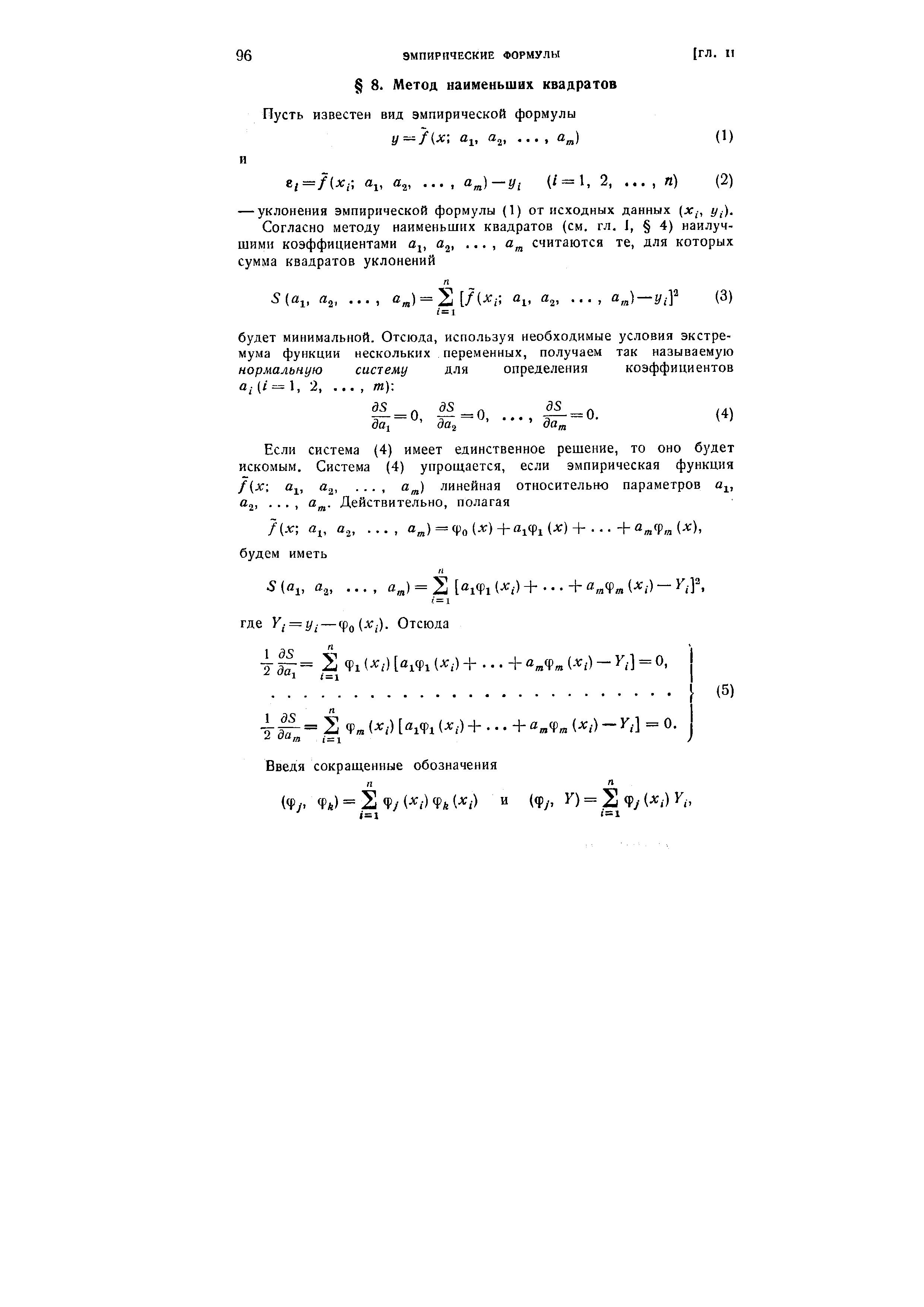
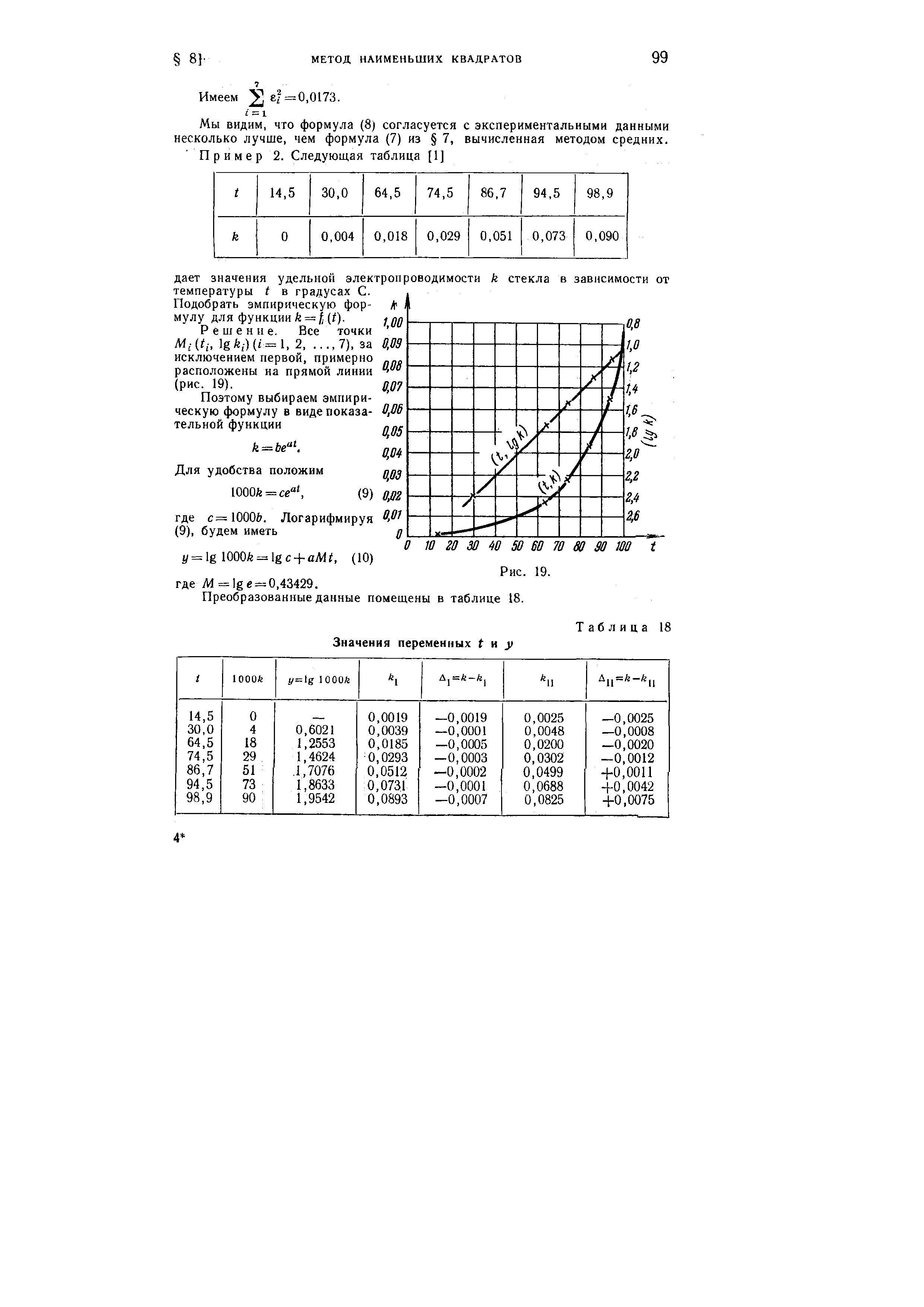
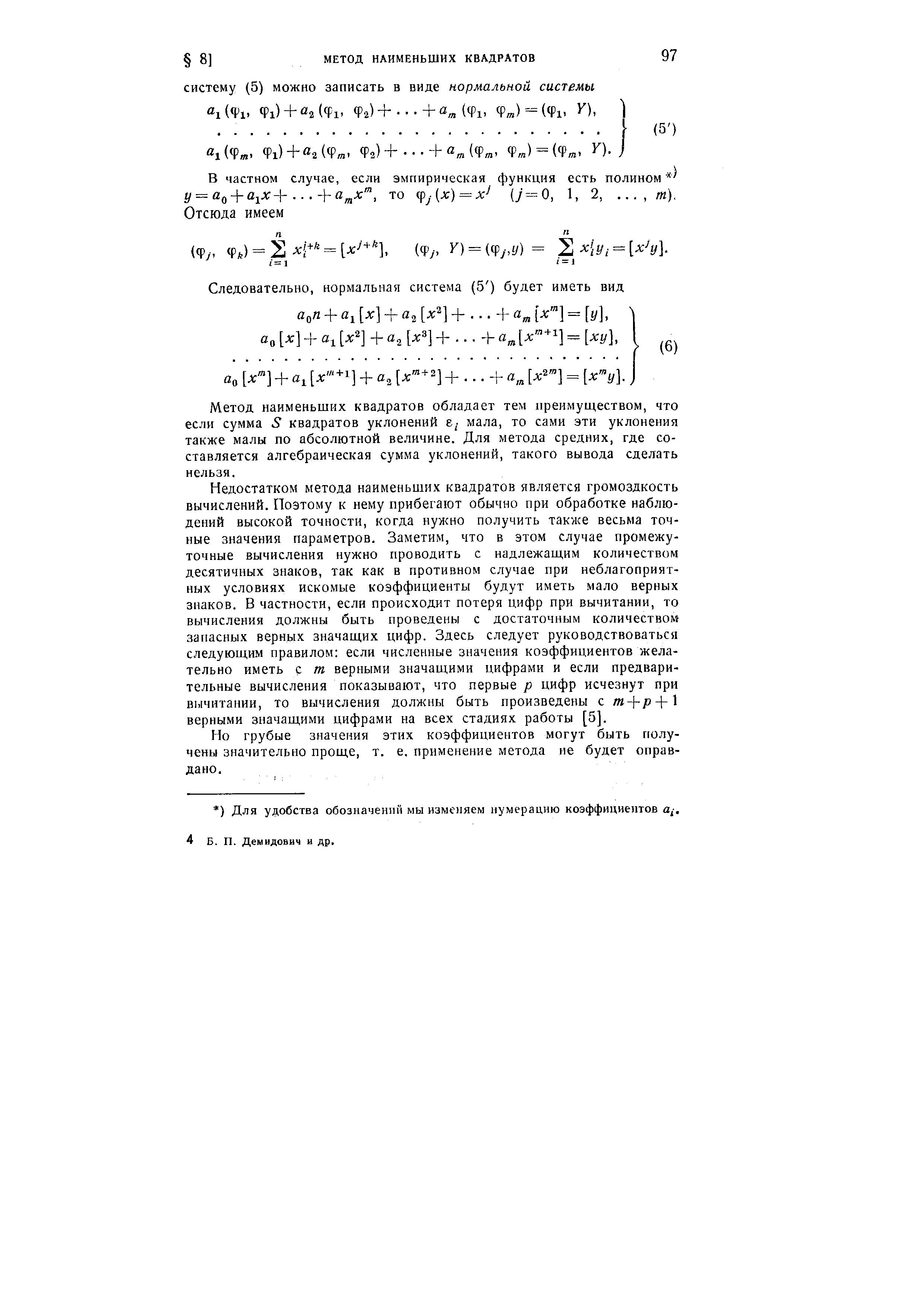
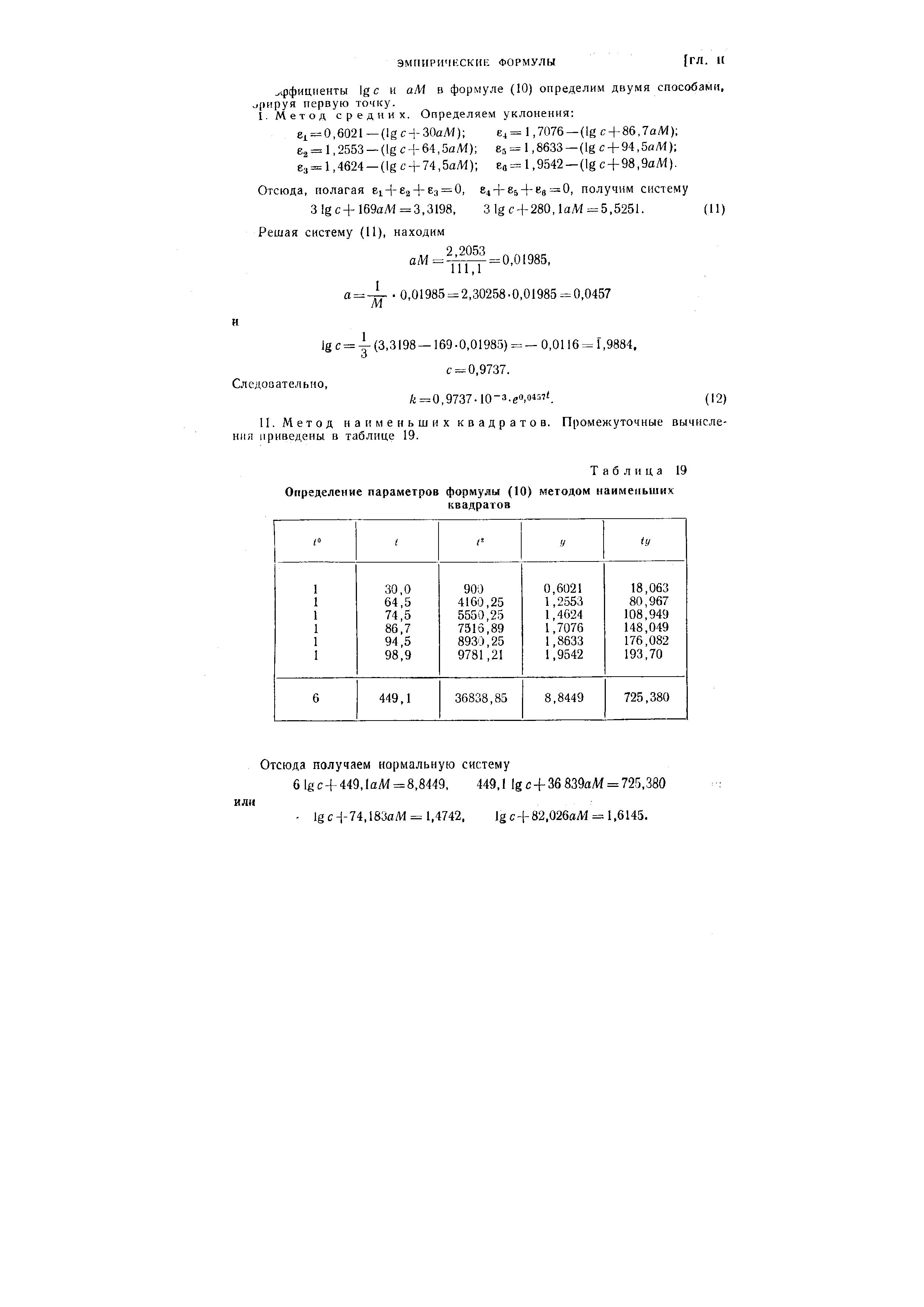
**Лабораторна робота №4**

**«Емпіричні методи»**

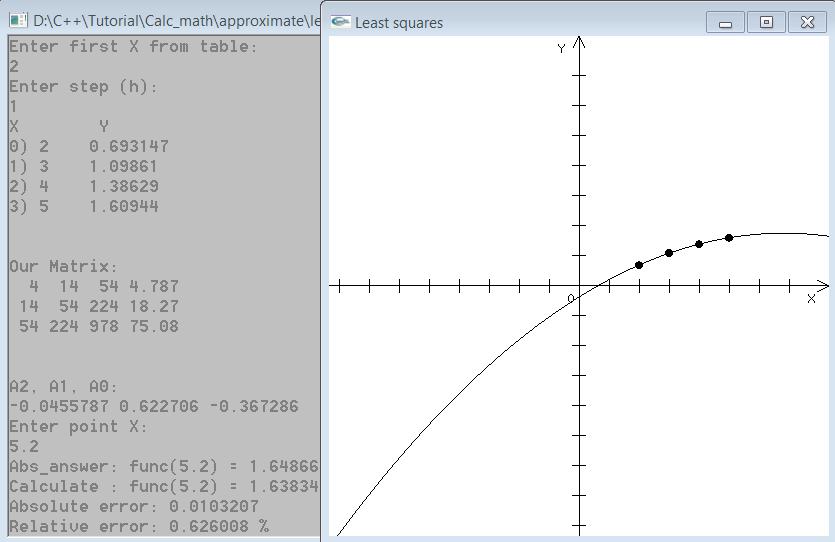
**1.Теоретична частина**



**2. Завдання**

Використовуючи метод найменших квадратів, вивести емпіричну формулу для заданої таблиці значень, а також знайти значення функції для введеного користувачем значення Х.

**3. Приклад роботи програми**

****

**4. Лістинг програми**

//метод наименьших квадратов

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

int main(int argc, char \*argv[]) {

float(\*func)(float) = log;

float h = 1;

float first = 2;

cout << "Enter first X from table:" << endl;

cin >> first;

cout << "Enter step (h):" << endl;

cin >> h;

///order of empiric function == y\_size

int x\_size = y\_size + 1; //extended matrix

float a[y\_size][x\_size];

//enter table of begin values: x, y

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

x\_t[i] = first + i\*h;

y[i] = func(x\_t[i]);} // Our function

cout << " X \t Y" << endl;

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

cout << i << ") " << x\_t[i] << "\t" << y[i] << endl; }

cout << endl;

//init array, set values = 0

for (int i = 0; i < y\_size; i++) {

for (int j = 0; j < y\_size; j++) {

a[i][j] = sigmaB(x\_t, N, i, j);} }

for (int i = 0; i < y\_size; i++) {

a[i][y\_size] = sigmaC(x\_t, y, N, i); }

//print matrix on the screen

cout << endl << "Our Matrix: " << endl;

for (int i=0; i<y\_size; i++) {

for (int j=0; j<y\_size+1; j++) {

cout << setw(3) << setprecision(4) << a[i][j] << " "; }

cout << endl; }

cout << endl;

///lets solve our system with Gauss method

//Straight Gauss

float mult;

for (int i=0;i<y\_size-1;i++)

for (int j=i+1;j<y\_size;j++) {

mult=a[i][i]/a[j][i];

for (int k=0;k<=y\_size;k++)

a[j][k]= a[j][k]\*mult-a[i][k]; }

//Inverse Gauss

//bottom last coeff

coeff[y\_size-1] = a[y\_size-1][y\_size] / a[y\_size-1][y\_size-1];

for (int i=y\_size-2;i>=0;i--) {

mult = 0;

for (int j=i+1;j<y\_size;j++)

mult+=a[i][j]\*coeff[j];

coeff[i]=(a[i][y\_size]-mult)/a[i][i]; }

cout << endl << "A2, A1, A0:" << endl;

for (int i = y\_size-1; i>=0; i--)

cout << setprecision(6) << coeff[i] << " ";

cout << endl;

cout << "Enter point X:" << endl;

float x\_pt = 5.2;

cin >> x\_pt;

float Y = formula(y\_size, coeff, x\_pt);//(x[2] \* x\_pt \* x\_pt) + (x[1] \* x\_pt) + x[0];

cout << "Abs\_answer: " << "func" << "(" << x\_pt << ") = " << func(x\_pt) << endl;

cout << "Calculate : " << "func" << "(" << x\_pt << ") = " << Y << endl;

double abs\_err = abs(Y - func(x\_pt));

cout << "Absolute error: " << abs\_err << endl;

cout << "Relative error: " << abs\_err / func(x\_pt) \* 100 << " %" << endl;

return 0;}

1. **Висновки**

Отже, я реалізував програму, за допомогою якої можна знаходити емпіричну формулу, яка задовольняє задані табличні дані, тобто різниця значень оригінальної функції та емпіричної прямує до нуля. Проаналізувавши точність апроксимації за допомогою многочлена, я знайшов сумарне квадратичне відхилення, яке становить 0.000045, що підтверджує точність даного методу.