4.7. SPLINE INTERPOLATION

Suppose on the interval $[a,b]$, there is a discrete set of mismatched points x_i, $i=0,...,n$, $x_0=a$, $x_n=b$ (interpolation nodes), at which the values of a certain function $y(x)$, generating the table $y_i=y(x_i)$, $i=0,...,n$, are known. It may be impractical to use a single interpolation formula (global interpolation) for a sufficiently large number of nodes $n>>1$. The obtained (by any method) interpolation polynomial $P_n(x)$ on the initial interval $[a,b]$ can exhibit its wave properties (that become higher, the higher the degree of the polynomial *n* is), which means that its values between the interpolation nodes can differ greatly from the values of the interpolated function $y(x)$.

Difficulties arising during polynomial interpolation are well illustrated by the Runge's phenomenon: interpolation on the interval $[-1;1]$ of the function $f(x)=1/(1+25x^2)$ in case of equally spaced distribution of nodes. With the order *n* of the interpolation polynomial increasing, its wave properties increase, and it leads to the interpolation error tending to infinity when $n\to\infty$.

One way to overcome this drawback is to use spline interpolation. The essence of spline interpolation is to determine the interpolating function using the formulas of the same type for various nonintersecting intervals of the initial interval $[a,b]$ and to match the values of the function and its derivatives at their boundaries. Splines have numerous uses both in mathematical theory and applied mathematics.

The most widely used case is when a polynomial of not higher than *m*-th degree is constructed between any two consecutive interpolation nodes of the initial interval $[a,b]$:

$$S_i(x)=\sum_{j=0}^{m}a_{ij}x^j,\ x_{i-1}\le x\le x_i,\ i=1,...,n.$$

Then, the piecewise-continuous on the interval $[a,b]$ interpolation polynomial $S_m(x)=\{S_i(x),\ i=1,...,n$, which at the interpolation nodes takes the values of the approximating function and is continuous with its $l<m$ derivatives, is

called an interpolation spline. Its coefficients are found from the equality conditions at the grid nodes x_i, $i=0,...,n$ of the spline values $S_i(x_i)$, $S_i(x_{i-1})$, $i=1,...,n$ and the values of the approximee function $y_i = y(x_i)$ and $y_{i-1} = y(x_{i-1})$ at the interval ends between every two successive nodes, as well as the equality at all the internal grid nodes x_i, $i=1,...,n-1$ of values of *l* derivatives of the corresponding polynomials on the left and right $S_i^{(k)}(x_i) = S_{i-1}^{(k)}(x_i)$, $k=1,...,l$.

Definition. The difference $m-l$ between the *degree* of the spline *m* and the order of the highest continuous derivative *l* (spline *smoothness*) is called a spline *defect*.

As an example, a continuous polyline is a spline of degree 1 and defect 1.

4.8. CUBIC SPLINE

In practice, the most widely used one is the interpolation spline of degree 3 and defect 1, which is conveniently presented as follows:

$$S_3(x) = \{S_i(x),\ i=1,...,n,$$

$$S_i(x) = a_i + b_i(x-x_{i-1}) + c_i(x-x_{i-1})^2 + d_i(x-x_{i-1})^3,\ x_{i-1} \le x \le x_i.$$

To set such a cubic spline, it is necessary to build *n* polynomials of the third degree, i.e. identify $4n$ of the unknown parameters a_i, b_i, c_i, d_i, $i=1,...,n$. These coefficients are sought from the required conditions of spline interpolation at the grid nodes x_i, $i=0,...,n$.

First, the conditions of equality of the values of the spline and the values of the approximee function at the grid nodes lead to the following $2n$ equations:

$$S(x_0) = a_1 = y_0$$
$$S(x_i) = a_i + b_i(x_i - x_{i-1}) + c_i(x_i - x_{i-1})^2 + d_i(x_i - x_{i-1})^3 = a_{i+1} = y_i,\ i=1,...,n-1.$$
$$S(x_n) = a_n + b_n(x_n - x_{n-1}) + c_n(x_n - x_{n-1})^2 + d_n(x_n - x_{n-1})^3 = y_n$$

Second, the conditions of equality at the internal nodes of the grid of the first and second derivatives of the corresponding cubic polynomials on the left and right lead to the following $2n-2$ equations:

$$\begin{aligned} &S'(x_i) = b_i + 2c_i(x_i - x_{i-1}) + 3d_i(x_i - x_{i-1})^2 = b_{i+1} \\ &S''(x_i) = 2c_i + 6d_i(x_i - x_{i-1}) = 2c_{i+1} \end{aligned}, \ i = 1,...,n-1.$$

Third, it is assumed that the splines have zero curvature at the ends of the interval $[a,b]$ (missing final 2 equations):

$$\begin{aligned} &S''(x_0) = 2c_1 = 0 \\ &S''(x_n) = 2c_n + 6d_n(x_n - x_{n-1}) = 0 \end{aligned}.$$

When these two conditions are used, the resulting spline is called natural. But in the general case, other conditions can be used to close the system (to compose all $4n$ equations).

If we introduce the notation $h_i = x_i - x_{i-1}$, $i = 1,...,n$, then the final system of equations will take the following form:

$$\begin{cases} a_i = y_{i-1}, \ i = 1,..,n \\ a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = y_i, \ i = 1,..,n \\ b_i + 2c_i h_i + 3d_i h_i^2 = b_{i+1}, \ i = 1,..,n-1 \\ c_1 = 0 \\ c_i + 3d_i h_i = c_{i+1}, \ i = 1,..,n-1 \\ c_n + 3d_n h_n = 0 \end{cases} \sim \begin{cases} a_i = y_{i-1}, \ i = 1,..,n \\ b_i h_i + c_i h_i^2 + d_i h_i^3 = y_i - y_{i-1}, \ i = 1,..,n \\ b_i + 2c_i h_i + 3d_i h_i^2 = b_{i+1}, \ i = 1,..,n-1 \\ d_i = \dfrac{c_{i+1} - c_i}{3h_i}, \ i = 1,..,n-1 \\ d_n = -\dfrac{c_n}{3h_n} \\ c_1 = 0 \end{cases}.$$

If we exclude a_i, b_i, d_i from the resulting system, then we can get a system of $n-1$ linear algebraic equations with respect to $c_i, i = 2,...,n$ with a tridiagonal matrix:

$$\begin{cases} 2(h_1 + h_2)c_2 + h_2 c_3 = 3\left(\dfrac{y_2 - y_1}{h_2} - \dfrac{y_1 - y_0}{h_1} \right) \\ h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3\left(\dfrac{y_i - y_{i-1}}{h_i} - \dfrac{y_{i-1} - y_{i-2}}{h_{i-1}} \right), \ i = 3,...,n-1. \\ h_{n-1}c_{n-1} + 2(h_{n-1} + h_n)c_n = 3\left(\dfrac{y_n - y_{n-1}}{h_n} - \dfrac{y_{n-1} - y_{n-2}}{h_{n-1}} \right) \end{cases}$$

Such a system can be effectively solved by the sweep method.

The remaining spline coefficients can then be recovered by the following formulas:

$$\begin{cases} a_i = y_{i-1},\ i = 1,..,n \\ b_i = \dfrac{y_i - y_{i-1}}{h_i} - \dfrac{h_i}{3}(c_{i+1} + 2c_i),\ i = 1,...,n-1, \\ b_n = \dfrac{y_n - y_{n-1}}{h_n} - \dfrac{h_n}{3}2c_n \end{cases} \begin{cases} d_i = \dfrac{c_{i+1} - c_i}{3h_i},\ i = 1,...,n-1 \\ d_n = -\dfrac{c_n}{3h_n} \\ c_1 = 0 \end{cases}.$$

As a result, the desired cubic spline of defect 1 is built.

The spline interpolation method leads to satisfactory results when interpolating continuous functions with smooth first and second derivatives. Moreover, the cubic interpolation spline built through the nodes $y_i = y(x_i)$, $i = 0,...,n$ will have a minimum integral curvature along the entire interval $[a,b]$ as compared to any other interpolation function that has continuous first and second derivatives. But in case of spline interpolation of functions with a sharp change in derivatives, the resulting splines can have large errors. Splines of an order higher that the third are rarely used, as when calculating a large number of coefficients, errors can accumulate, resulting in significant inaccuracy.

Example. Using a table of values of the function $y(x)$ (values y_i calculated at points x_i, $i = 0,...,n$), construct a cubic spline interpolation polynomial passing through points (x_i, y_i), assuming that the spline has zero curvature at the borders. Calculate the value of the interpolation error at point x^*.

$y = x\ln(x)$, $n = 5$, $i = 0,...,5$, $x_i = 0.1, 0.5, 0.9, 1.3, 1.7, 2.1$, $x^* = 0.7$.

The desired cubic spline can be written as follows:

$S_3(x) = \{S_i(x)$, $i = 1,...,5$.

On each interval $[x_{i-1}, x_i]$, the spline is defined by the function:

$S_i(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3$, $x_{i-1} \le x \le x_i$, $i = 1,...,5$.

Let's write the system of equations for cubic spline with respect to $c_i, i = 2,...,5$ ($c_1 = 0$) considering that $h_i = x_i - x_{i-1} = h = 0.4$, $i = 1,...,5$:

$$\begin{cases} 4hc_2 + hc_3 = \frac{3}{h}(y_2 - 2y_1 + y_0) \\ hc_2 + 4hc_3 + hc_4 = \frac{3}{h}(y_3 - 2y_2 + y_1) \\ hc_3 + 4hc_4 + hc_5 = \frac{3}{h}(y_4 - 2y_3 + y_2) \\ hc_4 + 4hc_5 = \frac{3}{h}(y_5 - 2y_4 + y_3) \end{cases}, \quad \begin{cases} 1.6c_2 + 0.4c_3 = 7.5(y_2 - 2y_1 + y_0) \\ 0.4c_2 + 1.6c_3 + 0.4c_4 = 7.5(y_3 - 2y_2 + y_1) \\ 0.4c_3 + 1.6c_4 + 0.4c_5 = 7.5(y_4 - 2y_3 + y_2) \\ 0.4c_4 + 1.6c_5 = 7.5(y_5 - 2y_4 + y_3) \end{cases}.$$

Having solved this system with the sweep method, we find c_2, c_3, c_4, c_5.

Then, let's use the formulas for the remaining coefficients:

$a_i = y_{i-1}$, $b_i = \frac{y_i - y_{i-1}}{h} - \frac{h}{3}(c_{i+1} + 2c_i)$, $d_i = \frac{c_{i+1} - c_i}{3h}$, $i = 1,...,5$ $(c_6 = 0)$.

Having calculated a_i, b_i, c_i, d_i, $i = 1,...,5$, we shall find all spline coefficients.

Now let's calculate the value of the interpolation cubic spline at point $x^* = 0.7$. The point $x^* = 0.7$ belongs to the interval $[0.5, 0.9]$; on this interval the function is represented by the spline $S_2(x)$:

$$S_2(x) = -0.3466 + 0.1459(x - 0.5) + 1.6377(x - 0.5)^2 - 1.0727(x - 0.5)^3, \; 0.5 \le x \le 0.9.$$

Let's calculate the value of the spline $S_2(x)$ and the exact value of the function at point $x^* = 0.7$: $S(0.7) = S_2(0.7) = -0.2605$, $y(0.7) = 0.7 \cdot \ln(0.7) = -0.2497$.

So, the absolute error of interpolation is:

$\Delta(S(x^*)) = |S_2(0.7) - y(0.7)| = 0.0108$.

The calculation results are presented in table 10.

Table 10

i	$[x_{i-1}, x_i]$	a_i	b_i	c_i	d_i
1	[0.1, 0.5]	-0.2303	-0.5091	0.0000	1.3647
2	[0.5, 0.9]	**-0.3466**	**0.1459**	**1.6377**	**-1.0727**
3	[0.9, 1.3]	-0.0948	0.9412	0.3505	0.0523
4	[1.3, 1.7]	0.3411	1.2467	0.4133	-0.0594
5	[1.7, 2.1]	0.9021	1.5488	0.3420	-0.2850

4.9. PROGRAM #14

Below is a proposed variant of the program algorithm for calculating the coefficients of the cubic spline and its value at a given point.

```
ALGORITHM “Cubic spline”
INPUT     n, x[n], y[n], u
OUTPUT    i, j, p[n-2], q[n-2], a[n-1], b[n-1], c[n-1], d[n-1], v
BEGIN
    CYCLE “Coefficients a” FOR i FROM 1 TO n-1 BY 1
        a[i]:=y[i]
    c[1]:=0
    #Sweep method for coefficients c
    p[1]:=-(x[3]-x[2])/(2*x[3]-2*x[1])
    q[1]:=3*((y[3]-y[2])/(x[3]-x[2])-(y[2]-y[1])/(x[2]-x[1]))/
(2*x[3]-2*x[1])
    CYCLE “Forward Path” FOR i FROM 2 TO n-2 BY 1
        p[i]:=-(x[i+2]-x[i+1])/((2*x[i+2]-2*x[i])+(x[i+1]-x[i])*p[i-1])
        q[i]:=(3*((y[i+2]-y[i+1])/(x[i+2]-x[i+1])-(y[i+1]-y[i])/
(x[i+1]-x[i]))-(x[i+1]-x[i])*q[i-1])/((2*x[i+2]-2*x[i])+(x[i+1]-x[i])*p[i-1])
    c[n-1]:=q[n-2]
    CYCLE “Backward Path” FOR i FROM n-2 TO 2 BY -1
        c[i]:=p[i-1]*c[i+1]+q[i-1]
    CYCLE “Coefficients b and d” FOR i FROM 1 TO n-2 BY 1
        b[i]:=(y[i+1]-y[i])/(x[i+1]-x[i])-(x[i+1]-x[i])*(c[i+1]+2*c[i])/3
        d[i]:=(c[i+1]-c[i])/(x[i+1]-x[i])/3
    b[n-1]:=(y[n]-y[n-1])/(x[n]-x[n-1])-(x[n]-x[n-1])*2*c[n-1]/3
    d[n-1]:=-c[n-1]/(x[n]-x[n-1])/3
    j:=0
    CYCLE “Interval” FOR i FROM 1 TO n-1 BY 1
        IF (u≥x[i])AND(u≤x[i+1]) j:=i
    IF (j=0) END
```

```
        u:=u-x[j]
        v:=a[j]+b[j]*u+c[j]*u*u+d[j]*u*u*u
        PRINT a, b, c, d, v
END
```

4.10. METHOD OF LEAST SQUARES

Often in various problems, for example, when approximating a large number of experimental points found with a certain error, interpolation, i.e. the method of approximation when the values of the approximee and the approximating functions coincide at the nodes of a certain grid, becomes unreasonable. In this case, it is advisable to construct the approximating function in such a way as to smooth out the influence of the measurement error and the number of experimental points. Such smoothing is implemented when constructing an approximating function using the method of least squares. Herewith is sought the minimization of the sum of the squares of the deviations of the values of the approximee and approximating functions at the grid nodes, which is called the squared deviation.

Suppose on the interval $[a,b]$, there is a discrete set of mismatched points x_i, $i=0,...,n$, $x_0=a$, $x_n=b$ (interpolation nodes), at which the values of a certain function $y(x)$, generating the table $y_i=y(x_i)$, $i=0,...,n$, are known. It is assumed that the values of the function $y_i=y(x_i)$ are determined with some error, while some other (for example, physical) considerations reveal the form of the function, for which the table points should approximately satisfy. The form of the approximating function can be arbitrary, but, as a rule, it is assumed that it is a polynomial of degree m ($m<n$), for which the coefficients a_j are unknown: $P_m(x)=\sum_{j=0}^{m} a_j x^j$. Then, the unknown coefficients are found from the condition of the minimum total squared error of the polynomial from the table function:

$$\Phi=\sum_{i=0}^{n}\left(P_m(x_i)-y_i\right)^2\to\min.$$

This form of the criterion has some obvious advantages over other deviation functions Φ (for example, the sum of deviation modules) and, accordingly, gives the name to the method of least squares (MLS). The average value of the approximation error at the node can then be expressed as $\varepsilon = \frac{\sqrt{\Phi}}{n+1}$.

The minimum of the function Φ can be achieved only by changing the coefficients a_j of the approximating polynomial $P_m(x) = \sum_{j=0}^{m} a_j x^j$. The necessary conditions for the extremum (minimum, which is obvious from the form of the function Φ) of the function of many variables is the equality of all its first order partial derivatives to zero. They look as follows ($k = 0,...,m$):

$$\frac{\partial \Phi}{\partial a_k} = \frac{\partial}{\partial a_k}\left[\sum_{i=0}^{n}\left(P_m(x_i) - y_i\right)^2\right] = \frac{\partial}{\partial a_k}\left[\sum_{i=0}^{n}\left(\sum_{j=0}^{m} a_j x_i^j - y_i\right)^2\right] = 2\sum_{i=0}^{n}\left[\sum_{j=0}^{m} a_j x_i^j - y_i\right] x_i^k = 0.$$

For convenience, this system is converted to the following form:

$$\sum_{j=0}^{m} a_j \sum_{i=0}^{n} x_i^{k+j} = \sum_{i=0}^{n} y_i x_i^k,\ k = 0,...,m.$$

This system is called the normal system of the method of least squares and represents a system of linear algebraic equations with respect to the coefficients a_j with a symmetric matrix. Having solved this system (by any method) and having found the coefficients a_j, we obtain the polynomial $P_m(x)$ that approximates the function $y(x)$ and minimizes the squared deviation Φ. Obviously, with the degree m of the approximating polynomial $P_m(x)$ increasing, the obtained values of the deviation function Φ decrease, and when $m = n$, $\Phi = 0$, and the polynomial will become an interpolation one and will pass through all the interpolation nodes. However, it should be noted that, with the increase of the degree m, this system becomes ill-conditioned and its solution can be tied up with a large loss of accuracy. Therefore, when using the method of least squares, as a rule, an approximating polynomial of a degree not higher than the third one is used.

Example. Using a table of values of the function $y(x)$ (values y_i calculated at points x_i, $i=0,...,n$), solve the normal system of the method of least squares for points (x_i, y_i) to construct approximating polynomials of the 1st and 2nd degree. Calculate the sum of squared errors for each of the approximating polynomials.

$y = x\ln(x)$, $n=5$, $i=0,...,5$, $x_i = 0.1, 0.5, 0.9, 1.3, 1.7, 2.1$.

Let's find the approximating polynomial of the 1st degree $P_1(x) = a_0 + a_1 x$.

To find the unknown coefficients a_0, a_1, let's write down the normal system of the method of least squares ($n=5, m=1$):

$$\begin{cases} a_0(n+1) + a_1\sum_{i=0}^{n} x_i = \sum_{i=0}^{n} y_i \\ a_0\sum_{i=0}^{n} x_i + a_1\sum_{i=0}^{n} x_i^2 = \sum_{i=0}^{n} y_i x_i \end{cases}, \quad \begin{cases} 6a_0 + 6.6a_1 = 2.1296 \\ 6.6a_0 + 10.06a_1 = 4.9672 \end{cases}.$$

Having solved this system, we get the coefficients $a_0 = -0.6762$, $a_1 = 0.9374$. Thus, the approximating polynomial of the 1st degree has been built as follows:

$P_1(x) = -0.6762 + 0.9374x$.

The sum of squared errors for it is $\Phi_1 = \sum_{i=0}^{5} \left(P_1(x_i) - y_i\right)^2 = 0.3236$.

Let's find the approximating polynomial of the 2nd degree $P_2(x) = a_0 + a_1 x + a_2 x^2$.

To find the unknown coefficients a_0, a_1, a_2, let's write down the normal system of the method of least squares ($n=5$, $m=2$):

$$\begin{cases} a_0(n+1) + a_1\sum_{i=0}^{n} x_i + a_2\sum_{i=0}^{n} x_i^2 = \sum_{i=0}^{n} y_i \\ a_0\sum_{i=0}^{n} x_i + a_1\sum_{i=0}^{n} x_i^2 + a_2\sum_{i=0}^{n} x_i^3 = \sum_{i=0}^{n} y_i x_i \\ a_0\sum_{i=0}^{n} x_i^2 + a_1\sum_{i=0}^{n} x_i^3 + a_2\sum_{i=0}^{n} x_i^4 = \sum_{i=0}^{n} y_i x_i^2 \end{cases}, \quad \begin{cases} 6a_0 + 6.6a_1 + 10.06a_2 = 2.1296 \\ 6.6a_0 + 10.06a_1 + 17.226a_2 = 4.9672 \\ 10.06a_0 + 17.226a_1 + 31.375a_2 = 9.8887 \end{cases}.$$

Having solved this system, we get the coefficients $a_0 = -0.2532$, $a_1 = -0.3145$, $a_2 = 0.5690$. Thus, the approximating polynomial of the 2nd degree has been built as follows:

$P_2(x) = -0.2532 - 0.3145x + 0.5690x^2$.

The sum of squared errors for it is $\Phi_2 = \sum_{i=0}^{5} \left(P_2(x_i) - y_i\right)^2 = 0.0141$.

The graphs of the obtained approximating polynomials and the original function are shown in fig. 8.

The calculation results are presented in table 11.

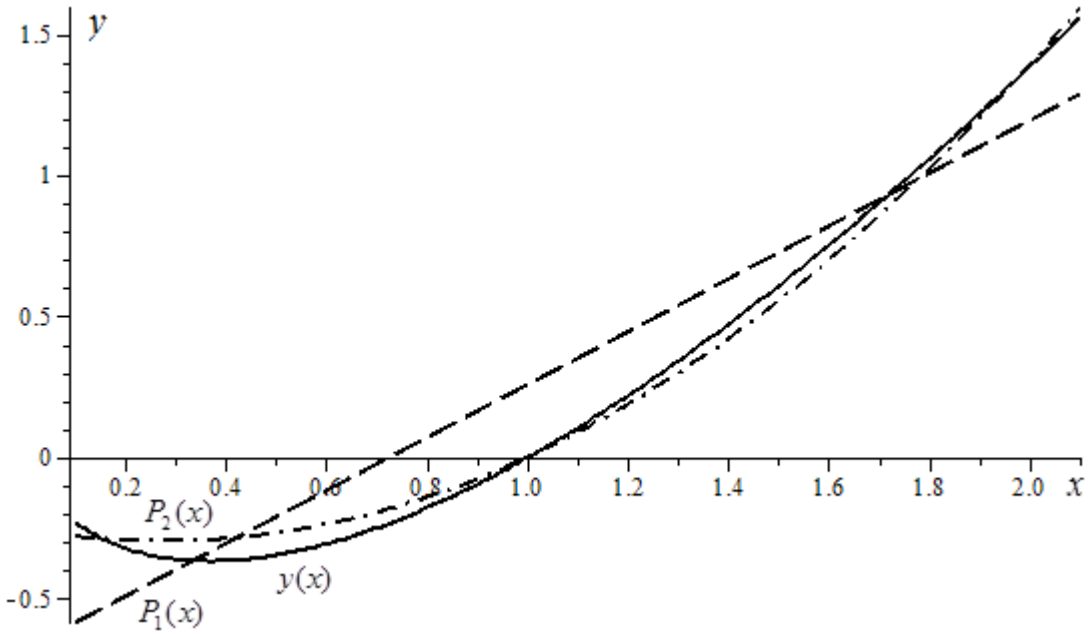

Fig. 8. The original function and the approximating polynomials of the 1st and 2nd degree

Table 11

i	**0**	**1**	**2**	**3**	**4**	**5**
x_i	0.1	0.5	0.9	1.3	1.7	2.1
y_i	-0.2303	-0.3466	-0.0948	0.3411	0.9021	1.5581
$P_1(x_i)$	-0.5825	-0.2075	0.1674	0.5424	0.9174	1.2923
$P_2(x_i)$	-0.2790	-0.2682	-0.0753	0.2996	0.8567	1.5958

4.11. PROGRAM #15

Below is a proposed variant of the program algorithm for compiling the normal system of the method of least squares, solving it using the Cramer's method for the approximating polynomials of the 1st and 2nd degree and calculating the sum of squared errors.

```
ALGORITHM “Method of least squares”
INPUT     n, x[n], y[n], m
OUTPUT    i, j, k, g, a[m][m], b[m], z[m], f
BEGIN
    #Normal system of the method of least squares
    CYCLE “Rows” FOR i FROM 1 TO m BY 1
        CYCLE “Columns” FOR j FROM 1 TO m BY 1
            a[i][j]:=0
            CYCLE “Summands” FOR k FROM 1 TO n BY 1
                a[i][j]:=a[i][j]+POWER(x[k],i+j-2)
        b[i]:=0
        CYCLE “Summands” FOR k FROM 1 TO n BY 1
            b[i]:=b[i]+y[k]*POWER(x[k],i-1)
    #Cramer's method for the normal system
    IF (m=1) z[1]:=b[1]/a[1][1]
    IF (m=2)
        g:=a[1][1]*a[2][2]-a[2][1]*a[1][2]
        z[1]:=(b[1]*a[2][2]-b[2]*a[2][1])/g
        z[2]:=(a[1][1]*b[2]-a[2][1]*b[1])/g
    IF (m=3)
        g:=a[1][1]*a[2][2]*a[3][3]+a[2][1]*a[3][2]*a[1][3]+
a[3][1]*a[1][2]*a[2][3]-a[3][1]*a[2][2]*a[1][3]-a[1][1]*a[3][2]*a[2][3]-
a[2][1]*a[1][2]*a[3][3]
        z[1]:=(b[1]*a[2][2]*a[3][3]+b[2]*a[3][2]*a[1][3]+
b[3]*a[1][2]*a[2][3]-b[3]*a[2][2]*a[1][3]-b[1]*a[3][2]*a[2][3]-
b[2]*a[1][2]*a[3][3])/g
        z[2]:=(a[1][1]*b[2]*a[3][3]+a[2][1]*b[3]*a[1][3]+
a[3][1]*b[1]*a[2][3]-a[3][1]*b[2]*a[1][3]-a[1][1]*b[3]*a[2][3]-
a[2][1]*b[1]*a[3][3])/g
```

```
                z[3]:=(a[1][1]*a[2][2]*b[3]+a[2][1]*a[3][2]*b[1]+
a[3][1]*a[1][2]*b[2]-a[3][1]*a[2][2]*b[1]-a[1][1]*a[3][2]*b[2]-
a[2][1]*a[1][2]*b[3])/g
        #Sum of squared errors
        f:=0
        CYCLE “Points” FOR k FROM 1 TO n BY 1
            g:=0
            CYCLE “Summands” FOR i FROM 1 TO m BY 1
                g:=g+z[i]*POWER(x[k],i-1)
            f:=f+(g-y[k])*(g-y[k])
        PRINT a, b, z, f
    END
```