4. NUMERICAL METHODS AND ALGORITHMS FOR THE APPROXIMATION OF FUNCTIONS

This chapter discusses basic numerical methods and algorithms of computational mathematics in the field of approximation of functions. At first, interpolation and construction of interpolation polynomials in the Lagrangian and Newtonian forms are presented. After that, spline interpolation and the method of least squares are considered. The last topics of this chapter are devoted to the problem of finding formulas for numerical differentiation (obtained by expanding in a Taylor series and by differentiating the Newton interpolation polynomial) and numerical integration (obtained based on the definition of the Riemann integral and by integrating the Lagrange interpolation polynomial). The accuracy of the derived formulas and ways to improve it are discussed, and the possibility of a-posteriori refinement of the result of numerical differentiation and numerical integration by the Runge-Romberg-Richardson method is considered.

4.1. APPROXIMATION OF FUNCTIONS

Many problems of practical importance imply the problem of replacing some analytical or table function with another one that is close to the first one in some sense, but is simpler, has been better investigated, and is more convenient for calculations. So, here we face two mathematical problems. The first problem is to replace the analytical function. For example, replacement with a polynomial allows getting simple numerical integration and differentiation formulas. The second problem is to restore the function on a certain interval using the values of the function specified on this interval in a discrete set of points. In this case, replacing the given table with an approximating function allows getting values at its intermediate points. In any case, when formulating any of the problems of function approximation, it is necessary to answer the following questions.

First, to decide which class of approximating functions it is necessary to use. The answer to this question depends on the type of the approximee function and the purposes, for which the approximating function will be used in the future. The

following classes of functions are widely used: polynomials, trigonometric functions, exponential functions, etc.

Second, to choose a criterion for the proximity of the original and approximating functions. As a criterion, we can choose, for example, the exact coincidence of the approximee and approximating functions at the nodal points (the Lagrange interpolation), the least of the sum of squared deviations at the nodal points (the method of least squares) and others. Like with the choice of the class of approximating functions, the choice of the criterion of proximity between the original and approximating functions is determined by the purpose of constructing the approximating function and can greatly affect the results. For example, when approximating experimental results, it is advisable to use the mean-square approximation, as interpolation can incorrectly describe the properties of functions and only exacerbate experimental errors. Such an example is shown in fig. 7, where the first curve is the result of approximation of nodal points by interpolation, and the second curve is the mean-square approximation.

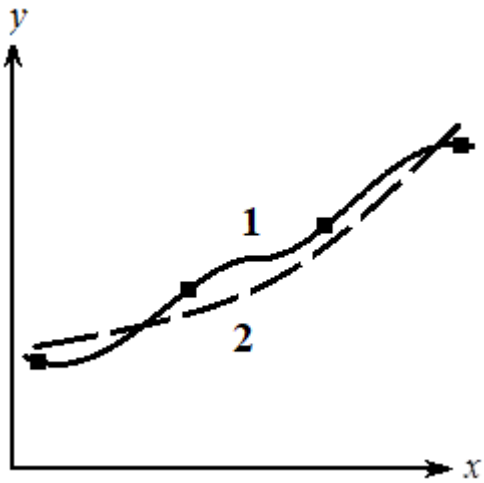

Fig. 7. Comparison of various criteria for the approximation of functions

Third, it is necessary to specify the rule that allows to obtain the value of the function in the intervals between the nodes with a given order of accuracy, in particular, to answer the questions, which nodes it is necessary to use to construct the approximating function and how to arrange them.

So, the solution to the problem of constructing an approximating function substantially depends on the answers to the questions stated above.

4.2. INTERPOLATION OF FUNCTIONS

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. Then, the approximating function $\varphi(x,a)$ with the parameter vector a is called the interpolation function if it coincides with the approximee function $y(x)$ in the $n+1$ table nodes, i.e. the equalities are satisfied:

$$\varphi(x_i,a)=y(x_i)=y_i,\ i=0,...,n.$$

This method of constructing the approximating function, when the values of the approximee and the approximating function coincide at the given nodes, is called interpolation (or Lagrange interpolation). The method of linear interpolation, when the approximating function is sought in the form of a linear combination of some basic functions $\varphi_j(x)$, $j=0,...,n$, is most widespread:

$$\varphi(x,a_0,...,a_n)=\sum_{j=0}^{n}a_j\varphi_j(x).$$

It is obvious that the selected system of functions $\varphi_j(x)$ must be linearly independent, as otherwise the number of parameters and terms in the sum could be reduced. In addition, it is required that

$$\det\begin{pmatrix}\varphi_0(x_0) & \varphi_1(x_0) & ... & \varphi_n(x_0)\\ \varphi_0(x_1) & \varphi_1(x_1) & ... & \varphi_n(x_1)\\ ... & ... & ... & ...\\ \varphi_0(x_n) & \varphi_1(x_n) & ... & \varphi_n(x_n)\end{pmatrix}\neq 0.$$

Only in this case, the system of linear algebraic equations for determining the coefficients a_j will have the unique solution:

$$\sum_{j=0}^{n}a_j\varphi_j(x_i)=y_i,\ i=0,...,n.$$

We can choose any linearly independent system of functions as basic functions $\varphi_j(x)$, but power functions $1,x,x^2,...,x^n$ are selected most often. This is because polynomials are easy to compute and the theory of interpolation by

polynomials is well developed. In this case, polynomials of *n*-th degree are used as the approximating function: $P_n(x) = \sum_{j=0}^{n} a_j x^j$.

Substituting the values of interpolation nodes into this polynomial and using the condition $P_n(x_i) = y_i$, $i = 0,...,n$, we obtain a system of linear algebraic equations with respect to the coefficients a_j, $j = 0,...,n$:

$$\sum_{j=0}^{n} a_j (x_i)^j = y_i,\ i = 0,...,n,$$

which, in case of the mismatched interpolation nodes, has the unique solution, as the Vandermonde determinant is then nonzero:

$$\det \begin{pmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ ... & ... & ... & ... & ... \\ 1 & x_n & x_n^2 & ... & x_n^n \end{pmatrix} = \prod_{0 \le i < j}^{n} (x_j - x_i) \neq 0 .$$

As a result, the constructed interpolation polynomial $P_n(x)$ exists and is unique up to the notation form and, if $y(x)$ is not the *n*-th degree polynomial, it approximates the original function $y(x)$ with a certain error.

4.3. LAGRANGE INTERPOLATION POLYNOMIAL

To find the interpolation polynomial $P_n(x) = \sum_{j=0}^{n} a_j x^j$, it is not necessary to solve the system of linear algebraic equations $\sum_{j=0}^{n} a_j (x_i)^j = y_i$, $i = 0,...,n$. An arbitrary similar polynomial of degree *n* can be written in the following alternative form:

$$L_n(x) = \sum_{j=0}^{n} y_j l_j(x).$$

Here, $l_j(x)$, $j = 0,...,n$ are some auxiliary polynomials of degree *n* called Lagrange polynomials of influence that satisfy the condition:

$$l_j(x_i) = \begin{cases} 1, & j = i \\ 0, & j \neq i \end{cases}.$$

In this case, the only possible representation for such polynomials is $l_j(x) = \prod_{i=0, i\neq j}^{n} \frac{(x-x_i)}{(x_j-x_i)}$ and, as a result, the interpolation polynomial $L_n(x) = \sum_{j=0}^{n} y_j l_j(x)$ will be written as follows:

$$L_n(x) = \sum_{j=0}^{n} y_j \prod_{i=0, i\neq j}^{n} \frac{(x-x_i)}{(x_j-x_i)}.$$

A polynomial written in this form is called a Lagrange interpolation polynomial. It is easy to see that it is identical to the desired interpolation polynomial $P_n(x) = \sum_{j=0}^{n} a_j x^j$. Indeed, in terms of construction, the polynomial $L_n(x)$ passes through all points (x_i, y_i), $i = 0,...,n$ and herewith is a polynomial of degree *n*, which means that it is the only such polynomial $P_n(x)$.

An important advantage of this form of writing the interpolation polynomial is that the number of arithmetic operations necessary to construct a Lagrange polynomial is proportional to n^2 and is the smallest for all forms of writing. The advantages of the Lagrange interpolation polynomial also include that the polynomial formula explicitly contains the values of the function at the interpolation nodes, which can be convenient for some calculations, in particular, when constructing formulas for numerical integration; it is also especially convenient when the values of the function change, but the interpolation nodes are unchanged, which happens in many experimental studies. Let's also note that the formula is applicable both to equally spaced and non-equally spaced nodes.

The disadvantage of the Lagrange interpolation polynomial is the need for a complete recalculation of all coefficients of $L_n(x)$ in case of a change in the number of interpolation nodes. This makes it difficult to conduct a-posteriori accuracy estimates (estimates obtained in the calculation process).

Also, if we introduce function $\omega_{n+1}(x) = (x-x_0)(x-x_1)...(x-x_n) = \prod_{i=0}^{n}(x-x_i)$, then the expression for the Lagrange interpolation polynomial can be written as follows:

$$L_n(x) = \sum_{j=0}^{n} y_j \frac{\omega_{n+1}(x)}{(x-x_j)\omega'_{n+1}(x_j)}.$$

Example. Using a table of values of the function $y(x)$ (values y_i calculated at points x_i, $i=0,...,n$), construct a Lagrange interpolation polynomial passing through points (x_i, y_i). Calculate the value of the interpolation error at point x^*.

$y = x\ln(x)$, $n=3$, $i=0,...,3$, $x_i = 0.1, 0.5, 0.9, 1.3$, $x^* = 0.7$.

The function $y = x\ln(x)$ is defined at four points; therefore, the desired one is the 3rd degree Lagrange polynomial: $L_3(x) = \sum_{j=0}^{3} y_j \prod_{i=0, i\neq j}^{3} \frac{(x - x_i)}{(x_j - x_i)}$.

$$L_3(x) = y_0 \frac{(x-x_1)}{(x_0-x_1)} \frac{(x-x_2)}{(x_0-x_2)} \frac{(x-x_3)}{(x_0-x_3)} + y_1 \frac{(x-x_0)}{(x_1-x_0)} \frac{(x-x_2)}{(x_1-x_2)} \frac{(x-x_3)}{(x_1-x_3)} +$$

$$+ y_2 \frac{(x-x_0)}{(x_2-x_0)} \frac{(x-x_1)}{(x_2-x_1)} \frac{(x-x_3)}{(x_2-x_3)} + y_3 \frac{(x-x_0)}{(x_3-x_0)} \frac{(x-x_1)}{(x_3-x_1)} \frac{(x-x_2)}{(x_3-x_2)}.$$

After substituting the values of coordinates of points (x_i, y_i) and calculating the coefficients, the desired Lagrange polynomial can be written as follows:

$$L_3(x) = 0.5996 \cdot (x-0.5)(x-0.9)(x-1.3) - 2.7076 \cdot (x-0.1)(x-0.9)(x-1.3) +$$

$$+ 0.7408 \cdot (x-0.1)(x-0.5)(x-1.3) + 0.8882 \cdot (x-0.1)(x-0.5)(x-0.9).$$

Let's calculate the value of the interpolation polynomial and the exact value of the function at point $x^* = 0.7$: $L_3(0.7) = -0.2552$, $y(0.7) = 0.7 \cdot \ln(0.7) = -0.2497$.

So, the absolute error of interpolation is:

$\Delta(L_3(x^*)) = |L_3(0.7) - y(0.7)| = 0.0055$.

The calculation results are presented in table 7.

Table 7

j	x_j	y_j	$\prod_{i=0, i\neq j}^{3} (x_j - x_i)$	$y_j \Big/ \prod_{i=0, i\neq j}^{3} (x_j - x_i)$
0	0.1	-0.2303	-0.3840	**0.5996**
1	0.5	-0.3466	0.1280	**-2.7076**
2	0.9	-0.0948	-0.1280	**0.7408**
3	1.3	0.3411	0.3840	**0.8882**

4.4. PROGRAM #12

Below is a proposed variant of the program algorithm for calculating the coefficients of the Lagrange interpolation polynomial and its value at a given point.

```
ALGORITHM “Lagrange polynomial”
INPUT     n, x[n], y[n], u
OUTPUT    i, j, f, w[n], v
BEGIN
    v:=0
    CYCLE “Summands” FOR j FROM 1 TO n BY 1
        w[j]:=y[j], f:=y[j]
        CYCLE “Multipliers” FOR i FROM 1 TO n BY 1
            IF (i≠j) w[j]:=w[j]/(x[j]-x[i]), f:=f*(u-x[i])/(x[j]-x[i])
        v:=v+f
    PRINT w, v
END
```

4.5. NEWTON INTERPOLATION POLYNOMIAL

The interpolation polynomial in the Newtonian form represents another alternative form of writing the interpolation polynomial. To derive it, let's introduce the concept of a divided difference $\Delta^{p}_{i_q \ldots i_{q+p}}$.

Zeroth order divided differences coincide with the values of the function at the corresponding nodes: $\Delta^{0}_{i_q} = y_{i_q}$.

First order divided differences of two neighboring nodes are determined through zeroth order divided differences as follows:

$$\Delta^{1}_{i_q i_{q+1}} = \frac{y_{i_q} - y_{i_{q+1}}}{x_{i_q} - x_{i_{q+1}}} = \frac{\Delta^{0}_{i_q} - \Delta^{0}_{i_{q+1}}}{x_{i_q} - x_{i_{q+1}}}.$$

Second order divided differences of three neighboring nodes are determined through first order divided differences as follows:

$$\Delta^2_{i_q i_{q+1} i_{q+2}} = \frac{\Delta^1_{i_q i_{q+1}} - \Delta^1_{i_{q+1} i_{q+2}}}{x_{i_q} - x_{i_{q+2}}}.$$

The divided difference of order *p* for $p+1$ neighboring nodes is determined by recursive relations through the divided differences of order $p-1$ as follows:

$$\Delta^p_{i_q \dots i_{q+p}} = \frac{\Delta^{p-1}_{i_q \dots i_{q+p-1}} - \Delta^{p-1}_{i_{q+1} \dots i_{q+p}}}{x_{i_q} - x_{i_{q+p}}}.$$

So, for $n+1$ points $x_0, x_1, ..., x_n$ of the table function $y_i = y(x_i)$, $i = 0, ..., n$, the divided differences of up to the *n*-th order can be constructed; the divided differences of higher orders are equal to zero.

Based on the mathematical form of divided differences, we can make a conclusion that they are analogs of the derivatives of the corresponding orders. And indeed, there is the following ratio between them:

$$y^{(n)}(x) = n! \cdot \Delta^n_{0 \dots n}.$$

As a result, the desired interpolation polynomial $P_n(x) = \sum_{j=0}^{n} a_j x^j$, which values at the interpolation nodes coincide with the values of the table function $y_i = y(x_i)$, $i = 0, ..., n$, can be written as follows:

$$N_n(x) = \sum_{j=0}^{n} \Delta^j_{0 \dots j} \prod_{i=0}^{j-1} (x - x_i),$$

$$N_n(x) = \Delta^0_0 + \Delta^1_{01}(x - x_0) + \Delta^2_{012}(x - x_0)(x - x_1) + \dots + \Delta^n_{0 \dots n}(x - x_0)(x - x_1)\dots(x - x_{n-1}).$$

A polynomial written in such form is called a Newton interpolation polynomial. It is easy to show that in terms of its construction it passes through all points (x_i, y_i), $i = 0, ..., n$, and since herewith it is a polynomial of degree *n*, then, based on the uniqueness condition of the interpolation polynomial, it is identical to the desired one, as well as the Newton polynomial coincides with the Lagrange polynomial.

When constructing a Newton interpolation polynomial, it is convenient to use a table of a pyramidal form called a table of divided differences, which is filled in by columns sequentially from left to right. To write the formula for the desired

Newton polynomial, the divided differences located in the first row of this table are then used. An example of such a table of divided differences for $n=4$ is given in table 8.

Table 8

x_0	$\Delta_0^0 = y_0$	$\Delta_{01}^1 = \frac{\Delta_0^0 - \Delta_1^0}{x_0 - x_1}$	$\Delta_{012}^2 = \frac{\Delta_{01}^1 - \Delta_{12}^1}{x_0 - x_2}$	$\Delta_{0123}^3 = \frac{\Delta_{012}^2 - \Delta_{123}^2}{x_0 - x_3}$	$\Delta_{01234}^4 = \frac{\Delta_{0123}^3 - \Delta_{1234}^3}{x_0 - x_4}$
x_1	$\Delta_1^0 = y_1$	$\Delta_{12}^1 = \frac{\Delta_1^0 - \Delta_2^0}{x_1 - x_2}$	$\Delta_{123}^2 = \frac{\Delta_{12}^1 - \Delta_{23}^1}{x_1 - x_3}$	$\Delta_{1234}^3 = \frac{\Delta_{123}^2 - \Delta_{234}^2}{x_1 - x_4}$	
x_2	$\Delta_2^0 = y_2$	$\Delta_{23}^1 = \frac{\Delta_2^0 - \Delta_3^0}{x_2 - x_3}$	$\Delta_{234}^2 = \frac{\Delta_{23}^1 - \Delta_{34}^1}{x_2 - x_4}$		
x_3	$\Delta_3^0 = y_3$	$\Delta_{34}^1 = \frac{\Delta_3^0 - \Delta_4^0}{x_3 - x_4}$			
x_4	$\Delta_4^0 = y_4$				

Let's note that to increase the accuracy of interpolation, additional interpolation nodes $x_{n+1}, x_{n+2}, \ldots$ can be added to the initial table function $y_i = y(x_i)$. In this case, the table of divided differences is supplemented, the Newton polynomial receives new summands into the sum, and all previous summands do not require recalculation. Herewith, it does not matter in which order new nodes are connected. This favorably differs the Newton formula from the Lagrange formula, where if new nodes are added, it is required to carry out all calculations again. The Newton interpolation polynomial is convenient when constructing formulas for numerical differentiation, which is obvious from its form. In addition, it is convenient for conducting a-posteriori estimates. The disadvantage of the Newton interpolation polynomial is the need for its complete recalculation when the value of the function at least at one node changes.

The error of the Lagrange and Newton interpolation polynomials outside of the interpolation nodes for the case of the analytical function $y(x)$ can be a-priori estimated using the formula

$$|\varepsilon_n(x)| = |y(x) - P_n(x)| \le \frac{M_{n+1}}{(n+1)!}|\omega_{n+1}(x)|, \text{ where } M_{n+1} = \max\left|y^{(n+1)}(\xi)\right|, \xi \in [x_0, x_n].$$

If it is difficult to estimate the value of the derivatives of the interpolated function (for example, for the table function), then a-posteriori estimate is used by the first discarded term of the Newton interpolation polynomial, since it includes divided differences that are analogues to the derivatives of the corresponding orders.

An approximation problem is called extrapolation when the point at which the value of the approximating function is sought is outside of the interpolation interval. Let's note that when extrapolating with the use of the Lagrange or Newton interpolation polynomials at points far from the ends of the interpolation interval, the approximation error can become really large.

Example. Using a table of values of the function $y(x)$ (values y_i calculated at points x_i, $i=0,...,n$), construct a Newton interpolation polynomial passing through points (x_i, y_i). Calculate the value of the interpolation error at point x^*.

$y = x\ln(x)$, $n=3$, $i=0,...,3$, $x_i = 0.1, 0.5, 0.9, 1.3$, $x^* = 0.7$.

The function $y = x\ln(x)$ is defined at four points; therefore, the desired one is the 3rd degree Newton polynomial: $N_3(x) = \sum_{j=0}^{3} \Delta_{0...j}^{j} \prod_{i=0}^{j-1} (x - x_i)$.

$$N_3(x) = \Delta_0^0 + \Delta_{01}^1 (x - x_0) + \Delta_{012}^2 (x - x_0)(x - x_1) + \Delta_{0123}^3 (x - x_0)(x - x_1)(x - x_2).$$

After substituting the values of coordinates of points (x_i, y_i) and calculating the coefficients, the desired Newton polynomial can be written as follows:

$$N_3(x) = -0.2303 - 0.2908 \cdot (x - 0.1) +$$

$$+1.1502 \cdot (x - 0.1)(x - 0.5) - 0.4789 \cdot (x - 0.1)(x - 0.5)(x - 0.9).$$

Let's calculate the value of the interpolation polynomial and the exact value of the function at point $x^* = 0.7$: $N_3(0.7) = -0.2552$, $y(0.7) = 0.7 \cdot \ln(0.7) = -0.2497$.

So, the absolute error of interpolation is:

$$\Delta(N_3(x^*)) = |N_3(0.7) - y(0.7)| = 0.0055.$$

The calculation results are presented in table 9.

Table 9

i	x_i	$\Delta_i^0 = y_i$	$\Delta_{i(i+1)}^1$	$\Delta_{i(i+1)(i+2)}^2$	$\Delta_{i(i+1)(i+2)(i+3)}^3$
0	0.1	**-0.2303**	**-0.2908**	**1.1502**	**-0.4789**
1	0.5	-0.3466	0.6294	0.5755	
2	0.9	-0.0948	1.0897		
3	1.3	0.3411			

4.6. PROGRAM #13

Below is a proposed variant of the program algorithm for calculating the coefficients of the Newton interpolation polynomial and its value at a given point.

```
ALGORITHM “Newton polynomial”
INPUT      n, x[n], y[n], u
OUTPUT     i, j, d[n][n], f, w[n], v
BEGIN
    #Table of divided differences
    CYCLE “Rows” FOR i FROM 1 TO n BY 1
        d[i][1]:=y[i]
    CYCLE “Columns” FOR j FROM 2 TO n BY 1
        CYCLE "Rows" FOR i FROM 1 TO n-j+1 BY 1
            d[i][j]:=(d[i][j-1]-d[i+1][j-1])/(x[i]-x[i+j-1])
    v:=0
    CYCLE “Summands” FOR j FROM 1 TO n BY 1
        w[j]:=d[1][j], f:=d[1][j]
        CYCLE “Multipliers” FOR i FROM 1 TO j-1 BY 1
            f:=f*(u-x[i])
        v:=v+f
    PRINT w, v
END
```