Note on Equidistant Polynomial Interpolation

Gao Shang*
School of Computer Science and Engineering
Jiangsu University of Science and Technology
Zhenjiang, China
e-mail: gao_shang@just.edu.cn

Qian qiang
School of Computer Science and Engineering
Jiangsu University of Science and Technology
Zhenjiang, China
e-mail: qianqiang_just@163.com

Abstract—In the mathematical field of numerical analysis, interpolation is a method of constructing new data points within the range of a discrete set of known data points. Based on analysis of basic polynomial interpolation, the equidistant polynomial interpolation problem is studied. Simple divided difference is given and it is proved by mathematical induction. The computation is smaller than the traditional method. At last, this calculation method is illustrated through an example.

Keywords-polynomial interpolation; equidistant interpolation; lagrange interpolation; newton interpolation

I. Introduction

In numerical analysis, polynomial interpolation is the interpolation of a given data set by a polynomial: given some points, find a polynomial which goes exactly through these points[1-3]. Polynomials [4-10] can be used to approximate more complicated curves, for example, the shapes of letters in typography, given a few points. A relevant application is the evaluation of the natural logarithm and trigonometric functions: pick a few known data points, create a lookup table, and interpolate between those data points. This results in significantly faster computations. Polynomial interpolation also forms the basis for algorithms in numerical quadrature and numerical ordinary differential equations. Most textbooks only discuss the general case (non-equidistant) polynomial interpolation problem. It has three methods[1-3], such as linear system method, Lagrange's Interpolation Formula and Newton's Interpolation Formula for determining the polynomial interpolation. In practical, the equidistant interpolation often occurs. Since the equidistant interpolation is a special kind of interpolation, so the solution should be more simple. The equidistant polynomial interpolation problem is studied in this paper.

II. General methods to solve polynomial interpolation

A. Linear System Method

Definition: Given a set of $n+1$ data points (x_i, y_i) where no two x_i are the same, one is looking for a polynomial $P_n(x)$ of degree at most n with the property $P_n(x_i) = y_i$, $i = 0,1,2,\cdots,n$.

Suppose that the interpolation polynomial is in the form

$$P_n(x) = a_0 + a_1 x + \cdots + a_n x^n \tag{1}$$

The statement that p interpolates the data points means that $P_n(x_i) = y_i$, $i = 0,1,2,\cdots,n$

If we substitute equation (1) in here, we get a system of linear equations in the coefficients a_k.

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n = y_1 \\ \cdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n = y_n \end{cases} \tag{2}$$

We have to solve this system for a_k to construct the interpolation $P_n(x)$.

B. Lagrange's Interpolation Formula

In numerical analysis, Lagrange polynomials are used for polynomial interpolation. the interpolation polynomial in the Lagrange form is a linear combination

$$P_n(x) = l_0(x)y_0 + l_1(x)y_1 + \cdots + l_n(x)y_n$$
$$= \sum_{k=0}^{n} l_k(x) y_k = \sum_{k=0}^{n} \Big(\prod_{\substack{j=0 \\ j\neq k}}^{n} \frac{x - x_j}{x_k - x_j} y_k \Big) \tag{3}$$

Where $l_k(x) = \prod_{\substack{j=0 \\ j\neq k}}^{n} \frac{x - x_j}{x_k - x_j}$.

C. Newton's Interpolation Formula

The divided differences for a function $f(x)$ are defined as follows:

$$f(x_0, x_1) = \frac{f(x_0) - f(x_1)}{x_0 - x_1} = \frac{f(x_0)}{x_0 - x_1} + \frac{f(x_1)}{x_1 - x_0}$$

$$f(x_0, x_1, x_2) = \frac{f(x_0, x_1) - f(x_1, x_2)}{x_0 - x_2}$$
$$= \frac{f(x_0)}{(x_0 - x_1)(x_0 - x_2)} + \frac{f(x_1)}{(x_1 - x_0)(x_1 - x_2)} + \frac{f(x_2)}{(x_2 - x_0)(x_2 - x_1)}$$

The recursive rule for constructing higher-order divided differences is

$$f(x_0,x_1,\cdots,x_n)=\frac{f(x_0,x_1,\cdots,x_{n-1})-f(x_1,x_2,\cdots,x_n)}{x_0-x_n}$$

$$=\sum_{k=0}^{n}\frac{f(x_k)}{\prod_{\substack{j=0\\ j\neq k}}^{n}(x_k-x_j)}$$

The divided difference formulae are used to construct the divided difference table 1.

TABLE I. DIVIDED-DIFFERENCE TABLE

x_k	$f(x_k)$	First divided difference	Second divided difference	Third divided difference
x_0	$f(x_0)$			
x_1	$f(x_1)$	$f(x_0,x_1)$		
x_2	$f(x_2)$	$f(x_1,x_2)$	$f(x_0,x_1,x_2)$	
x_3	$f(x_3)$	$f(x_2,x_3)$	$f(x_1,x_2,x_3)$	$f(x_0,x_1,x_2,x_3)$

The Newton polynomial of degree $\le n$ is

$$P_n(x)=f(x_0)+f(x_0,x_1)(x-x_0)$$
$$+f(x_0,x_1,x_2)(x-x_0)(x-x_1)+\cdots$$
$$+f(x_0,x_1,\cdots,x_n)(x-x_0)(x-x_1)\cdots(x-x_{n-1})$$

III. EQUIDISTANT INTERPOLATION

A. Equidistant Lagrange's Interpolation Formula

The equidistant polynomial interpolation is that the interpolation nodes x_i are equal intervals. That is $x_i=x_{i-1}+h\,(i=1,2,\cdots,n)$.

$$P_n(x)=\sum_{k=0}^{n}\frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}y_k$$

$$=\sum_{k=0}^{n}\frac{y_k}{(-1)^k k!(n-k)!h^n}(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)$$

$$=\sum_{k=0}^{n}[\frac{y_k}{(-1)^k k!(n-k)!h^n}\prod_{\substack{j=0\\ j\neq k}}^{n}(x-x_j)]$$

$$=\sum_{k=0}^{n}[E_k\prod_{\substack{j=0\\ j\neq k}}^{n}(x-x_j)]$$

where $E_k=\dfrac{y_k}{(-1)^k k!(n-k)!h^n}$.

B. Equidistant Newton's Interpolation Formula

The first divided differences are calculated as follows:

$$f(x_0,x_1)=\frac{y_1-y_0}{h}$$

$$f(x_1,x_2)=\frac{y_2-y_1}{h}$$

$$f(x_{n-1},x_n)=\frac{y_n-y_{n-1}}{h}$$

The second divided differences are calculated as follows:

$$f(x_0,x_1,x_2)=\frac{y_2-2y_1+y_0}{2h^2}$$

$$f(x_1,x_2,x_3)=\frac{y_3-2y_2+y_1}{2h^2}$$

$$f(x_{n-2},x_{n-1},x_n)=\frac{y_n-2y_{n-1}+y_{n-2}}{2h^2}$$

The third divided differences are calculated as follows:

$$f(x_0,x_1,x_2,x_3)=\frac{y_3-3y_2+3y_1+y_0}{3!h^3}$$

$$f(x_1,x_2,x_3,x_4)=\frac{y_4-3y_3+3y_2+y_1}{3!h^3}$$

$$f(x_{n-3},x_{n-2},x_{n-1},x_n)=\frac{y_n-3y_{n-1}+3y_{n-2}+y_{n-3}}{3!h^3}$$

The kth divided difference is calculated as follows:

$$f(x_0,x_1,\cdots,x_k)=\frac{y_k-C_k^1y_{k-1}+C_k^2y_{k-2}+\cdots(-1)^k y_0}{k!h^k} \quad (4)$$

$$=\frac{1}{k!h^k}\sum_{i=0}^{k}(-1)^i C_k^i y_{k-i}$$

Theorem: When $x_0,x_1,\cdots,x_n$ are equal intervals. That is $x_i=x_{i-1}+h\,(i=1,2,\cdots,n)$. then

$$f(x_0,x_1,\cdots,x_n)=\frac{y_n-C_n^1y_{n-1}+C_n^2y_{n-2}+\cdots(-1)^n y_0}{k!h^k}$$

$$=\frac{1}{n!h^k}\sum_{i=0}^{n}(-1)^i C_n^i y_{n-i}$$

Proof: (Proof by Mathematical Induction)

When $n=1$,

$$f(x_0,x_1)=\frac{y_1-y_0}{h}=\frac{y_1+(-1)^1C_1^1y_0}{h}.$$

Hence, the formula is true.

Assume that the formula is true for $n=k$,

$$f(x_0,x_1,\cdots,x_k)=\frac{y_k-C_k^1y_{k-1}+C_k^2y_{k-2}+\cdots(-1)^k y_0}{k!h^k}$$

Then, $f(x_1,x_2,\cdots,x_{k+1})=\dfrac{y_{k+1}-C_k^1y_k+C_k^2y_{k-1}+\cdots(-1)^k y_1}{k!h^k}$.

When $n=k+1$,

$$f(x_0,x_1,\cdots,x_k,x_{k+1})=\frac{f(x_1,\cdots,x_k,x_{k+1})-f(x_0,x_1,\cdots,x_k)}{(k+1)h}$$

$$=\frac{1}{(k+1)!h^{k+1}}\{[y_{k+1}-C_k^1y_k+C_k^2y_{k-1}+\cdots(-1)^ky_1]$$
$$-[y_k-C_k^1y_{k-1}+C_k^2y_{k-2}+\cdots(-1)^ky_0]\}$$
$$=\frac{y_{k+1}-C_{k+1}^1y_k+C_{k+1}^2y_{k-1}+\cdots(-1)^{k+1}y_0}{(k+1)!h^{k+1}}$$
$$=\frac{1}{(k+1)!h^{k+1}}\sum_{i=0}^{k+1}(-1)^iC_{k+1}^iy_{k+1-i}$$

Hence, formula is true for $n=k+1$. The formula, therefore, is true for every divided difference.

Because the interpolating polynomial is formed as above using the topmost entries of divided-difference table in each column as coefficients, we only calculate the topmost entries of divided-difference table, needn't calculate entire divided-difference table. So the computation is smaller than the traditional method.

IV. NUMERICAL EXAMPLE

Find interpolating polynomials $P_4(x)$ based on the five points (-4,30),(-2,-42),(0,30),(2,6) and (4,30).

This is an equidistant interpolation problem. The equal width h is $h=2$.

(1) Equidistant Lagrange's Interpolation

$$E_0=\frac{30}{(-1)^4\cdot 0!\cdot 4!\cdot 2^4}=\frac{5}{64}$$
$$E_1=\frac{-42}{(-1)^3\cdot 1!\cdot 3!\cdot 2^4}=\frac{7}{16}$$
$$E_2=\frac{30}{(-1)^2\cdot 2!\cdot 2!\cdot 2^4}=\frac{15}{32}$$
$$E_3=\frac{6}{(-1)^1\cdot 3!\cdot 1!\cdot 2^4}=-\frac{1}{16}$$
$$E_4=\frac{30}{(-1)^0\cdot 4!\cdot 0!\cdot 2^4}=\frac{5}{64}$$

$$P_4(x)=\frac{5}{64}(x+2)x(x-2)(x-4)$$
$$+\frac{7}{16}(x+4)x(x-2)(x-4)$$
$$+\frac{15}{32}(x+4)(x+2)(x-2)(x-4)$$
$$-\frac{1}{16}(x+4)(x+2)x(x-4)$$
$$+\frac{5}{64}(x+4)(x+2)x(x-2)$$
$$=x^4-x^3-16x^2+16x+30$$

$P_4(x)$ is shown in figure 1.

(2) Equidistant Newton's Interpolation

$$f(-4,-2)=\frac{y_1-y_0}{h}=\frac{-42-30}{2}=-36$$
$$f(-4,-2,0)=\frac{y_2-2y_1+y_0}{2h^2}$$
$$=\frac{30+84+30}{8}=18$$
$$f(-4,-2,0,2)=\frac{y_3-3y_2+3y_1+y_0}{3!h^3}$$
$$=\frac{6-90-126-30}{48}=-5$$
$$f(-4,-2,0,2,4)=\frac{y_4-4y_3+6y_2-4y_1+y_0}{4!h^4}$$
$$=\frac{30-24+180+168+30}{24\times 16}=1$$

Therefore,

$$P_4(x)=30-36(x+4)+18(x+4)(x+2)$$
$$-5(x+4)(x+2)x+(x+4)(x+2)x(x-2)$$
$$=x^4-x^3-16x^2+16x+30$$

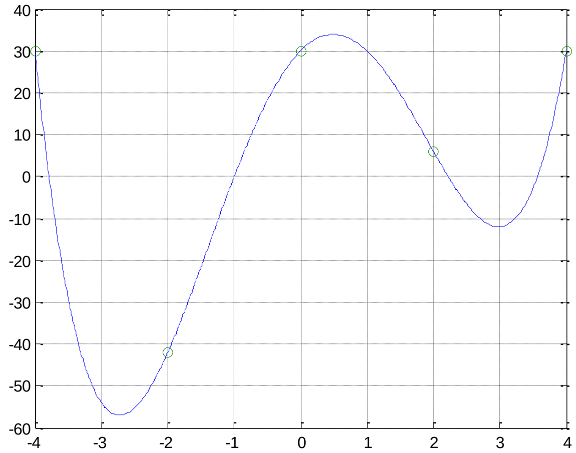

Figure 1. Graph of $P_4(x)$

ACKNOWLEDGMENT

This work was partially supported by Natural Science Foundation of Jiangsu Province of China (No. BK2012700), the Open Project Program of Key Laboratory of Intelligent Computing & Information Processing (Xiangtan University), Ministry of Education (No. 2011ICIP05), Artificial Intelligence of Key Laboratory of Sichuan Province（2012RYJ04）and the National Natural Science Foundation of China under Grant No. 61305058.

REFRENCES

[1] S. Gao, X. C. Bie and B. Qin. Computing Method. xidian university press, 2009. pp.33-46(in Chinese).

[2] R. L., Burden, and J. D. Faires, Numerical Analysis. Higher Education Press & Thomson Learning, Inc., 2001.

[3] J. H. Mathews, and K. D. Fink, Numerical Methods Using MATLAB. Publishing House of Electronics Industry, 2002.

[4] G. Q. Chen, J. J. Liu and L. Q. Wang, “Color image sharing method based on Lagrange's interpolating polynomial”, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7231, pp. 63-75, 2012.

[5] M. Hiroyuki and Y. Souhei, “Continuation power flow with the nonlinear predictor of the Lagrange's polynomial interpolation formula”, Proceedings of the IEEE Power Engineering Society Transmission and Distribution Conference, vol.2, no.ASIA PACIFIC, pp. 1133-1138, 2002.

[6] J. S. Wang, S. G. Cui, X. Q. Deng, X. S. Xu and Y. D. Li, “ A novel Associative Memory System based on Newton's Forward Interpolation”, Proceedings of the International Conference on Active Media Technology, pp. 352-357, 2003.

[7] K. Bahman, “Generalization of Taylor's theorem and Newton's method via a new family of determinantal interpolation formulas and its applications”, Journal of Computational and Applied Mathematics, vol.126, no.1-2, pp. 287-318, December 30, 2000.

[8] A. Cordero and J. R.Torregrosa, “On interpolation variants of Newton's method for functions of several variables”, Journal of Computational and Applied Mathematics, vol.234, no.1, pp. 34-43, May 1, 2010.

[9] F. Zhou, Z. Y. Huang,C. Y. Zhao,X. M. Wei and D. Y. Chen, Dayue, “Time-domain quasi-synchronous sampling algorithm for harmonic analysis based on Newton's interpolation”,IEEE Transactions on Instrumentation and Measurement, vol.60, no.8, pp.2804-2812, August 2011.

[10] D. Sukanta, B. Gautam and S. Prasid “A solution to CT saturation by Gregory Newton's backward interpolation”, International Journal of Emerging Electric Power Systems, vol.12, no. 6, 2011.