A STUDY OF CUBIC SPLINE INTERPOLATION

Kai Wang '14G*
Graduate Student, M.S. Program in Computer Science, Rivier University

Abstract

The paper is an overview of the theory of interpolation and its applications in numerical analysis. It specially focuses on cubic splines interpolation with simulations in Matlab™.

1 Introduction: Interpolation in Numerical Methods

Numerical data is usually difficult to analyze. For example, numerous data is obtained in the study of chemical reactions, and any function which would effectively correlate the data would be difficult to find. To this end, the idea of the interpolation was developed.

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 (see below). Interpolation provides a means of estimating of the value at the new data points within the range of parameters.

t	0	.5	1.0	1.5	2.0
y	0	.19	.26	.29	.31

What is the value *y* when *t*=1.25?

2 Types of Interpolations

There are several different interpolation methods based on the accuracy, how expensive is the algorithm of implementation, smoothness of interpolation function, etc.

2.1 Piecewise constant interpolation

This is the simplest interpolation, which allows allocating the nearest value and assigning it to the estimating point. This method may be used in the higher dimensional multivariate interpolation, because of its calculation speed and simplicity.

2.2 Linear interpolation

Linear interpolation takes two data points, say (x_a,y_a) and (x_b,y_b), and the interpolation function at the point (x, y) is given by the following formula:

$$y = y_a + (y_b - y_a)\frac{x - x_a}{x_b - x_a}$$

Linear interpolation is quick and easy, but not very precise. Below is the error estimate formula, where the error is proportional to the square of the distance between the data points.

ISSN 1559-9388 (online version), ISSN 1559-9396 (CD-ROM version).

$$\left|f(x)-g(x)\right| \le C(x_b - x_a)^2 \text{, where } C=\frac{1}{8}\max\left|g''(y)\right|,\ y\in\left[x_a, x_b\right].$$

Here $g(x)$ is the interpolating function, which is twice continuously differentiable.

2.3 Polynomial interpolation

Polynomial interpolation is a generalization of linear interpolation. It replaces the interpolating function with a polynomial of higher degree.

If we have n data points, there is exactly one polynomial of degree at most $n-1$ going through all the data points:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0.$$

The interpolation error is proportional to the distance between the data points to the power n.

$$\left|f(x)-g(x)\right| = \frac{(x-x_1)(x-x_2).....(x-x_n)}{n!} f^{(n)}(c) \text{, where c lies in } \left[x_1, x_n\right].$$

The interpolant is a polynomial and thus infinitely differentiable. With higher degree polynomial ($n > 1$), the interpolation error can be very small. So, we see that polynomial interpolation overcomes most of the problems of linear interpolation. However, polynomial interpolation also has some disadvantages. For example, calculating the interpolating polynomial is computationally expensive compared to linear interpolation.

Polynomial interpolation may exhibit oscillatory artifacts, especially at the end points (known as Runge's phenomenon). More generally, the shape of the resulting curve, especially for very high or low values of the independent variable, may be contrary to common sense. These disadvantages can be reduced by using spline interpolation or Chebyshev polynomials [1-3].

2.4 Spline interpolation

Spline interpolation is an alternative approach to data interpolation. Compare to polynomial interpolation using on single formula to correlate all the data points, spline interpolation uses several formulas; each formula is a low degree polynomial to pass through all the data points. These resulting functions are called *splines*.

Spline interpolation is preferred over polynomial interpolation because the interpolation error can be made small even when using low degree polynomials for the spline. Spline interpolation avoids the problem of Runge's phenomenon, which occurs when the interpolating uses high degree polynomials.

The mathematical model for spline interpolation can be described as following:

For $i = 1,\dots,n$ data points, interpolate between all the pairs of knots (x_{i-1}, y_{i-1}) and (x_i, y_i) with polynomials

$$y = q_i(x),\ i=1, 2,\dots,n.$$

The curvature of a function $y = f(x)$ is

$$k = \frac{y''}{(1+y'^2)^{\frac{3}{2}}}$$

As the spline will take a function (shape) more smoothly (minimizing the bending), both y' and y'' should be continuous everywhere and at the knots. Therefore:

$$q_i'(x_i) = q_{i+1}'(x_{i+1}) \text{ and } q_i''(x_i) = q_{i+1}''(x_{i+1}) \text{ for } i, \text{ where } 1 \le i \le n-1$$

This can only be achieved if polynomials of degree 3 or higher are used. The classical approach uses polynomials of degree 3, which is the case of *cubic splines*.

3 Cubic Spline Interpolation

The goal of cubic spline interpolation is to get an interpolation formula that is continuous in both the first and second derivatives, both within the intervals and at the interpolating nodes. This will give us a smoother interpolating function. The continuity of first derivative means that the graph $y = S(x)$ will not have sharp corners. The continuity of second derivative means that the radius of curvature is defined at each point.

3.1 Definition

Given the n data points $(x_1,y_1),\ldots,(x_n,y_n)$, where x_i are distinct and in increasing order. A cubic spline $S(x)$ through the data points $(x_1,y_1),\ldots,(x_n,y_n)$ is a set of cubic polynomials:

$$S_1(x) = y_1 + b_1(x-x_1) + c_1(x-x_1)^2 + d_1(x-x_1)^3 on [x_1, x_2]$$

$$S_2(x) = y_2 + b_2(x-x_2) + c_2(x-x_2)^2 + d_2(x-x_2)^3 on [x_2, x_3]$$

$$S_{n-1}(x) = y_{n-1} + b_{n-1}(x-x_{n-1}) + c_{n-1}(x-x_{n-1})^2 + d_{n-1}(x-x_{n-1})^3 on [x_{n-1}, x_n]$$

With the following conditions (known as *properties*):

a. $S_i(x_i) = y_i$ and $S_i(x_{i+1}) = y_{i+1}$ for i=1,…,n-1

   This property guarantees that the spline $S(x)$ interpolates the data points.

b. $S_{i-1}'(x_i) = S_i'(x_i)$ for i=2,…,n-1

   $S'(x)$ is continuous on the interval $[x_1, x_n]$; this property forces the slopes of neighboring parts to agree when they meet.

c. $S_{i-1}''(x_i) = S_i''(x_i)$ for i=2,…,n-1

$S''(x)$ is continuous on the interval $\left[x_1, x_n\right]$, which also forces the neighboring spline to have the same curvature, to guarantee the smoothness.

3.2 Construction of cubic spline

How to determine the unknown coefficients b_i, c_i, d_i of the cubic spline $S(x)$ so that we can construct it? Given S(x) is cubic spline that has all the properties as in the definition section 3.1,

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 on[x_i, x_{i+1}] \tag{1}$$

for i=1, 2,…,n-1

The first and second derivatives:

$$S_i'(x) = 3d_i(x - x_i)^2 + 2c_i(x - x_i) + b_i \tag{2}$$

$$S_i''(x) = 6d_i(x - x_i) + 2c_i \tag{3}$$

for i=1, 2,…,n-1

From the first property of cubic spline, $S(x)$ will interpolate all the data points, and we can have

$$S_i(x_i) = y_i\,.$$

Since the curve $S(x)$ must be continuous across its entire interval, it can be concluded that each sub-function must join at the data points

$$S_i(x_i) = S_{i-1}(x_i)$$

Therefore,

$$y_i = S_{i-1}(x_i)$$

$$S_{i-1}(x_i) = y_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1})^2 + d_{i-1}(x_i - x_{i-1})^3 \tag{4}$$

$$y_i = y_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1})^2 + d_{i-1}(x_i - x_{i-1})^3 \tag{5}$$

for i=2,3,…,n-1.

Letting $h = x_i - x_{i-1}$ in Eq. (5), we have:

$$y_i = y_{i-1} + b_{i-1}h + c_{i-1}h^2 + d_{i-1}h^3 \tag{6}$$

for i=2,3,…,n-1

Also, with properties 2 of cubic spline, the derivatives must be equal at the data points, that is

$$S'_{i-1}(x_i) = S'_i(x_i) \tag{7}$$

By Eq. (2), $S'_i(x_i) = b_i$, and

$$S'_{i-1}(x_i) = 3d_{i-1}(x_i - x_{i-1})^2 + 2c_{i-1}(x_i - x_{i-1}) + b_{i-1} \tag{8}$$

Therefore,

$$b_i = 3d_{i-1}(x_i - x_{i-1})^2 + 2c_{i-1}(x_i - x_{i-1}) + b_{i-1} \tag{9}$$

Again, letting h = $x_i - x_{i-1}$, we find:

$$b_i = 3d_{i-1}h^2 + 2c_{i-1}h + b_{i-1} \tag{10}$$

for i=2,3,…,n-1

From Eq. (3), $S''_i(x) = 6d_i(x - x_i) + 2c_i$, we have

$$S''_i(x_i) = 6d_i(x_i - x_i) + 2c_i \tag{11}$$

$$S''_i(x_i) = 2c_i$$

Since $S''_i(x)$ should be continuous across the interval, therefore $S''_{i-1}(x_i) = S''_i(x_i)$

$$S''_{i-1}(x_i) = 6d_{i-1}(x_i - x_{i-1}) + 2c_{i-1} \tag{12}$$

$$2c_i = 6d_{i-1}(x_i - x_{i-1}) + 2c_{i-1}$$

Letting $h = x_i - x_{i-1}$,

$$2c_i = 6d_{i-1}(x_i - x_{i-1}) + 2c_{i-1} \tag{13}$$

$$2c_i = 6d_{i-1}h + 2c_{i-1}$$

Simplified these equation above by substituting DD_i for $S''_i(x_i)$, from Eq. (11)

$$S''_i(x_i) = 2c_i, \; DD_i = 2c_i$$

$$c_i = \frac{DD_i}{2} \tag{14}$$

From Eq. (13):

$$2c_i = 6d_{i-1}h + 2c_{i-1}, \; 6d_{i-1}h = 2c_i - 2c_{i-1}$$

$$d_{i-1} = \frac{2c_i - 2c_{i-1}}{6h}$$, substitute c_i

$$d_{i-1} = \frac{DD_i - DD_{i-1}}{6h},$$

$$\text{or} \quad d_i = \frac{DD_{i+1} - DD_i}{6h} \tag{15}$$

From Eq. (6):

$$y_i = y_{i-1} + b_{i-1}h + c_{i-1}h^2 + d_{i-1}h^3$$

$$\text{or} \quad y_{i+1} = y_i + b_i h + c_i h^2 + d_i h^3$$

Therefore,

$$b_i = \frac{y_{i+1} - y_i - c_i h^2 - d_i h^3}{h} = \frac{y_{i+1} - y_i}{h} - (c_i h + d_i h^2)$$

Substitute c_i and d_i

$$b_i = \frac{y_{i+1} - y_i}{h} - h(\frac{2DD_i + DD_{i+1}}{6}) \tag{16}$$

Put these systems into matrix form as follow:

From Eq. (10),

$$3d_{i-1}h^2 + 2c_{i-1}h + b_{i-1} = b_i .$$

Substitute b_i, c_i d_i :

$$3(\frac{DD_i - DD_{i-1}}{6h})h^2 + 2\,\frac{DD_{i-1}}{2}\,h + \frac{y_i - y_{i-1}}{h} - h(\frac{2DD_{i-1} + DD_i}{6}) = \frac{y_{i+1} - y_i}{h} - h(\frac{2DD_i + DD_{i+1}}{6})$$

$$\frac{h}{6}(DD_{i-1} + 4DD_i + DD_{i+1}) = \frac{y_{i-1} - 2y_i + y_{i+1}}{h}$$

$$DD_{i-1} + 4DD_i + DD_{i+1} = 6(\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2}) \tag{17}$$

for i =2, 3,…, n-1

Transform into the matrix equation:

$$\begin{bmatrix} 1 & 4 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 4 & 1 \end{bmatrix} \begin{bmatrix} DD_1 \\ DD_2 \\ DD_3 \\ \vdots \\ DD_{n-1} \\ DD_n \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{bmatrix} \tag{18}$$

This system has n-2 rows and n columns, it is under-determined. In order to construct a unique cubic spline, two other conditions must be imposed upon the system.

3.3 Cubic spline interpolation types

3.3.1 Natural spline

There are several ways to add the two conditions. Let's review the first scenario, *Natural Spline*.

Natural-spline boundary conditions:

$$S_1''(x_1) = 0 \quad \text{and} \quad S_{n-1}''(x_n) = 0 \tag{19}$$

The Eq. (18) can be adapted accordingly to Eq. (20), because of DD_1=0, DD_n=0,

$$\begin{bmatrix} 4 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 4 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 4 \end{bmatrix} \begin{bmatrix} DD_2 \\ DD_3 \\ \vdots \\ DD_{n-1} \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{bmatrix} \tag{20}$$

This is a diagonal linear system of the form HM = V, which involves DD_2, DD_3,…, DD_{n-1}. This linear system in Eq. (20) is strictly diagonally dominant and has a unique solution. Then, using Eqs. (14), (15) and (16), coefficients $(b_i, c_i \; d_i)$ are determined. The cubic spline can be constructed.

For example, data points (0,0), (1,0.5), (2, 1.8), (3,1.5), using the Matalab™ code for the solutions above, construct the unique cubic spline (namely, natural spline) for the data points (see Fig. 1).

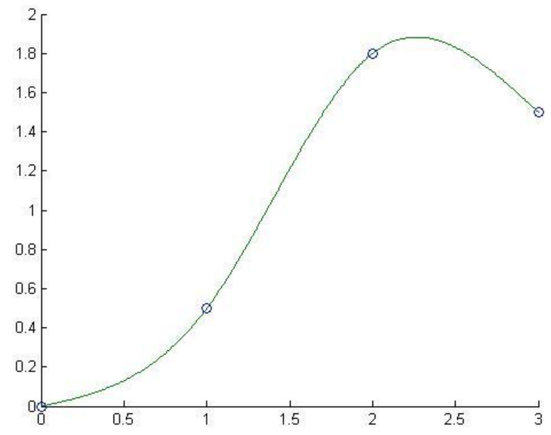

Figure 1. The Natural Cubic Spline.

3.3.2 Curvature-adjusted cubic spline

This type of spline is an alternative of natural spline, and sets $S_1''(x_1)$ and $S_{n-1}''(x_n)$ to arbitrary value. This value corresponds to desired curvature settings at both end points.

Boundary or Ending Points Conditions are: $S_1''(x_1) = v1$ and $S_{n-1}''(x_n) = vn$. The curvature-adjusted cubic spline is shown in Fig. 2 for the same data points (0,0), (1,0.5), (2, 1.8), (3,1.5), but with the conditions $S_1''(x_1) = v1 = 1, S_{n-1}''(x_n) = vn = 1$.

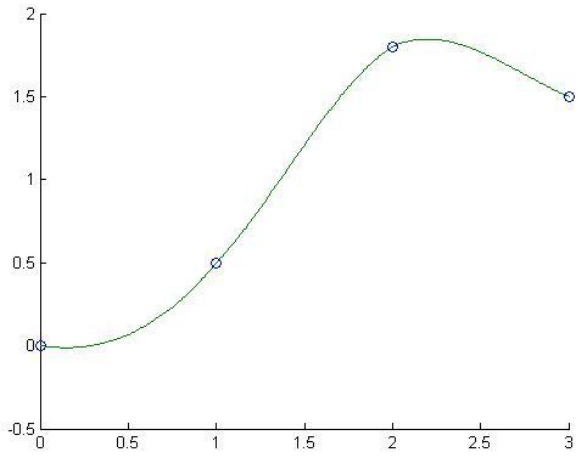

Figure 2. Curvature-Adjusted Cubic Spline.

3.3.3 Clamped Cubic Spline

This type of spline is set end point first derivatives $S_1'(x_1)$ and $S_n'(x_n)$ to certain value, thus the slope at the beginning and end of the spline are under user's control.

From Eq. (8) and Eq. (16), set $S_1'(x_1)$ =v1,

$$S_1'(x_1) = b_1 = \frac{y_2 - y_1}{h} - h(\frac{2DD_1 + DD_2}{6})$$

We can have:

$$DD_1 = \frac{6(y_2 - y_1) - 6hb_1 - h^2 DD_2}{2h^2} \qquad (21)$$

Also from equation 8 and 16, set $S_{n-1}'(x_n) = \text{vn} = b_{n-1}$,

$$DD_n = \frac{6(y_n - y_{n-1}) - 6hb_{n-1} - 2h^2 DD_{n-1}}{h^2} \qquad (22)$$

The clamped cubic spline is shown in Fig. 3 (below) for the same data points (0,0),(1,0.5), (2, 1.8), (3,1.5), and $S_1'(x_1) = 0.5$, $S_{n-1}'(x_n) = 0.5$.

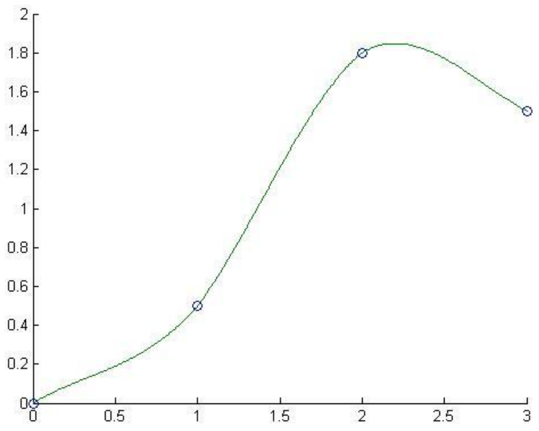

Figure 3. Clamped Cubic Spline.

3.3.4 Parabolically-Terminated Cubic Spline

For this type of spline, $S_1(x)$ and $S_{n-1}(x)$ are forced to be the functions of degree 2, which means that $d_1 = d_{n-1} = 0$.

From Eq. 15, $d_i = \dfrac{DD_{i+1} - DD_i}{6h}$, therefore:

$$d_1 = \frac{DD_2 - DD_1}{6h_1} = 0, \quad \text{and } DD_1 = DD_2$$

$$d_{n-1} = \frac{DD_n - DD_{n-1}}{6h_{n-1}} = 0, \text{ and } DD_n = DD_{n-1}$$

The parabolically-terminated cubic spline is shown in Fig. 4 (below) for the same data points (0,0),(1,0.5), (2, 1.8), (3,1.5).

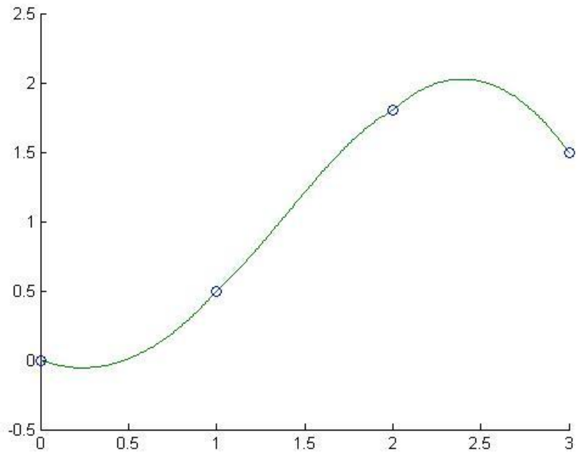

Figure 4 Parabolically-Terminated Cubic Spline

3.3.5 Not-a-Knot cubic spline

Two conditions $d_1 = d_2$, $d_{n-2} = d_{n-1}$ were added to construct the unique spline. $S_1(x)$ and $S_2(x)$ already agree at zeros, first and second derivatives. The condition $d_1 = d_2$ causes $S_1(x)$ and $S_2(x)$ to be identical cubic polynomials; thus the data-base point x_2 is not needed anymore. Same thing is for $S_{n-2}(x)$ and $S_{n-1}(x)$, and data point x_{n-1}.

From Eq. (15), $d_i = \dfrac{DD_{i+1} - DD_i}{6h}$, therefore:

$d_1 = \dfrac{DD_2 - DD_1}{6h_1}$, $d_2 = \dfrac{DD_3 - DD_2}{6h_2}$, and $d_1 = d_2$.

Hence, $DD_1 = DD_2 - \dfrac{h_1(DD_3 - DD_2)}{h_2}$.

Also, $d_{n-2} = d_{n-1}$,

$d_{n-1} = \dfrac{DD_n - DD_{n-1}}{6h_{n-1}}$, $d_{n-2} = \dfrac{DD_{n-1} - DD_{n-2}}{6h_{n-2}}$, and therefore:

$$DD_n = \frac{h_{n-1}(DD_{n-1} - DD_{n-2})}{h_{n-2}} + DD_{n-1}$$

The not-a-knot cubic spline is shown below in Fig. 5 for data points (0,0), (1,0.5), (2, 1.8), (3,1.5), (4,0.8), and n≥4.

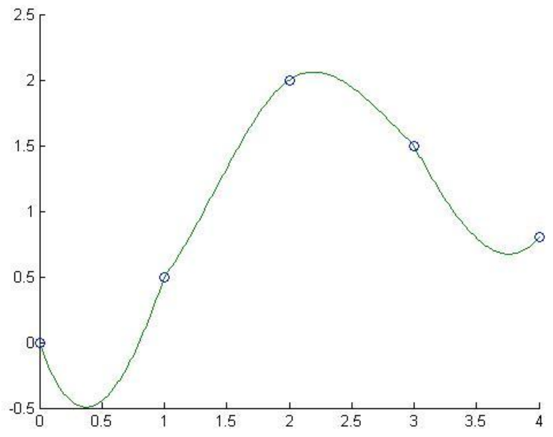

Figure 5. Not-a-Knot Cubic Spline.

4. Cubic Spline Applications

4.1 Representing functions by approximating polynomials

First, let's review the application of a cubic spline to approximate polynomials, or to evaluate a cubic spline at certain point within the given interval [a, b].

As an example, consider the polynomial function $f(x) = \frac{1}{x^2}\sin(x)$, on the interval [π/4, 3π/2]. We can take few coupled data points: (π/4, 1.1463), (π/2, 0.4053), (3π/4, 0.1274), (π, 0), (5π/4, -0.0459), (3π/2, -0.0450). With natural cubic spline, the interpolating function is shown as a solid line in Fig. 6 below. The exact values of the polynomial function $f(x)$ (red markers) are also shown in Fig. 6. Comparing the results, we can see that the cubic-spline interpolating function approximates the exact function with a good accuracy.

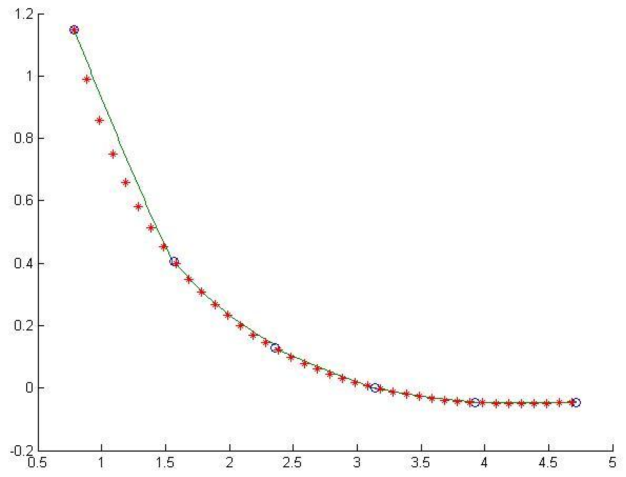

Figure 6. Comparing the Natural Cubic Spline Interpolating Function and the Exact Solution.

4.2 Data correlation and interpolation

Data interpolation is widely used in real world, especially with large volume of irregular data. Find polynomial functions guide these data within certain interval, can greatly help with data analysis and data predicting. Cubic splines could be used for finding an interpolation function to correlate data.

For example, in the chemical experiment, the following data was obtained (see arrays *t* and *D* below and Fig. 7):

t = [0	0.1	0 .499	0.5	0.6	1.0	1.4	1.5	1.899	1.9	2.0]
D = [0	0.06	0.17	0.19	0.21	0.26	0.29	0.29	0.30	0.31	0.31]

We need to estimate the value *D* when *t* = 1.2. Using the natural cubic spline interpolation (the formula $S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 on[x_i, x_{i+1}]$), we found *D* = 0.27527649.

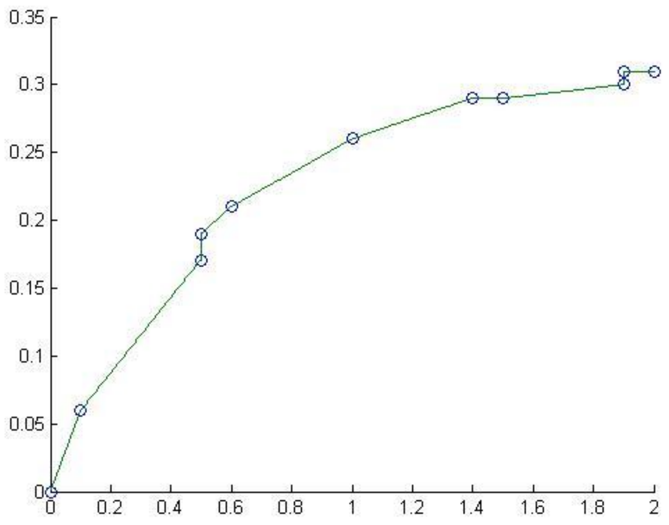

Figure 7. The Chemical Experiment Data, *D* = *f*(*t*).

Conclusions

In this paper, we have presented an overview of the methods of interpolation and especially the cubic spline interpolation, which is widely used in numerous real-world applications. The study presents the definition of cubic spline interpolation and properties of different types of cubic splines, including a natural spline, a curvature-adjusted spline, a clamped spline, a parabolically-terminated spline, and a not-a-knot spline. The spline construction formulas and corresponding Matlab™ codes were developed for different types of cubic splines with excellent accuracy of calculation. Two general applications of cubic spline interpolation were also analyzed.

References

1. Interpolation. Wikipedia. Retrieved October 22, 2013, from http://en.wikipedia.org/wiki/Interpolation

2. Spline Interpolation. Wikipedia. Retrieved October 22, 2013, from http://en.wikipedia.org/wiki/Spline_interpolation
3. Polynomial Interpolation. Wikipedia. Retrieved October 22, 2013, from http://en.wikipedia.org/wiki/Polynomial_interpolation
4. Mathews, J. H., and Fink, K. K. Chapter 5: Curve Fitting. In: Numerical Methods Using Matlab, 4th edition, Pearson, 2004.
5. Introduction to Plotting with Matlab. Math Sciences Computing Center, University of Washington, September, 1996. Retrieved October 22, 2013, from http://www.math.lsa.umich.edu/~tjacks/tutorial.pdf
6. Friedman, J., Hastie, T., and Tibshirani, R. Chapter5: Splines and Applications. In: The Elements of Statistical Learning. Retrieved October 22, 2013, from http://www.cs.ubbcluj.ro/~csatol/mach_learn/bemutato/BagyiIbolya_SplinesAndApplications.pdf
7. Grandine, T. A. The Extensive Use of Splines at Boeing. *SIAM News*, Vol. 38, No. 4, May 2005. Retrieved October 22, 2013, from http://www.me.ucsb.edu/~moehlis/ME17/splines.pdf
8. Cubic Spline Interpolation. Retrieved October 22, 2013, from http://www.mathworks.com/help/curvefit/cubic-spline-interpolation.html

Apendix A: Matlab™ Code for the Project

```
% Calculation of spline coefficients
% Calculates coefficients of cubic spline
% Input: x,y vectors of data points
%   plus two optional extra data v1, vn

% Output: matrix of coefficients b1,c1,d1;b2,c2,d2;...

function coeff=Myproject(x,y,v1,vn)
n=length(x);
A=zeros(n-2,n-2);         % matrix A is n-2xn-2
r=zeros(n-2,1);
h=zeros(n-1,1);

for i=1:n-1               % define the deltas
    h(i) = x(i+1)-x(i);
end

% load the A matrix
for i=2:n-2
    A(i,i-1)=1;
end
for i=1:n-2
     A(i,i)=4;
end
for i=1:n-3
    A(i,i+1)=1;
end

%fprintf('display Matrix A\n');
%disp(A);
```

```
for i=1:n-2
    r(i)=6*h(i)*(y(i)-2*y(i+1)+y(i+2)); % right-hand side
end

coeff=zeros(n-1,3);
DD=zeros(n,1);

% natural spline conditions  v1 vn;

%vn = 0;
%v1 = 0;

% curvature-adj conditions
%vn >0;
%v1 >0;

MDD=A\r;

for i=2:n-1
    DD(i)=MDD(i-1);
end

% natural spline conditions and curvature-adj conditions

DD(1)=v1;
DD(n)=vn;

%Clamped cubic spline

%coeff(1,1)=v1;
%coeff(n,1)=vn;
%DD(1)=(6*(y(2)-y(1))-6*h(1)*v1-h(1)^2*DD(2))/2*h(1)^2;
%DD(n)=(6*(y(n)-y(n-1))-6*h(n-1)*vn-2*h(n-1)^2*DD(n-1))/h(n-1)^2;

 % parabol-term conditions

 %DD(1)=DD(2);
 %DD(n)=DD(n-1);

 % not-a-knot

 %DD(1)=DD(2)-(h(1)/h(2))*(DD(3)-DD(2));
 %DD(n)=DD(n-1)+(h(n-1)/h(n-2))*(DD(n-1)-DD(n-2));

% solve for b, c d coefficients

for i=1:n-1

    coeff(i,1)=(y(i+1)-y(i))/h(i)-h(i)*(2*DD(i)+DD(i+1))/6;
    coeff(i,2)=DD(i)/2;
    coeff(i,3)=(DD(i+1)-DD(i))/6;
end
```

```
%calculate the certain value on interval [a,b], i=6

%xd=1.2;

%Dt=y(6)+coeff(6,1)*(xd-x(6))+coeff(6,2)*(xd-x(6))^2+coeff(6,3)*(xd-x(6))^3;

%fprintf('%12.8f\n',xd,Dt);

%plot the figure after calculation.

clf;hold on;             % clear figure window and turn hold on

for i=1:n-1
    x0=linspace(x(i),x(i+1),100);
    dx=x0-x(i);
    y0=coeff(i,3)*dx;    % evaluate using nested multiplication
    y0=(y0+coeff(i,2)).*dx;
    y0=(y0+coeff(i,1)).*dx+y(i);
    plot([x(i) x(i+1)],[y(i) y(i+1)],'o',x0,y0)
end

% plot the (1/x^2)*sin(x)

%hold on

%x1=pi/4:0.1:3*pi/2;
%y1=x1.^(-2).*sin(x1);
%plot(x1, y1,'r*');

hold off
end
```

* **KAI WANG** is a senior medical device and system product design engineer in Philips Healthcare Corp. He is a major contributor to several prime medical projects on software systems design, development, test and implementation in Philips Healthcare Corp. He completed Master Degree in Computer Science at Rivier University, Nashua, NH, in January 2014. He also holds B.S. Degree from Harbin University of Science and Technology and M.S. Degree from TianJin University.