


11-1-2014

# Cubic Spline Interpolation by Solving a Recurrence Equation Instead of a Tridiagonal Matrix

Peter Revesz

*University of Nebraska-Lincoln*, prevezs1@unl.edu

Follow this and additional works at: <http://digitalcommons.unl.edu/cseconfwork>

 Part of the [Computer Engineering Commons](#), [Electrical and Computer Engineering Commons](#), [Numerical Analysis and Scientific Computing Commons](#), and the [Other Computer Sciences Commons](#)

---

Revesz, Peter, "Cubic Spline Interpolation by Solving a Recurrence Equation Instead of a Tridiagonal Matrix" (2014). *CSE Conference and Workshop Papers*. 319.

<http://digitalcommons.unl.edu/cseconfwork/319>

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Conference and Workshop Papers by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

# Cubic Spline Interpolation by Solving a Recurrence Equation Instead of a Tridiagonal Matrix

Peter Z. Revesz

Department of Computer Science and Engineering

University of Nebraska-Lincoln

Lincoln, Nebraska 68588-0115

Email: revesz@cse.unl.edu

http://cse.unl.edu/revesz

Telephone: (1+) 402 472-3488

**Abstract**—The cubic spline interpolation method is probably the most widely-used polynomial interpolation method for functions of one variable. However, the cubic spline method requires solving a tridiagonal matrix-vector equation with an  $O(n)$  computational time complexity where  $n$  is the number of data measurements. Even an  $O(n)$  time complexity may be too much in some time-critical applications, such as continuously estimating and updating the flight paths of moving objects. This paper shows that under certain boundary conditions the tridiagonal matrix solving step of the cubic spline method could be entirely eliminated and instead the coefficients of the unknown cubic polynomials can be found by solving a single recurrence equation in much faster time.

## I. INTRODUCTION

Cubic spline interpolation is a widely-used polynomial interpolation method for functions of one variable [2]. Cubic splines can be described as follows. Let  $f$  be a function from  $\mathcal{R}$  to  $\mathcal{R}$ . Suppose we know about  $f$  only its value at locations  $x_0 < \dots < x_n$ . Let  $f(x_i) = a_i$ . Piecewise cubic spline interpolation of  $f$  is the problem of finding the  $b_i, c_i$  and  $d_i$  coefficients of the cubic polynomials  $S_i$  for  $0 \leq i \leq n-1$  written in the form:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \quad (1)$$

where each piece  $S_i$  interpolates the interval  $[x_i, x_{i+1}]$  and fits the adjacent pieces by satisfying certain smoothness conditions. Taking once and twice the derivative of Equation (1) yields, respectively the equations:

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

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

Equations (1-3) imply that  $S_i(x_i) = a_i$ ,  $S_i'(x_i) = b_i$  and  $S_i''(x_i) = 2c_i$ . For a smooth fit between the adjacent pieces the cubic spline interpolation requires that the following conditions hold for  $0 \leq i \leq n-2$ :

$$S_i(x_{i+1}) = S_{i+1}(x_{i+1}) = a_{i+1}, \quad (4)$$

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}) = b_{i+1} \quad (5)$$

$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}) = 2c_{i+1} \quad (6)$$

This paper is organized as follows. Section II review the usual solution for cubic splines by solving a tridiagonal matrix. Section ??

## II. THE TRIDIAGONAL MATRIX-BASED SOLUTION

In this section we review the usual tridiagonal matrix-based solution for cubic splines. Let  $h_i = x_{i+1} - x_i$ . Substituting Equations (1-3) into Equations (4-6), respectively, yields:

$$a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = a_{i+1} \quad (7)$$

$$b_i + 2c_i h_i + 3d_i h_i^2 = b_{i+1} \quad (8)$$

$$c_i + 3d_i h_i = c_{i+1} \quad (9)$$

Equation (9) yields a value for  $d_i$ , which we can substitute into Equations (7-8). Hence Equations (7-9) can be rewritten as:

$$a_{i+1} - a_i = b_i h_i + \frac{2c_i + c_{i+1}}{3} h_i^2 \quad (10)$$

$$b_{i+1} - b_i = (c_i + c_{i+1}) h_i \quad (11)$$

$$d_i = \frac{1}{3h_i} (c_{i+1} - c_i). \quad (12)$$

Solving Equation (10) for  $b_i$  yields:

$$b_i = (a_{i+1} - a_i) \frac{1}{h_i} - \frac{2c_i + c_{i+1}}{3} h_i \quad (13)$$

which implies for  $j \leq n-3$  the condition:

$$b_{i+1} = (a_{i+2} - a_{i+1}) \frac{1}{h_{i+1}} - \frac{2c_{i+1} + c_{i+2}}{3} h_{i+1} \quad (14)$$

Substituting into Equation (11) the values for  $b_i$  and  $b_{i+1}$  from Equations (13-14) yields:

$$(a_{i+1} - a_i) \frac{1}{h_i} - (2c_i + c_{i+1}) \frac{h_i}{3} + (c_i + c_{i+1})h_i = \\ (a_{i+2} - a_{i+1}) \frac{1}{h_{i+1}} - (2c_{i+1} + c_{i+2}) \frac{h_{i+1}}{3}$$

The above can be rewritten as:

$$h_i c_i + 2(h_i + h_{i+1})c_{i+1} + h_{i+1}c_{i+2} = \\ \frac{3}{h_i} a_i - \left( \frac{3}{h_i} + \frac{3}{h_{i+1}} \right) a_{i+1} + \frac{3}{h_{i+1}} a_{i+2}$$

The above holds for  $0 \leq i \leq n-3$ . However, changing the index downward by one the following holds for  $1 \leq j \leq n-2$ :

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = \\ \frac{3}{h_{i-1}} a_{i-1} - \left( \frac{3}{h_{i-1}} + \frac{3}{h_i} \right) a_i + \frac{3}{h_i} a_{i+1} \quad (15)$$

The above is a system of  $n-1$  linear equations for the unknowns  $c_i$  for  $0 \leq i \leq n$ . By Equation (3)  $S''_0(x_0) = 2c_0$  and by extending Equation (6) to  $j = n-1$ ,  $S''_{n-1}(x_n) = 2c_n$ .

The cubic spline interpolation allows us to specify several possible boundary conditions regarding the values of  $c_0, c_n$ . A commonly used boundary condition called a natural cubic spline assumes that  $c_0 = c_n = 0$ , which is equivalent to setting the second derivative of the splines at the ends to zero. Alternatively, in the clamped cubic spline interpolation, the assumed boundary condition is  $b_0 = f'(x_0)$  and  $b_n = f'(x_n)$  where the derivatives of the  $f$  at  $x_0$  and  $x_n$  are known constants.

In addition, in solving a cubic spline a uniform sampling is also commonly assumed and available, that is, each  $h_i$  has the same constant value  $h$ . Then dividing Equation (15) by  $h$  yields:

$$c_{i-1} + 4c_i + c_{i+1} = \frac{3}{h^2}(a_{i-1} - 2a_i + a_{i+1}) \quad (16)$$

Since the values of  $a_i$  are known, the values of  $c_i$  can be found by solving the tridiagonal matrix-vector equation  $Ax = B$ . Under the natural cubic spline interpolation, we have:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix}$$

the vector of unknowns is:

$$x = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}$$

and the vector of constants is:

$$B = \begin{bmatrix} 0 \\ \frac{3}{h^2}(a_0 - 2a_1 + a_2) \\ \vdots \\ \frac{3}{h^2}(a_{n-2} - 2a_{n-1} + a_n) \\ 0 \end{bmatrix}.$$

Similarly, under the clamped spline interpolation we have:

$$A = \begin{bmatrix} 2 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 2 \end{bmatrix}$$

the same vector of unknowns:

$$x = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}$$

and the following vector of constants:

$$B = \begin{bmatrix} \frac{3}{h^2}(a_1 - a_0) - \frac{3}{h}f'(x_0) \\ \frac{3}{h^2}(a_0 - 2a_1 + a_2) \\ \vdots \\ \frac{3}{h^2}(a_{n-2} - 2a_{n-1} + a_n) \\ \frac{3}{h}f'(x_n) - \frac{3}{h^2}(a_n - a_{n-1}) \end{bmatrix}.$$

Both the natural cubic spline and the clamped cubic spline boundary conditions yield a system of  $n+1$  linear equations with only  $n+1$  unknowns. Such a system normally yields a unique solution except in some special cases. Moreover, either system is a tridiagonal matrix system that can be solved in  $O(n)$  time. Once the  $c_i$  values are found, the  $d_i$  and the  $b_i$  values also can be found by Equations (12) and (13), respectively. Computing the  $b_i$  and  $d_i$  coefficients can be done also within  $O(n)$  time.

### III. A NEW RECURRENCE EQUATION-BASED SOLUTION

In our solution to the cubic spline interpolation problem, we chose a boundary condition that requires solving the following tridiagonal system where  $x_i$  are rational variables,  $d_i$  are rational constants and  $r \neq 0$  is a rational constant, and  $A$  is:

$$A = \begin{bmatrix} r & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Furthermore,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n-1} \\ e_n \end{bmatrix}.$$

*A. Relationship to Clamped and Natural Cubic Splines*

Our new matrix is closely related to clamped cubic splines. Consider the first equation for the clamped cubic spline, which can be written as:

$$2c_0 + c_1 = \frac{3}{h} \left( \frac{(a_1 - a_0)}{h} - f'(x_0) \right)$$

The above equation becomes the following after multiplying by  $r/2$ :

$$rc_0 + \frac{r}{2}c_1 = \frac{3r}{2h} \left( \frac{(a_1 - a_0)}{h} - f'(x_0) \right)$$

Adding  $(1 - r/2)c_1$  yields:

$$rc_0 + c_1 = \frac{3r}{2h} \left( \frac{(a_1 - a_0)}{h} - f'(x_0) \right) + \left( 1 - \frac{r}{2} \right) c_1$$

Hence the first row of our new matrix  $A$  is equivalent to first row of the clamped cubic spline for any  $r \neq 0$  if  $e_1$  is:

$$e_1 = \frac{3r}{2h} \left( \frac{(a_1 - a_0)}{h} - f'(x_0) \right) + \left( 1 - \frac{r}{2} \right) \tilde{c}_1.$$

where  $\tilde{c}_1$  is an estimate for the value of  $c_1$ .

The last row of the new matrix allows fixing the value of  $c_n$ . This is a generalization of natural cubic spline which fixes the value to be 0.

*B. A Recurrence Equation-Based Solution*

In this section, we solve the new system using the value  $r = 2 + \sqrt{3} \approx 3.732$ . In that case, the first three equations can be written as:

$$rx_1 + x_2 = e_1$$

$$x_1 + 4x_2 + x_3 = e_2$$

$$x_2 + 4x_3 + x_4 = e_3$$

Multiplying the second row by  $r$ , subtracting from it the first row, and then dividing it by  $r$  gives:

$$rx_1 + x_2 = e_1$$

$$rx_2 + x_3 = e_2 - \frac{e_1}{r}$$

$$x_2 + 4x_3 + x_4 = e_3$$

Multiplying now the third row by  $r$ , subtracting from it the second row, and then dividing it by  $r$  gives:

$$rx_1 + x_2 = e_1$$

$$rx_2 + x_3 = e_2 - \frac{e_1}{r}$$

$$rx_3 + x_4 = e_3 - \frac{e_2}{r} + \frac{e_1}{r^2}$$

Continuing this process until the last row, we get:

$$rx_{n-3} + x_{n-2} = e_{n-3} - \frac{e_{n-4}}{r} + \frac{e_{n-5}}{r^2} - \dots + (-1)^{n-4} \frac{e_1}{r^{n-4}}$$

$$rx_{n-2} + x_{n-1} = e_{n-2} - \frac{e_{n-3}}{r} + \frac{e_{n-4}}{r^2} - \dots + (-1)^{n-3} \frac{e_1}{r^{n-3}}$$

$$rx_{n-1} + x_n = e_{n-1} - \frac{e_{n-2}}{r} + \frac{e_{n-3}}{r^2} - \dots + (-1)^{n-2} \frac{e_1}{r^{n-2}}$$

$$x_n = e_n$$

Dividing each row except the last one by  $r$  yields:

$$x_{n-3} + \frac{x_{n-2}}{r} = \frac{e_{n-3}}{r} - \frac{e_{n-4}}{r^2} + \dots + (-1)^{n-4} \frac{e_1}{r^{n-3}}$$

$$x_{n-2} + \frac{x_{n-1}}{r} = \frac{e_{n-2}}{r} - \frac{e_{n-3}}{r^2} + \frac{e_{n-4}}{r^3} - \dots + (-1)^{n-3} \frac{e_1}{r^{n-2}}$$

$$x_{n-1} + \frac{x_n}{r} = \frac{e_{n-1}}{r} - \frac{e_{n-2}}{r^2} + \frac{e_{n-3}}{r^3} - \dots + (-1)^{n-2} \frac{e_1}{r^{n-1}}$$

$$x_n = e_n$$

Note that each row  $1 \leq i \leq n - 1$  will be the following:

$$x_i + \frac{x_{i-1}}{r} = \sum_{0 \leq k \leq (i-1)} (-1)^k \frac{e_{i-k}}{r^{k+1}}$$

We define the values for  $\alpha_0, \alpha_i$  for  $1 < i \leq n - 1$ , and  $\alpha_n$ , respectively, as follows:

$$\alpha_0 = 0$$

$$\alpha_i = \frac{e_i - \alpha_{i-1}}{r} = \sum_{0 \leq k \leq (i-1)} (-1)^k \frac{e_{i-k}}{r^{k+1}}$$

$$\alpha_n = e_n \tag{17}$$

The solution to the linear equation system can be described in terms of the  $\alpha$  constants as follows:

$$\begin{aligned} & \vdots \\ x_{n-3} &= \alpha_{n-3} - \frac{\alpha_{n-2}}{r} + \frac{\alpha_{n-1}}{r^2} - \frac{\alpha_n}{r^3} \\ x_{n-2} &= \alpha_{n-2} - \frac{\alpha_{n-1}}{r} + \frac{\alpha_n}{r^2} \\ x_{n-1} &= \alpha_{n-1} - \frac{\alpha_n}{r} \\ x_n &= \alpha_n \end{aligned}$$

Therefore,  $x_i$  for each row  $1 \leq i \leq n$  will be:

$$\begin{aligned} x_n &= \alpha_n \\ x_i &= \alpha_{i-1} - \frac{x_{i+1}}{r} \end{aligned} \tag{18}$$

The above can be solved in closed form as follows:

$$x_i = \sum_{0 \leq k \leq (n-i)} \left(\frac{-1}{r}\right)^k \alpha_{i+k} \tag{19}$$

Note that no matter what exactly are the initial values for  $e$ , we have pre-solved the system. This can lead to a faster evaluation of the cubic spline than solving the tridiagonal system each time. We need only  $O(n)$  multiplications and subtractions to compute the values of all the  $x_i$ . Moreover, when any new measurement is made, the conventional tridiagonal matrix-based algorithm requires a complete redo of the entire computation in  $O(n)$  time. In contrast, Equation (18) leads to a faster update because to each  $x_i$  for  $i \leq n$  we need to add only the term:

$$\left(\frac{-1}{r}\right)^{n+1-i} \alpha_{n+1}.$$

We also need to make  $x_{n+1} = \alpha_{n+1}$ . Afterward updating the other  $\alpha_i$  constants can be done also similarly efficiently.

### C. A Moving Object Example

Suppose that an object is released from a height of 400 feet with zero initial velocity. Suppose also that we measure the object's position to be 384, 336 and 256 feet from earth at one, two and three seconds after release. We also suspect that the object is in free fall with a gravitational acceleration of  $32ft/sec^2$  at one second after release and at three seconds after release. Find a cubic spline approximation for the object's position at all times from the release to three seconds after.

We will measure the distance traveled from the release point. The cubic polynomials we need to find for the intervals  $[0, 1]$ ,  $[1, 2]$  and  $[2, 3]$  can be expressed as follows:

$$\begin{cases} S_0(x) = a_0 + b_0x + c_0x^2 + d_0x^3 \\ S_1(x) = a_1 + b_1(x-1) + c_1(x-1)^2 + d_1(x-1)^3 \\ S_2(x) = a_2 + b_2(x-2) + c_2(x-2)^2 + d_2(x-2)^3 \end{cases}$$

We have  $n = 4$ ,  $a_0 = 400$ ,  $a_1 = 384$ ,  $a_2 = 336$ ,  $a_3 = 256$  and the uniform step size is  $h = 1$ . By our assumptions of zero initial velocity  $f'(0) = 0$  and free fall at one second  $c_1 = -16$  and free fall at four seconds  $c_3 = -16$ , which implies  $e_4 = -16$ . The matrix  $A$  and the vectors  $x$  and  $B$  are:

$$A = \begin{bmatrix} r & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad x = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} -16r - 16 \\ -96 \\ -96 \\ -16 \end{bmatrix}$$

$$\text{because } B = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = \begin{bmatrix} \frac{3r}{2}(-16) + \left(1 - \frac{r}{2}\right)(-16) \\ 3(400 - (2 \times 384) + 336) \\ 3(384 - (2 \times 336) + 256) \\ -16 \end{bmatrix}$$

By Equation (17), we have:

$$\alpha_1 = \frac{e_1}{r} = -16 - \frac{16}{r}$$

$$\alpha_2 = \frac{e_2 - \alpha_1}{r} = -16 - \frac{16}{r}$$

$$\alpha_3 = \frac{e_3 - \alpha_2}{r} = -16 - \frac{16}{r}$$

$$\alpha_4 = e_4 = -16$$

By Equation (18) we also have when calculating in reverse order:

$$c_3 = \alpha_4 = -16$$

$$c_2 = \alpha_3 - \frac{c_3}{r} = -16$$

$$c_1 = \alpha_2 - \frac{c_2}{r} = -16$$

$$c_0 = \alpha_1 - \frac{c_1}{r} = -16$$

Solving for the  $b_i$  coefficients by Equation (13) gives:

$$b_0 = \frac{1}{1}(384 - 400) - \frac{1}{3}(-16 - 32) = 0$$

$$b_1 = \frac{1}{1}(336 - 384) - \frac{1}{3}(-16 - 32) = -32$$

$$b_2 = \frac{1}{1}(256 - 336) - \frac{1}{3}(-16 - 32) = -64$$

Solving for the  $d_i$  coefficients by Equation (12) gives:

$$d_0 = \frac{1}{3}(-16 - (-16)) = 0$$

$$d_1 = \frac{1}{3}(-16 - (-16)) = 0$$

$$d_2 = \frac{1}{3}(-16 - (-16)) = 0$$

The above values show that an object in free fall has an increasing velocity but its acceleration remains constant. Using the above values, the cubic spline interpolation can be described as:

$$\begin{cases} S_0(x) = 400 - 16x^2 \\ S_1(x) = 384 - 32(x-1) - 16(x-1)^2 = 400 - 16x^2 \\ S_2(x) = 336 - 64(x-2) - 16(x-2)^2 = 400 - 16x^2 \end{cases}$$

Hence in each piece the cubic spline interpolation gives  $400 - 16x^2$ , which agrees with the expected physics equation for the position of a moving object that starts with zero velocity from an elevation of 400 feet and freely falls downward with an acceleration of  $32ft/sec^2$ .

#### IV. CONCLUSION

The general method described in this paper can be used in a wide variety of applications which require interpolation of a function of one variable. For example, interpolation of measurement data can generate constraint databases that can be efficiently queried using constraint query languages [5], [6]. The simple one-variable function interpolation can be also extended to higher dimensions yielding interpolations of higher-dimensional functions that describe surfaces [4] and three-dimensional spatio-temporal or moving objects [1], [3]. This extension remains an interesting future work.

#### REFERENCES

- [1] S. Anderson and P. Z. Revesz, Efficient MaxCount and threshold operators of moving objects, *Geoinformatica*, 13 (4), 355-396, 2009.
- [2] R. L. Burden and J. D. Faires, *Numerical Analysis*, 9th ed. New York, USA: Springer, 2014.
- [3] J. Chomicki and P. Z. Revesz, Constraint-based interoperability of spatiotemporal databases, *Geoinformatica*, 3 (3), 211-243, 1999.
- [4] L. Li and P. Z. Revesz, Interpolation methods for spatio-temporal geographic data, *Computers, Environment and Urban Systems*, 28 (3), 201-227, 2004.
- [5] P. C. Kanellakis, G. M. Kuper and P. Z. Revesz, Constraint query languages, *Journal of Computer and System Sciences*, 51 (1), 26-52, 1995.
- [6] P. Z. Revesz, *Introduction to Databases: From Biological to Spatio-Temporal*, New York, USA: Springer, 2010.