An Efficient Cubic Spline Interpolation Scheme for Wireless Communication Systems

Byung-Jae Kwak
Terrestrial & Non-Terrestrial Integrated Telecommunications Research Laboratory
ETRI
Daejeon, Korea
bjkwak@etri.re.kr

Jin Kyeong Kim
Terrestrial & Non-Terrestrial Integrated Telecommunications Research Laboratory
ETRI
Daejeon, Korea
jkkim@etri.re.kr

Keunyoung Kim
Terrestrial & Non-Terrestrial Integrated Telecommunications Research Laboratory
ETRI
Daejeon, Korea
kykim12@etri.re.kr

Young-Jo Ko
Terrestrial & Non-Terrestrial Integrated Telecommunications Research Laboratory
ETRI
Daejeon, Korea
koyj@etri.re.kr

***Abstract*— We propose a cubic spline interpolation scheme with low complexity for wireless communication systems. The proposed scheme is significantly more efficient in terms of computational complexity compared to the conventional interpolation schemes based on FFT/IFFT, and allows arbitrary time precision. The performance of the proposed scheme is evaluated using an example.**

Keywords—interpolation, Cubic spline, Wireless communication systems, Semantic Communication

I. Introduction

In wireless communication systems, interpolation of a discrete-time signal is often required to perform various functions of the system (e.g., finding the maximum of cross-correlation signal when the maximum is between data samples).

A. The problem

Given discrete-time samples $y_n = y(nT_S)$, $n = \cdots, -2, -1, 0, 1, 2, \cdots$, of a signal $y(t)$, find $y(t)$ at $t = mT_S + \tau$, where m is an arbitrary integer and $0 < \tau < 1$. For simplicity of notation, without loss of generality, we assume $T_S = 1$.

The most common method is to use fast algorithms such as FFT/IFFT to perform interpolation [1]. However, using FFT/IFFT for interpolation has the following problems:

- The number of samples (the number of FFT points) has to be a power of 2.
- The computational cost is proportional to the number of FFT points.
- Interpolation for arbitrary $0 < \tau < 1$ is impossible.

In this paper, we propose an interpolation scheme based on cubic spine to address these restrictions.

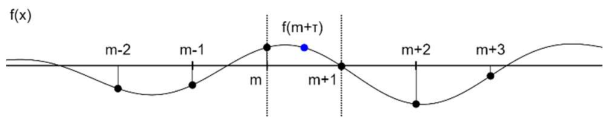

Fig. 1. Finding an interpolation sample between y_m and y_{m+1}

B. Assumptions

To make the problem simpler, a number of assumptions are made.

- Uniform sampling interval: Cubic spline interpolation is a versatile operation and works equally well for non-uniform sampling intervals. However, the discrete-time signals found in wireless communications are uniformly sampled signals, and thus we focus on signals with uniform sampling intervals.
- Six consecutive samples are available, where 3 samples are on the left-hand side of the interval of interest and three samples are on the right-hand side of the interval of interest as illustrated in Fig. 1.

II. Cubic Spline Interpolation

A. Spline segmentation of a curve

Cubic spline interpolation works by first dividing a curve into segments and then finding a cubic polynomial that best fits for each segment [2]. In this paper, we assume all segments are of equal length because that is the case in all wireless communication systems.

Fig. 2 a) shows an example of a conventional segmentation scheme of $f(x)$ with three segments and 4 control points $y_i = f(m-1+i), i = 0, 1, \cdots, 3$.

- Segment 0: $x \in [m-1, m]$
- Segment 1: $x \in [m, m+1]$
- Segment 2: $x \in [m+1, m+2]$

The curve $f(x)$ can be constructed by piecing the segments together. Let $f_0(x)$, $f_1(x)$, and $f_2(x)$ be segment 0, 1, and 2 of $f(x)$, respectively, then $f(x)$ can be written in terms of the segments as follows.

$$f(x) = f_0(x-(m-1)) + f_0(x-m) + f_0(x-(m+1))$$

where $m-1 \le x \le m+2$, and

$$f_0(x) = a_0 + b_0 x + c_0 x^2 + d_0 x^3, \quad 0 \le x \le 1$$
$$f_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3, \quad 0 \le x \le 1$$
$$f_2(x) = a_2 + b_2 x + c_2 x^2 + d_2 x^3, \quad 0 \le x \le 1$$

Note that the segments $f_0(x)$, $f_1(x)$, and $f_2(x)$ are defined only for $0 \le x \le 1$, and are zero elsewhere.

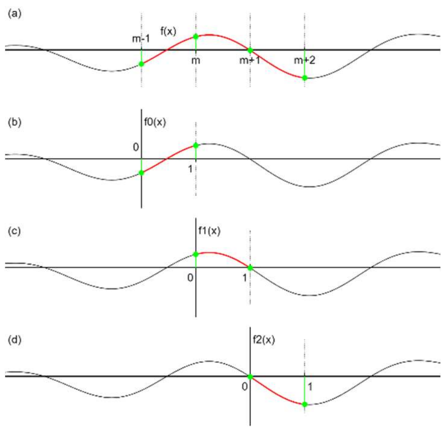

Fig. 2. Segmentation in the original cubic spline interpolation.

There are $4N = 4 \times 3 = 12$ coefficients to solve for, where N is the number of segments. The coefficients can be calculated by solving a system of $4N$ linear equations obtained from the following continuity conditions.

- C^0 continuity: Each segment must pass their control points (i.e., $f_i(0) = y_i$, $f_i(1) = y_{i+1}$, $i = 0, 1, 2$, when $N = 3$), from which we obtain $2N = 6$ equations.
- C^1 continuity: The segments must have the same slope where they meet. That is, $f_i'(1) = f_{i+1}'(0)$, $i = 0, 1$, from which we obtain $N - 1 = 2$ equations.
- C^2 continuity: The segments must have the same curvature where they meet. That is, $f_i''(1) = f_{i+1}''(0)$, $i = 0, 1$, from which we obtain $N - 1 = 2$ equations.
- Two more equations: The missing 2 equations can be obtained by providing first derivative at either ends of the curve. That is, $f_0'(0) = S_0$ and $f_2'(1) = S_1$.

In this conventional scheme, S_0 and S_1 are not known. To address this problem, the proposed scheme uses two more control points as illustrated in Fig. 3, making it possible to apply cubic spline interpolation without the knowledge of S_0 and S_1. Fig. 3 (a) shows a curve $f(x)$ with three segments and 6 control points $y_i = f(m - 1 + i)$, $i = -1, 0, \cdots, 4$.

Similarly to the conventional cubic spline interpolation, the curve $f(x)$ can be constructed by piecing the segments together. Let $f_0(x)$, $f_1(x)$, and $f_2(x)$ be segment 0, 1, and 2 of $f(x)$, respectively, then $f(x)$ can be written in terms of the segments as follows.

$f(x) = f_0(x - (m-1)) + f_0(x - m) + f_0(x - (m+1))$ where $m - 2 \le x \le m + 3$, and

$$\begin{aligned} f_0(x) &= a_0 + b_0x + c_0x^2 + d_0x^3, \quad -1 \le x \le 1 \\ f_1(x) &= a_1 + b_1x + c_1x^2 + d_1x^3, \quad 0 \le x \le 1 \\ f_2(x) &= a_2 + b_2x + c_2x^2 + d_2x^3, \quad 0 \le x \le 2 \end{aligned} \tag{1}$$

Note that the segment $f_0(x)$ is defined for $-1 \le x \le 1$, the segment $f_1(x)$ is defined for $0 \le x \le 1$, and the segment $f_2(x)$ is defined for $0 \le x \le 2$, and are zero elsewhere.

There are $4N = 4 \times 3 = 12$ coefficients to solve for, where N is the number of segments. The coefficients can be calculated by solving a system of $4N$ linear equations obtained from the following continuity conditions.

- C^0 continuity: Each segment must pass their control points (i.e., $f_0(-1) = y_{-1}$, $f_2(2) = y_4$, and $f_i(0) = y_i$, $f_i(1) = y_{i+1}$, $i = 0, 1, 2$, when $N = 3$), from which we obtain $2N + 2 = 8$ equations.
- C^1 continuity: The segments must have the same slope where they meet. That is, $f_i'(1) = f_{i+1}'(0)$, $i = 0, 1$, from which we obtain $N - 1 = 2$ equations.
- C^2 continuity: The segments must have the same curvature where they meet. That is, $f_i''(1) = f_{i+1}''(0)$, $i = 0, 1$, from which we obtain $N - 1 = 2$ equations.

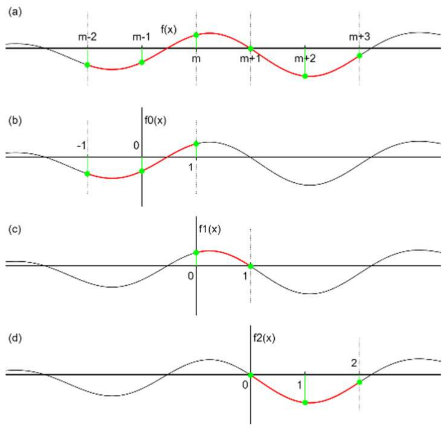

Fig. 3. Segmentation in the proposed cubic spline interpolation

B. System of linear equations

Let y_i, $i = -1, 1, \cdots, 4$, be the 6 discrete-time samples of $f(x)$. Then, from the continuity conditions, we obtain

$$\begin{aligned} f_0(-1) &= y_{-1} \\ f_0(0) &= y_0 \\ f_0(1) &= y_1 \\ f_0'(1) &= f_1'(0) \\ f_0''(1) &= f_1''(0)) \\ f_1(0) &= y_1 \\ f_1(1) &= y_2 \\ f_1'(1) &= f_2'(0) \\ f_1''(1) &= f_2''(0) \\ f_2(0) &= y_2 \\ f_2(1) &= y_3 \\ f_2(2) &= y_4 \end{aligned} \tag{2}$$

By substituting (1) into (2), (2) can be written in matrix form as follows.

$$\begin{bmatrix} a_0 \\ b_0 \\ c_0 \\ d_0 \\ a_1 \\ b_1 \\ c_1 \\ d_1 \\ a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix} = A^{-1} \begin{bmatrix} y_{-1} \\ y_0 \\ y_1 \\ 0 \\ 0 \\ y_1 \\ y_2 \\ 0 \\ 0 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{3}$$

where

$$A^{-1}=\frac{1}{90}\begin{bmatrix} 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -26 & -62 & 88 & -24 & -7 & 30 & -30 & 6 & 2 & -7 & 8 & -1 \\ 45 & -90 & 45 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -19 & 62 & -43 & 24 & 7 & -30 & 30 & -6 & -2 & 7 & -8 & 1 \\ 0 & 0 & 0 & 0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & -56 & 49 & -42 & 14 & -60 & 60 & -12 & -4 & 14 & -16 & 2 \\ -12 & 96 & -84 & 72 & -24 & -90 & 90 & -18 & -6 & 21 & -24 & 3 \\ 5 & -40 & 35 & -30 & 10 & 60 & -60 & 30 & 10 & -35 & 40 & -5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 90 & 0 & 0 \\ -2 & 16 & -14 & 12 & -4 & -60 & 60 & -48 & 14 & -49 & 56 & -7 \\ 3 & -24 & 21 & -18 & 6 & 90 & -90 & 72 & -21 & -84 & 96 & -12 \\ -1 & 8 & -7 & 6 & -2 & -30 & 30 & -24 & 7 & 43 & -62 & 19 \end{bmatrix}$$

Note that A^{-1} can be written in terms of integers, which makes implementation of the scheme much easier.

Since we are only interested in $\{a_1, b_1, c_1, d_1\}$, equation (3) can be simplified as follows:

$$\begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \end{bmatrix} = \mathcal{A}\begin{bmatrix} y_{-1} \\ y_0 \\ y_1 \\ y_1 \\ y_2 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \quad (4)$$

where $\mathcal{A}$ is a (4×8) matrix whose elements are obtained from rows 4, 5, 6, 7 and columns 0, 1, 2, 5, 6, 9, 10, 11 of matrix A^{-1}. That is,

$$\mathcal{A}=\frac{1}{90}\begin{bmatrix} 0 & 0 & 0 & 90 & 0 & 0 & 0 & 0 \\ 7 & -56 & 49 & -60 & 60 & 14 & -16 & 2 \\ -12 & 96 & -84 & -90 & 90 & 21 & -24 & 3 \\ 5 & -40 & 35 & 60 & -60 & -35 & 40 & -5 \end{bmatrix}$$

Since y_1 and y_2 each appears twice in (4), (4) can be further simplified as follow.

$$\begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \end{bmatrix} = \mathcal{B}\begin{bmatrix} y_{-1} \\ y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \quad (5)$$

where

$$\mathcal{B}=\frac{1}{90}\begin{bmatrix} 0 & 0 & 90 & 0 & 0 & 0 \\ 7 & -56 & -11 & 74 & -16 & 2 \\ -12 & 96 & -174 & 111 & -24 & 3 \\ 5 & -40 & 95 & -95 & 40 & -5 \end{bmatrix}$$

Note that the matrix $\mathcal{B}$ does not depend on the data samples and can be calculated in advance. Also note that the inner product of the first row of $\mathcal{B}$ with another vector requires only a single scalar multiplication.

C. Calculating intermediate points

When 6 discrete-time samples are given as follows

$$\{y_{-1} = f(m-2), y_0 = f(m-1), \cdots, y_4 = f(m+3)\}$$

from (5), the interpolation point $f(m+\tau)$ can be found by

$$f(m+\tau) = a_1 + b_1\tau + c_1\tau^2 + d_1\tau^3 = \begin{bmatrix} 1 & \tau & \tau^2 & \tau^3 \end{bmatrix} \mathcal{B}\begin{bmatrix} y_{-1} \\ y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

where $0 < \tau < 1$. In general, if M interpolation points are required for $\{\tau_1, \tau_2, \cdots, \tau_M\}$, the interpolation points can be obtained by

$$\begin{bmatrix} f(m+\tau_1) \\ f(m+\tau_2) \\ \vdots \\ f(m+\tau_M) \end{bmatrix} = \begin{bmatrix} 1 & \tau_1 & \tau_1^2 & \tau_1^3 \\ 1 & \tau_2 & \tau_2^2 & \tau_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \tau_M & \tau_M^2 & \tau_M^3 \end{bmatrix} \mathcal{B}\begin{bmatrix} y_{-1} \\ y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

III. Evaluation of the Proposed Interpolation Scheme

Suppose we are trying to estimate the time of arrival of a synchronization signal, and assume that the cross-correlator output is an ideal sinc function oversampled by factor 2:

$$f(n) = \operatorname{sinc}\left(\frac{n - t_0}{2}\right)$$

where $\operatorname{sinc}(t) = \frac{\sin \pi t}{\pi t}$ is a normalized sinc function, and t_0 is the arrival time normalized by the sampling interval T_S. The objective is to estimate t_0 with high accuracy.
$f(n)$ is illustrated in Fig. 4

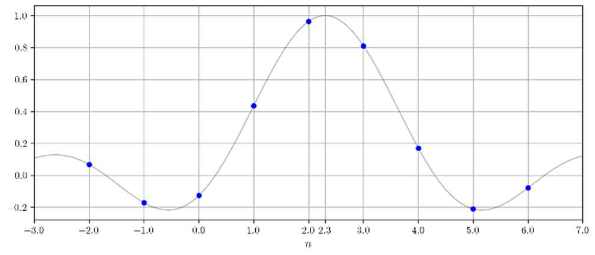

Fig. 4. The cross-correlator output signal $f(n)$.

Without interpolation, the estimated time of arrival is $\hat{t}_0 = 2$. For a more accurate estimation of the arrival time, we perform interpolation with up-sampling factor 10 using the proposed scheme, whose result is illustrated in Fig. 5. With interpolation, the peak is 0.9954 at $\hat{t}_0 = 2.3$. Note that the interpolation factor 10 is not a power of 2. The interpolation factor 2 was chosen to demonstrate that the interpolation factor is not limited to a power of 2 in the proposed scheme.

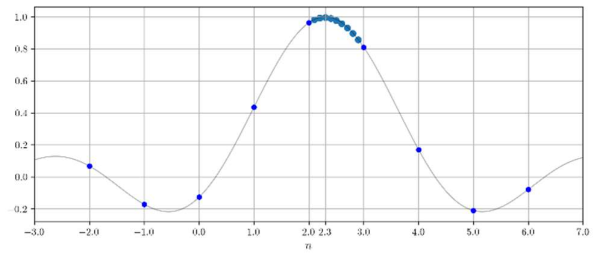

Fig. 5. Interpolated $f(n)$ at $n = 2.0 + \tau$, where $\tau = 0.1i$, $i = 1, 2, \cdots, 9$.

IV. Conclusion

We have proposed an efficient cubic spline interpolation scheme tailored for wireless communication systems, with a focus on OFDM systems. The conventional cubic spline interpolation cannot be used because the slopes at the boundaries of f(x) are not available in wireless communication systems. To address this problem, the proposed scheme uses two more control points making it possible to apply cubic spline. The proposed scheme is significantly more efficient in terms of computational complexity compared to the conventional interpolation schemes based FFT/IFFT. Furthermore, the proposed scheme allows arbitrary time precision by making it possible to choose any point between two samples for interpolation. The performance evaluation showed that the estimation error of the interpolated sample was less than 0.5% in a simple scenario without noise. However, the proposed scheme uses only 6 adjacent samples to perform interpolation, and thus susceptible to low SNR. Because of this, application of the proposed interpolation scheme can be limited. One area of application of the proposed scheme is high precision time of arrival estimation as illustrated in the example. By the cross-correlator, the noise signal is smoothed over a period of time (the span of the cross-correlator) and the output of the cross-correlation has significantly higher SNR compared to the signal (pulse compression gain). We would like to further explore the possibility of applying this scheme to semantic communication, which is more focused on conveying semantic information not bit-level information.

ACKNOWLEDGMENT

This research is supported by an Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00746, Development of Tbps wireless communication technology) and an Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korea government (MSIT) (No. 23YH1510, AI-Based Semantic Communications)

REFERENCES

[1] John G. Proakis, Dimitris G. Manolakis, Digital Signal Processing: Principles, Algorithms and Applications, 3rd Edition, Prentice Hall, October 5, 1995.

[2] Gary D. Knott, Interpolating Cubic Splines, Birkhäuser Boston, December 28, 1999.