An Efficient Signal Resampling Scheme for Wideband-to-Narrowband Doppler Conversion in Underwater Acoustic Communications

Çağatay Candan

Abstract

An efficient resampling scheme particularly suitable for the implementation of sampling rate changes close to the unity rate, which are typically required at the frontend of underwater communication receivers, is given. The suggested scheme is based on the fractional delay filtering operation via the Lagrange interpolation and can be efficient implemented with the Newton series. The scheme yields a very good performance at a significantly reduced complexity in comparison to the conventional upsampling/downsampling based schemes. A ready-to-use Matlab implementation of the suggested method is also provided.

Index Terms

Signal Time-Scale Change, Sampling Rate Change, Fractional Delay, Farrow Structure.

I. Introduction

Orthogonal frequency-division multiplexing [1] and orthogonal signal-division multiplexing [2], [3] are closely related methods utilized to improve the reliability and communication rate in underwater acoustic channels. Different from the terrestrial radio communications, the Doppler effect in the underwater channel can not considered as a narrowband phenomena in general and has to be carefully compensated, [4], [5]. As noted in [5, p.52], the time dilation/compression of the received signal, which is the wideband Doppler effect, as low as 1 ± 0.001 can significantly deteriorate the reliability of the communication. Since the time dilation/compression rate lies in the range $(0.99, 1.01)$ for the relative speeds up to 29 knots, the compensation of wideband Doppler effect becomes an important

Author is with the Department of Electrical and Electronics Engineering, Middle East Technical University (METU), Ankara, Turkey. (e-mail: ccandan@metu.edu.tr).

issue for the underwater acoustic communications. In [1], a receiver front-end resampling operation is suggested to partially compensate the wideband Doppler effect, that is to approximately convert the difficult to handle wideband Doppler effect to a narrowband one. For example, the resultant signal after resampling operation is considered to be free of Doppler effect; but contaminated with time-varying phase distortion, in [6], [7], [8], [9]. To further improve the performance, the residual phase distortion can also be estimated along with the channel impulse response. Here, we present a filtering scheme, based on the fractional delay filtering via Lagrange interpolator, for the efficient implementation of the front-end resampling operation required in similar underwater communication systems. Different from conventional upsampling/downsapling schemes, the suggested scheme uses a time-varying filter whose structure is well matched for the implementation of sampling rate changes close to the unity rate.

The goal of resampling is to generate the samples of the scaled signal, $x_\beta[n] = x(\beta t)|_{t=nT}$, by digitally processing the samples of $x(t)$, i.e. $x[n] = x(t)|_{t=nT}$. The problem of generating $x_\beta[n]$ from $x[n]$ can also be interpreted as the change of sampling period from T to βT. Our main interest is the case of $\beta \approx 1$, i.e. the rates close to unity, say $\beta = 24/25$. The conventional techniques for the implementation of such changes results in a significant computational burden. For example, the rate change of $24/25$ requires upsampling by 25 units followed by downsampling by 24 units which can be prohibitively costly to implement. The main goal of this study is to describe a very low cost filtering scheme specifically to realize the rate changes around unity.

The suggested approach uses the efficient realization of Lagrange interpolation described in [10]. The principle of the suggested method has been first given in the context of audio format conversion from CD (sampled at 44.1 kHz) to DAT (sampled at 48 kHz), [11]. Some earlier studies also study the application of fractional delay filters for the sampling rate conversion, [12], [13], [14]. The main contribution of the present work is the adaptation of the Newton series based Lagrange interpolation scheme given in [10] to the sampling rate conversion application and its performance examination. A ready-to-use MATLAB implementation of the suggested method and scripts reproducing the results given in this paper is available in [15].

II. Preliminaries

The signal $x(t)$ and its scaled version $x_\beta(t) = x(\beta t)$ are assumed to be sampled with the sampling period T, i.e. $x[n] = x(t)|_{t=nT}$ and $x_\beta[n] = x(\beta t)|_{t=nT}$. Note that the signal $x_\beta[n]$ can also be considered as the samples of $x(t)$ with the sampling period βT, that is $x_\beta[n] = x(\beta t)|_{t=nT} = x(t)|_{t=n(\beta T)}$. Hence, the process of generating $x_\beta[n]$ from $x[n]$ can be interpreted as either the change of sampling period from T to βT or the change of signal time-scale from 1 to β.

For some applications, the desired sampling rate change (or scale factor) can be close to unity, $\beta \approx 1$. The working principle of the suggested scheme for such applications can be described by introducing α, which is assumed to be $|\alpha| \ll 1$, through the relation $\beta = 1 - \alpha$. Then, the scaling relation becomes as follows:

$$x_\beta(t) = x(\beta t) = x(t - \alpha t). \tag{1}$$

It is possible to interpret αt in (1) as the amount of delay (or advance, if $\alpha < 0$) of signal $x(t)$ at time t. The mentioned delay is an increasing function of time. Provided that the increasing delay can be synthesized for all t, the scaled signal can be properly generated. In terms of the sampled data, the delay of αt seconds corresponds to $\alpha t/T$ samples.

Figure 1 illustrates the approach for $\alpha = 1/3$. The presented value for α corresponds to the scale change of $\beta = 1 - \alpha = 2/3$ or the sampling rate change of $1/\beta = 3/2$. From Figure 1, it can be noted that the n'th output sample ($y[n]$) has a time delay of $n \times 1/3$ samples from the n'th input sample ($x[n]$).

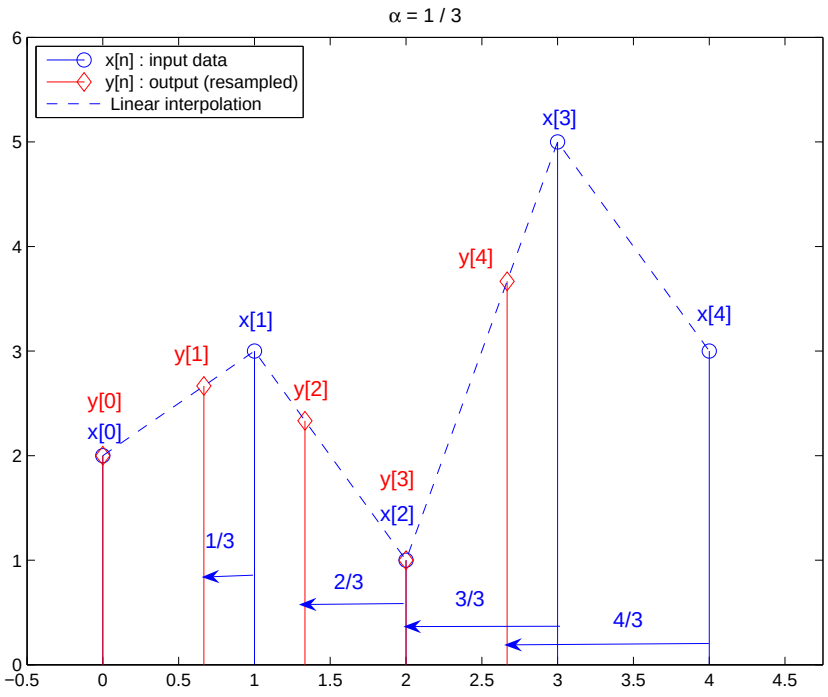

Fig. 1. Resampling for the sampling rate change of $1/\beta = 1/(1-\alpha)$ or the signal scaling of $\beta = 1 - \alpha$ for $\alpha = 1/3$.

The dotted line in Figure 1 is the result of linear interpolation. By evaluating the linear interpolation result at an increasing delay of $D[n] = \alpha n$ samples from the n'th input sample, the scaled output can be generated. Even though, this interpretation (output being the fractionally delayed version of input) is valid; we prefer to interpret the output as the resampled version of input. Our goal is to generalize the resampling operation from the linear interpolator scheme shown in Figure 1 to an arbitrary order

polynomial interpolation and present an efficient mechanism for its implementation. In this section, a detailed description of the easiest case, the linear interpolation case, is given to assist the discussion of the general case.

It can be noted from Figure 1 that each output sample is generated by processing two closest samples around the output sample. For example, $y[1]$ is generated from $x[0]$ and $x[1]$; upon the arrival of $x[2]$, the next output sample $y[2]$ is generated from $x[1]$ and $x[2]$ and so on. The system illustrated is to increase the sampling rate change by $3/2$; hence, the system should produce more than one output sample per input sample on the average. The increase in the rate can be noted by studying the output sample $y[4]$. It should be noted that the required delay for $y[4]$ is $4/3$ samples, exceeding the unit delay for the first time. The sample $y[4]$ is generated from the input samples of $x[2]$ and $x[3]$ which are two neighboring input samples. It should also be noted that the same set of input samples ($x[2]$ and $x[3]$) is also used for the generation of $y[3]$. Hence, the set of samples $x[2]$ and $x[3]$ is utilized twice for the generation of two output samples and this leads to the increase in the sampling rate. It should also be noted that the case described for $y[4]$ occurs repeatedly whenever the output sample has a delay exceeding a full integer value.

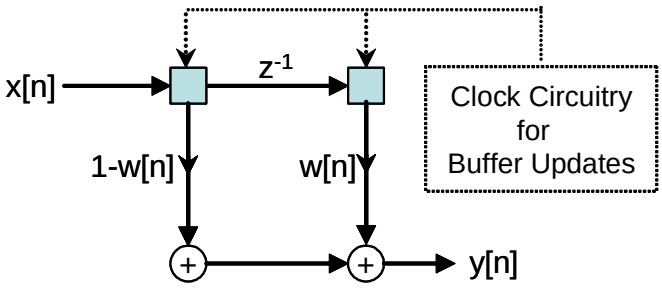

Fig. 2. Filtering scheme for the implementation of the resampling process shown in Figure 1. The multiplier values are given in Table I.

Figure 2 shows a possible implementation for the realization of the fractional delays depicted in Figure 1. The delay of $D[n] = \alpha n$ ($\alpha = 1/3$) and the multiplier $w[n]$, appearing in the filtering diagram in Figure 2, are listed in Table I. The table is presented to emphasize the time-varying nature of the scheme. It should be noted that when $D[n]$ exceeds a full integer delay; the samples in the buffers, which are shown with dark squares, are not shifted; hence the same buffer content is utilized twice to produce two output samples. This occurs at $n = 4$ and $n = 7$, where $D[n]$ exceeds a full integer delay, as noted in Table I.

In this section, the case of upsampling ($\alpha > 0$) is presented through an example having $\alpha = 1/3$. For $\alpha < 0$, the system becomes a downsampling system. For the downsampling case, in analogy with the upsampling case, some buffer contents are discarded and are not used to produce an output. We

TABLE I

IMPLEMENTATION OF THE SCHEME SHOWN IN FIGURE 1

n	$D[n]$	$w[n]$	buffer contents
0	0	0	x[0], x[-1]=0
1	1/3	1/3	x[1], x[0] (updated)
2	2/3	2/3	x[2], x[1] (updated)
3	3/3	3/3	x[3], x[2] (updated)
4	4/3	1/3	x[3], x[2] (not updated)
5	5/3	2/3	x[4], x[3] (updated)
6	6/3	3/3	x[4], x[3] (updated)
7	7/3	1/3	x[4], x[3] (not updated)
$\vdots$	$\vdots$	$\vdots$	$\vdots$

do not present further details of this case; since a detailed MATLAB implementation of the proposed scheme encompassing both upsampling and downsampling cases, is provided at [15].

III. PROPOSED SCHEME

The proposed scheme generalizes the scheme described in the earlier section to higher order interpolators. Figure 3 illustrates the suggested scheme for $\alpha = 1/3$. As before, the first two samples are generated through the linear interpolator (shown with dotted lines) having the delays 0 and $1/3$, respectively. As shown in Figure 3, the next output sample, $y[2]$, is generated by combining the closest *two* input samples on each side of $y[2]$, instead of using one sample on each side as in the linear interpolator. The output sample is generated by finding the unique 3rd order polynomial passing through 4 input points in the neighborhood of $y[2]$. The method of interpolation by fitting an Nth degree polynomial to $N + 1$ data inputs is called as the Lagrange interpolation, [10]. The Lagrange interpolation is known to provide maximally flat frequency response around DC frequency and frequently utilized when the input is oversampled.

The Lagrange interpolation can be implemented in various ways. One of most efficient implementation is through the Newton series expansion, [10]. The Newton series implementation has the complexity of $O(N)$ multiplications and additions per output sample for the Nth degree interpolation.

The Newton series is the discrete time equivalent of Taylor series. The backward difference operator Δ, $\Delta\{f[n]\} = f[n] - f[n-1]$, in analogy with the derivative operator, and the Mth degree factorial polynomial

$$x^{[M]} = \begin{cases} 1, & M = 0 \\ x(x+1)\dots(x+M-1), & M \geq 1 \end{cases}, \tag{2}$$

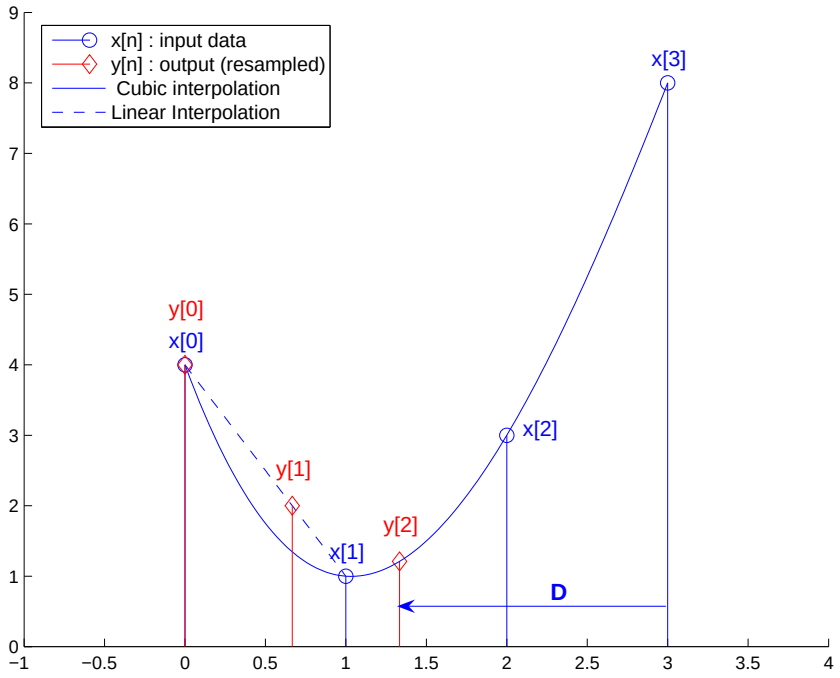

Fig. 3. The resampling process for the proposed scheme

in analogy with x^M, are defined to introduce the Newton series. The presented definitions lead to the fact that $\Delta\{x^{[M]}\} = Mx^{[M-1]}$, in analogy with $\frac{d}{dx}x^M = Mx^{M-1}$. Given these, Newton series can be defined as follows, [10]:

$$\widetilde{x}(k-D) \;=\; \sum_{i=0}^{N-1} \Delta^i\{x[k]\}\frac{(-D)^{[i]}}{i!} \tag{3}$$

It can be noted from (3) that $\widetilde{x}(k-D) = x[k-D]$ for $D = \{0, 1, \ldots, N-1\}$. This fact can be most easily verified by applying Δ operator on both sides of (3) and evaluating the result at $D = 0$, as described in [10]. It should be noted that the polynomial $\widetilde{x}(k-D)$ is the Lagrange interpolation polynomial; since it passes through $N+1$ consecutive data samples of $\{x[k-N], \ldots, x[k]\}$.

The main advantage of Newton series formulation is the ease of its digital implementation. Figure 4 shows the proposed Newton Series based implementation exactly following the analytical relation given in (3). The implementation shown has N multipliers and N first order differencing blocks, shown with $1 - z^{-1}$. The value of the multiplier at the output of ith differencing block at the time instant of n is denoted by $w_i[n]$. The time-varying nature of the filtering scheme is evident from the implementation.

The weights $w_i[n]$ can be specified as follows. As before, $D[n]$ represents the value of the desired delay at the sample n. The proposed N point ($N-1$'th order) Lagrange interpolator scheme uses the nearest $N/2$ input samples on each side of the output sample for the interpolation. The delay at

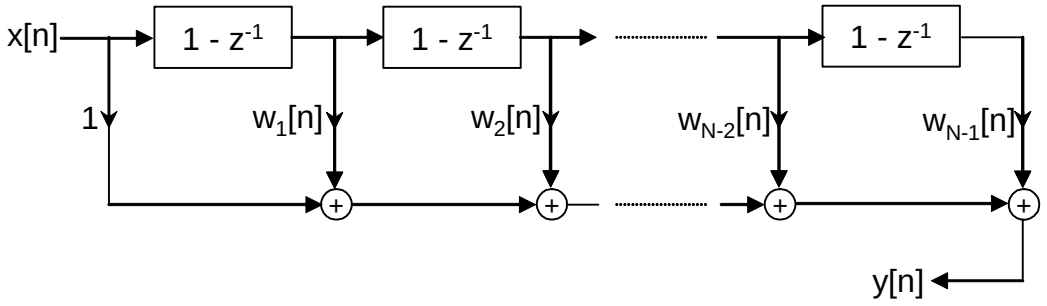

Fig. 4. Proposed structure for the sampling rate change / signal scaling (The clock circuitry controlling the buffer updates is not shown).

the time instant n can be written as follows:

$$D[n] = \frac{N-2}{2} + \alpha n, \quad N: \text{ even} \tag{4}$$

The term $\frac{N-2}{2}$ in (4) corresponds to the bulk delay due to the usage of $N/2$ input points on the right side of the output sample. (The bulk delay is required for the causality of the implementation.) The term of αn in (4) corresponds to the time varying delay. When $N = 2$, the equation (4) reduces to the definition given in the previous section for the linear interpolator.

Following the formulation of (3), the filter coefficients $w_i[n]$ $i = \{0, \ldots, N-1\}$ can be explicitly written as follows:

$$w_i[n] = \frac{1}{i!}\left(-\left(\frac{N-2}{2} + \text{rem}(\alpha n, 1)\right)\right)^{[i]} \tag{5}$$

The term of $(N-2)/2 + \text{rem}(\alpha n, 1)$ corresponds to the effective delay. The function $\text{rem}(\cdot, 1)$ represents the non-integer (fractional) part of the argument. More specifically, $\text{rem}(\cdot, A)$ corresponds to the remainder operation when the argument is divided by $A+\epsilon$ for an arbitrary small positive ϵ, i.e. the range of $\text{rem}(\cdot, A)$ is $[0, A]$. It should be noted that both $\text{rem}(\alpha n, 1)$ and $w_i[n]$ can be recursively calculated following [14, Fig. 3] and [10, Eq. 4], respectively.

Similar to the linear interpolation scheme discussed before, when delay $D[n]$ exceeds a full integer value; the buffer contents are not updated. By choosing not to update the buffer contents for such instances, the additional unit sample increase in the delay is integrated into the system. The remaining part of the delay after the integration of its integral part becomes $\text{rem}(\alpha n, 1)$.

The described operation can be most easily understood from the listing given in Table I. At the sample of $n = 4$, the desired delay is $4/3$ samples. To implement this delay, the buffer contents for $n = 3$ is used to the produce the 4th output sample and the required fractional delay becomes $4/3 - 1 = 1/3$ (which is identical to the multiplier $w[n]$) for the linear interpolation scheme (also see Figure 1).

The present discussion is focused on the case of upsampling ($\alpha > 0$) with even number of input points (N : even) for the interpolation. Both restrictions can be easily lifted as follows:

Case of Odd N: The description given above assumes the number of points utilized in the Lagrange interpolator is even. The approach and the filtering scheme given in Figure 4 can be used as it is for the odd values of N by adopting $D[n] = \frac{N-3}{2} + \alpha n$ instead of (4). This choice corresponds to usage of $\frac{N-1}{2}$ samples on the right side of output sample and $\frac{N+1}{2}$ samples on the left side of the output sample. (There can be other alternatives.) The bulk delay in (5) for the calculation of multiplication coefficients becomes $\frac{N-3}{2}$. This applies to both upsampling and downsampling cases.

Case of Downsampling ($\alpha < 0$): The downsampling case follows quite easily from the given discussion. One important change is the change in the argument of $\text{rem}(\cdot, 1)$ appearing in (5) from αn to $(1 + \alpha)n$. Specific to the downsampling case, if $D[n]$ crosses an integer boundary more than once, the buffer contents updated the same number of times in an update cycle. For further details, readers can examine the MATLAB code given for both downsampling and upsampling in [15].

IV. Numerical Results

In all three sets of experiments presented in this section, the input is taken as $x[n] = \cos(\omega n)$. The output generated by the scheme is compared with the desired output of $x_\beta[n] = \cos(\beta\omega n)$ in the root mean square (RMS) sense for various values of signal frequency (ω) and the scaling factor (β).

Figure 5 shows the output produced by the scheme for $\omega = \pi/4$ and $\beta = 4/5$; stated differently, the input $x[n] = \cos(\pi/4n)$ is resampled such that the output is $x_\beta[n] = \cos(\pi/5n)$. In this comparison, the period of the input changes from 8 samples to 10 samples. Figure 5 shows the scheme output for linear ($N = 2$) and cubic interpolation ($N = 4$). It can be noted that the case of $N = 4$ is almost identical to the true output in this experiment.

Figure 6 shows the RMS error when the input $\cos(\pi/4n)$ is scaled by $\beta \in [0.8, 1.2]$ by different order Lagrange interpolators. The examined range for β corresponds to the sampling rate change from $5/6$ (downsampling) units to 1.25 (upsampling) units. The range of β given on the x-axis of Figure 6 is scanned with the step-size of 0.01 units. The case of $\beta = 1$ corresponds to the case of no sampling rate change for which RMS error is zero. In addition to the RMS value of the resampling error, the RMS value of $\cos(\pi/4\beta n)$ (desired output), which is $1/\sqrt{2}$, is also indicated in Figure 6 for comparison purposes.

Figure 7 shows the error for $\beta = 0.99$ when the frequency of the input $\cos(\omega n)$ is varied. As $\omega \to \pi$, the polynomial interpolation ceases to be useful; since rapid oscillations can not be faithfully approximated by a small degree polynomial. It can be concluded from Figure 7 that the presented scheme works well for $|\omega| \leq \pi/2$. Hence, it is possible to use the suggested method for a wide range of β, not only for $\beta \approx 1$, provided that the input is at least two times oversampled. Hence,

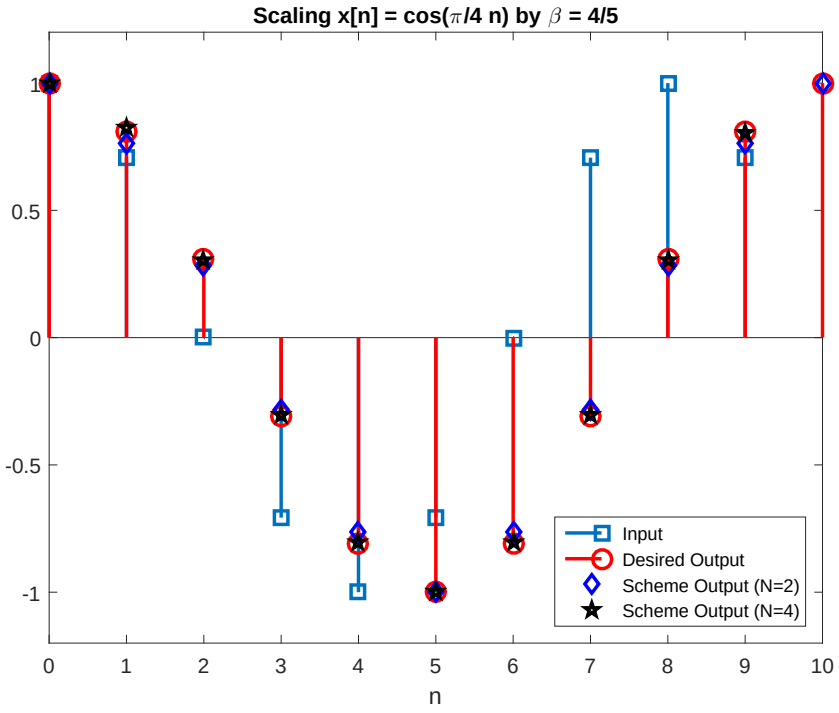

Fig. 5. Resampling $\cos(\pi/4n)$ by the rate of $\beta = 4/5$.

if the input provided is not oversampled by 2, the designer implementing a rate of change around unity may choose to upsample the input by 2 first via the conventional methods and then apply the suggested method.

V. Conclusions

An efficient signal resampling scheme based on the Lagrange interpolation via Newton's series is described. The target application venue of the scheme is the sampling rate change operation at the frontend of the underwater acoustic communication receivers where the desired rate of the change is typically very close to unity. For such rates, the suggested scheme yields a very good performance at a significantly reduced complexity in comparison to the conventional upsampling/downsampling based schemes. The overall complexity of the suggested scheme is N multiplications per output sample for the Nth order interpolation and the order of interpolation can be as low as 5 in many cases. A ready-to-use MATLAB implementation of the scheme is available in [15].

References

[1] B. Li, S. Zhou, M. Stojanovic, L. Freitag, and P. Willett, "Multicarrier communication over underwater acoustic channels with nonuniform Doppler shifts," *IEEE J. Ocean. Eng.*, vol. 33, no. 2, pp. 198–209, April 2008.

[2] T. Ebihara and K. Mizutani, "Underwater acoustic communication with an orthogonal signal division multiplexing scheme in doubly spread channels," *IEEE J. Ocean. Eng.*, vol. 39, no. 1, pp. 47–58, Jan 2014.

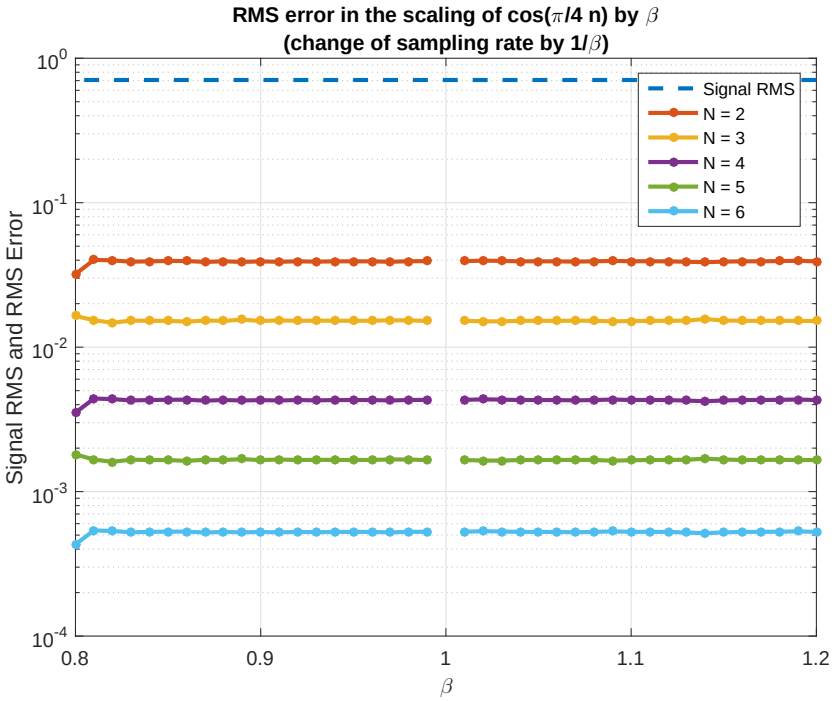

Fig. 6. RMS error for the resampling of $\cos(\pi/4n)$ by β.

[3] T. Ebihara and G. Leus, "Doppler-resilient orthogonal signal-division multiplexing for underwater acoustic communication," *IEEE J. Ocean. Eng.*, vol. 41, no. 2, pp. 408–427, April 2016.

[4] M. Stojanovic, J. A. Catipovic, and J. G. Proakis, "Phase-coherent digital communications for underwater acoustic channels," *IEEE J. Ocean. Eng.*, vol. 19, no. 1, pp. 100–111, Jan 1994.

[5] R. Istepanian and M. Stojanovic, *Underwater Acoustic Digital Signal Processing and Communication Systems*. USA: Springer Press, 2002.

[6] S. F. Mason, C. R. Berger, S. Zhou, and P. Willett, "Detection, synchronization, and Doppler scale estimation with multicarrier waveforms in underwater acoustic communication," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 9, pp. 1638–1649, December 2008.

[7] Z. Wang, S. Zhou, G. B. Giannakis, C. R. Berger, and J. Huang, "Frequency-domain oversampling for zero-padded OFDM in underwater acoustic communications," *IEEE J. Ocean. Eng.*, vol. 37, no. 1, pp. 14–24, Jan 2012.

[8] K. Tu, T. M. Duman, M. Stojanovic, and J. G. Proakis, "Multiple-resampling receiver design for ofdm over doppler-distorted underwater acoustic channels," *IEEE J. Ocean. Eng.*, vol. 38, no. 2, pp. 333–346, April 2013.

[9] J. Han, S. P. Chepuri, Q. Zhang, and G. Leus, "Iterative per-vector equalization for orthogonal signal-division multiplexing over time-varying underwater acoustic channels," *IEEE J. Ocean. Eng.*, vol. 44, no. 1, pp. 240–255, Jan 2019.

[10] C. Candan, "An Efficient Filtering Structure for Lagrange Interpolation," *IEEE Signal Processing Lett.*, vol. 14, no. 1, pp. 17–19, Jan. 2007.

[11] K. Rajamani, Y.-S. Lai, and C. Furrow, "An efficient algorithm for sample rate conversion from CD to DAT," *IEEE Signal Processing Lett.*, vol. 7, no. 10, pp. 288–290, Oct. 2000.

[12] A. Tarczynski, W. Kozinski, and G. Cain, "Sampling rate conversion using fractional-sample delay," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Apr. 1994, pp. 285–288.

[13] F. Harris, "Performance and design of Farrow filter used for arbitrary resampling," in *13th Int. Conf. on Digital Signal*

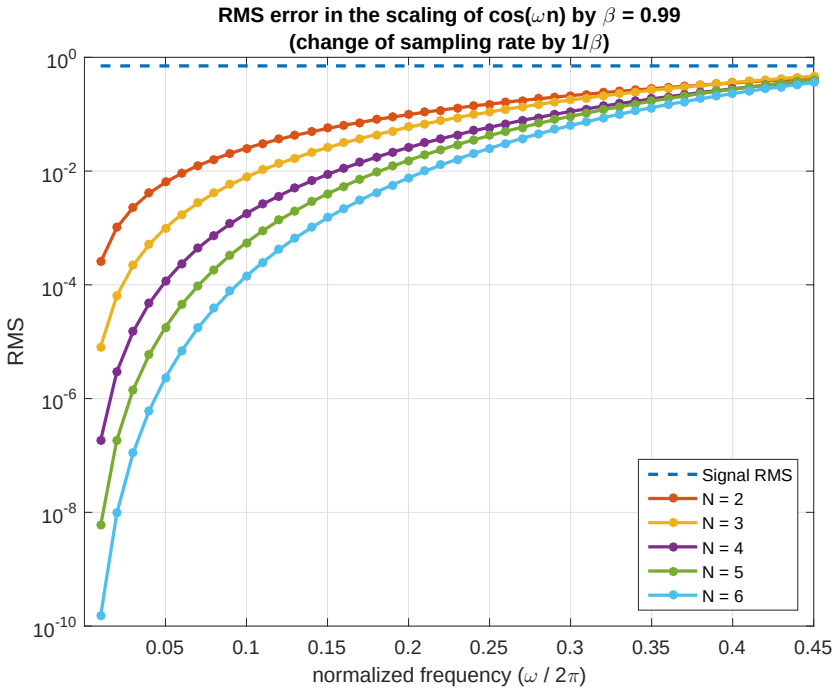

Fig. 7. RMS error for the resampling of $\cos(\omega n)$ by $\beta = 0.99$.

Processing Proceedings,, July 1997, pp. 595–599.

[14] M. Blok, "Fractional Delay Filter Design for Sample Rate Conversion," in *Proc. Federated Conference on Computer Science and Information Systems*, Sept. 2012, pp. 701–706.

[15] C. Candan. (2019) An Efficient Signal Resampling Scheme for Wideband-to-Narrowband Doppler Conversion in Underwater Acoustic Communications, MATLAB Code. [Online]. Available: http://users.metu.edu.tr/ccandan/a/resampling.zip