

PERFORMANCE AND DESIGN OF FARROW FILTER USED FOR ARBITRARY RESAMPLING

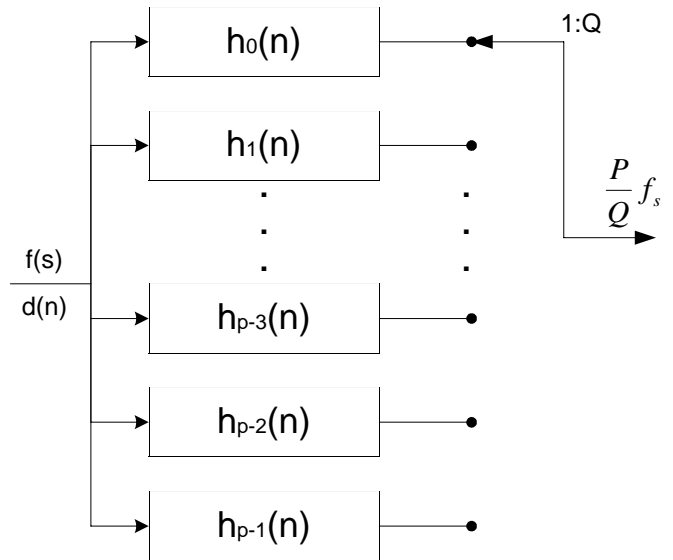
The Farrow filter [1] is a multirate filter structure which offers the option of continuously adjustable resample ratio. This paper presents a derivation of the method proposed by Farrow, and demonstrates the performance and complexity of resampling filters using his technique. The paper also develops some important system options made available to the designer as spin-offs of the derivation.

ABSTRACT

INTRODUCTION:

Multirate filters are designed and implemented as polyphase P-path filters with each path providing a delay of an integer multiple of $1/P^{\text{th}}$ of the output sample rate for down-sampling or $1/P^{\text{th}}$ of the input sample rate for up-sampling. Since each path corresponds to different phase slopes, their outputs are normally referred to by Their phase index. For an upsampling filter, we say $d(nP+r)$ or $d_r(n)$ is the output from the r^{th} phase in response to the input at time n . Traditionally the coefficients for the bank of P-path filters are precomputed and stored and then accessed by a Q-to-1 sequencing rule to implement a P/Q resampling operation. This structure is shown in figure 1 and while this structure presents the functional oper-

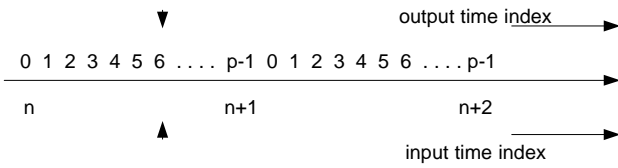
ation, the implementation is actually performed as memory management of the P-coefficient sets.



■ Figure 1. A polyphase Up-Sampling Filter

For filters designed to operate over a large ratio of P/Q, the number of stages P is made large so that time jitter associated with selecting a phase path nearest to the desired output time position is made acceptably small. This time jitter is suggested in figure 2 where the desired output sample is located between two time points available from a P-stage polyphase filter bank. The output is selected from the stage nearest the desired time loca-

tion.



■ Figure 2. *Input and Output Sample Locations For P-Stage Resampling Filter*

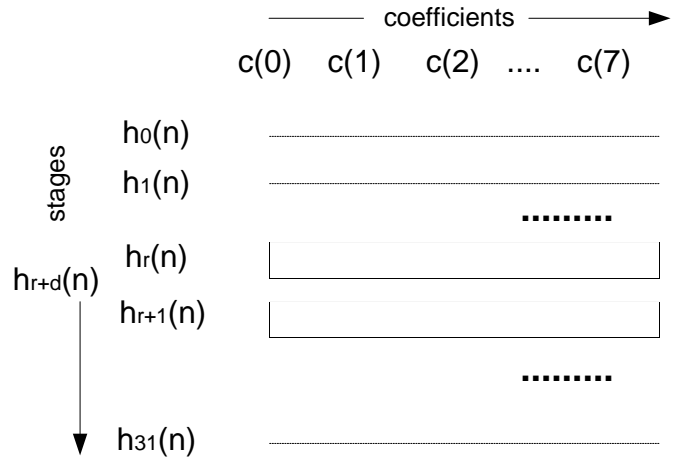
The newest neighbor rule is a course interpolator and results in spectral artifacts bounded by $1/(2P)$ [2]. Linear interpolation between two adjacent outputs reduces the level of spectral artifacts to $1/(2P)^2$. For a specific example, to hold spectral artifacts 50 dB below peak signal spectra, we would require 158 stages using the nearest neighbor rule and would only require 9-stages using the linear interpolator. In both cases the in band distortion due to the sinc/x or $(\text{sinc}/x)^2$ would have to be controlled by pre or post equalization.

COMPACT REPRESENTATION OF FILTER:

One option available to implement an arbitrary resampling filter is to have precomputed the weights of the polyphase filter stages for each resampling ratio and then download them to the processor on startup. Another option is to form a large array representing a highly oversampled filter (large oversample ratio) with the desired low-pass spectral response and then use the nearest neighbor or linear interpolation rule to form samples of the filter at the desired points required for the specific P stage polyphase filter bank. If the filter requires L symbols per stage to compute an output, the oversampled prototype is of length $9 \cdot L$. As a useful reference, for a square-root Nyquist Filter with 0.25 dB implementation loss and with rolloff $\alpha = 0.4$ and 50dB sidelobes, L is 22, and with rolloff $\alpha=0.2$ with the same 50 dB sidelobes, L is 44. This meat's a prototype table of length 198 to 396 taps would have to be available

for linear interpolation to an arbitrary set of filter weights. This table does not include the precompensation for a DAC's sinc/x distortion!

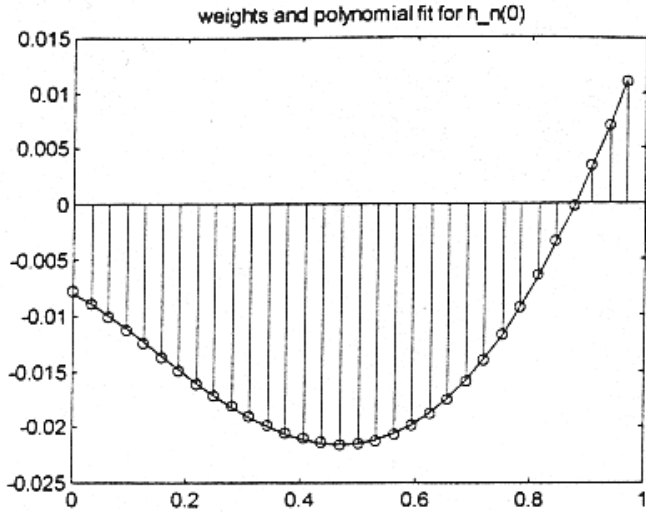
A third option is to approximate the filter weights In a segment of its impulse response by a low order polynomial and then use the polynomial to compute the filter weights at their desired positions. Consider a 32 stage polyphase filter with S-taps per stage. This filter is defined by its 256 taps which can be visualized as having been loaded into the poyphase filter set as a two dimensional array by loading successive columns but processing data by rows. The c-th column of the two dimensional coefficient set contains the c-th coefficient of each polyphase filter. We note that it takes 32 entries to define each column. The filter weights corresponding to the r-th stage are located r-steps down each column or the fraction (r/P) between input samples. The filter weights corresponding to the $(r+1)$ -th stage are of course located $r+1$ steps down the each column. If we want a filter between the r-th and $(r+1)$ -th position, we can interpolated between the stored values in each column. This process can be visualized with the aid of figure 3.



■ Figure 3. *Two Dimensional Mapping Of Polyphase Filter Coefficient Set.*

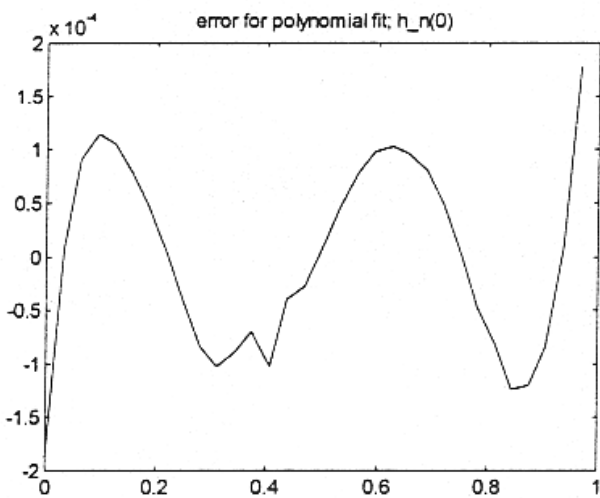
Rather than define each column by the 32 samples of the polyphase partition we can pass a low degree polynomial through the coefficients and use the polynomial as an equivalent description. Figure 4 presents the 32 samples of column zero

of a prototype filter along with a fourth order polynomial that has approximated these samples. Figure 5 presents the error between the samples and the fourth order polynomial expansion. The error is seen to be on the order of 2×10^{-4} or approx-



■ Figure 4. 32 Samples of Column Zero Of Polyphase Filter And Fourth Order Polynomial Approximation

imately the error of a 12-bit quantizer. Applying the rule of thumb that the sidelobe structure of a filter follows a 5-dB per bit requirement this 12-bit error is sufficient to support the 60 dB sidelobes of the prototype filter we are currently examining.

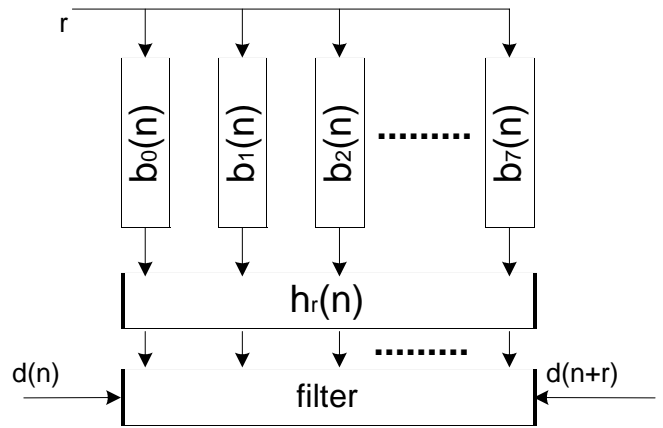


■ Figure 5. Error Between Actual Samples And Fourth Order Polynomial Approximation

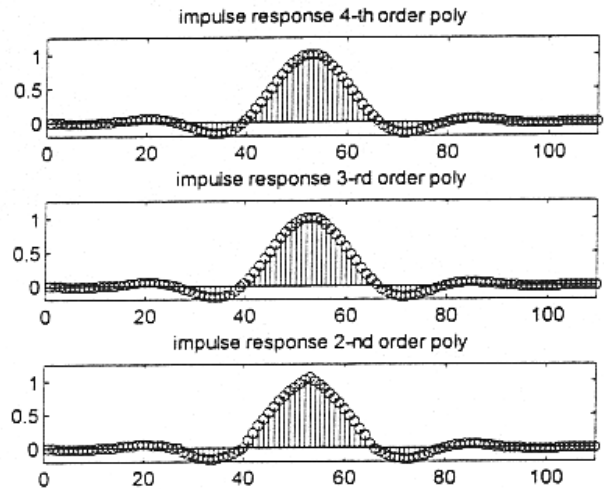
In Like fashion each column of the polyphase par-

tion is cast into a fourth order polynomial so that the entire 32 stages of the prototype are now represented by 40 coefficients. In fact, filters with an arbitrary number of stages are now represented by the polynomial description. The two dimensional filter set can now be replaced with an algorithm structure of the form shown in figure 6. Here the columns are the polynomials representing the coefficients in each of the coefficient set. The r-th filter set is computed by the polynomials and presented to the processor to form the inner product of the filter.

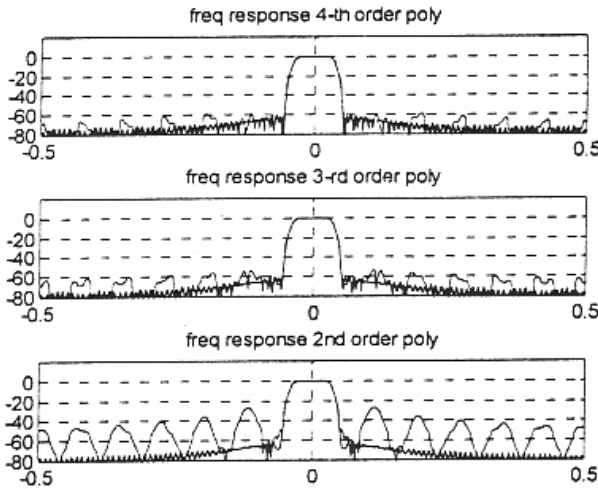
Figure 7 presents the impulse response formed by the polynomial segments for polynomials of degree 4, 3, and 2, while figure 8 presents the fre-



■ Figure 6. Polynomial Representation of Resampling Filter: Coefficients Computed For Each Sample Time



■ Figure 7. Fourth, Third and Second Degree Polynomial Approximations to Prototype Filter



■ Figure 8. Frequency Response of Polynomial Approximations to Prototype Filter

frequency responses of the corresponding filter approximations. As we can see, the spectral artifacts generated by the fourth order polynomial approximation were more than 60 dB below the filters' passband level. Similarly the third degree polynomial created 55 dB artifacts while the second order filter generated unacceptably high 28 dB artifacts.

FARROW FILTER

The filter structure of a polyphase partition is presented in eq(1) where the parameter Δ replaces the index r specifying the polyphase stage to emphasize that the displacement Δ is now continuous.

$$d(n + \Delta) = \sum_{k=0}^7 c_{\Delta}(k) d(n - k) \quad (1)$$

Equation 1 is recast in the polynomial form in eq (2).

$$d(n + \Delta) = \sum_{k=0}^7 \left[\sum_{l=0}^4 b(k, l) \Delta^l \right] d(n - k) \quad (2)$$

Exchanging the order of the summations we

obtain eq (3) which we write more compactly as eq (4).

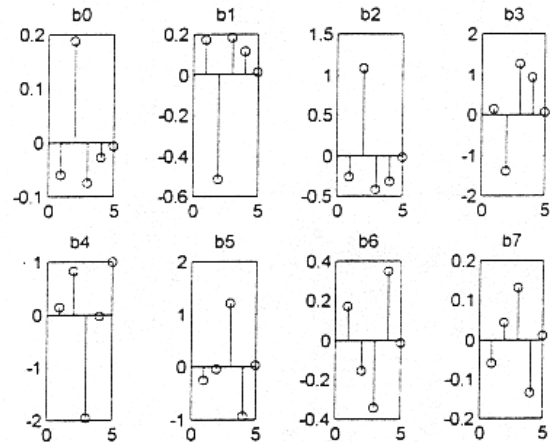
$$d(n + \Delta) = \sum_{l=0}^4 \left[\sum_{k=0}^7 b(k, l) d(n - k) \right] \Delta^l \quad (3)$$

$$d(n + \Delta) = \sum_{l=0}^4 h_l(d) \Delta^l \quad (4)$$

It is a convenient perspective to think of the polynomial expansion of the coefficient sets at their Taylor Series (actually Tchebyshev) expansion. From this perspective we conclude that the term $h_l(d)$, defined in eq (5), the result of processing the data with the Taylor Series expansion of the coefficient set becomes the data dependent Taylor Series expansion of the data.

$$h_l(d) = \sum_{k=0}^7 b(k, l) d(n - k) \quad (5)$$

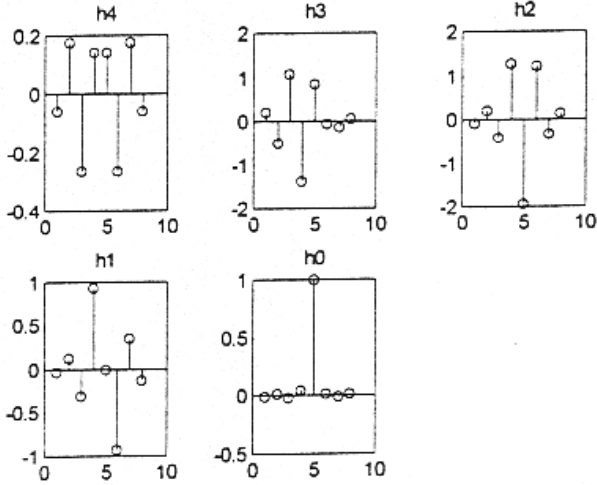
This is more obvious if we examine the two dimensional list of polynomial coefficients $b(k, l)$ where k is the index defining the tap (or column in fig 6.) and l is the index within the k -th column. The Taylor series coefficients for each column represented in figure 9.



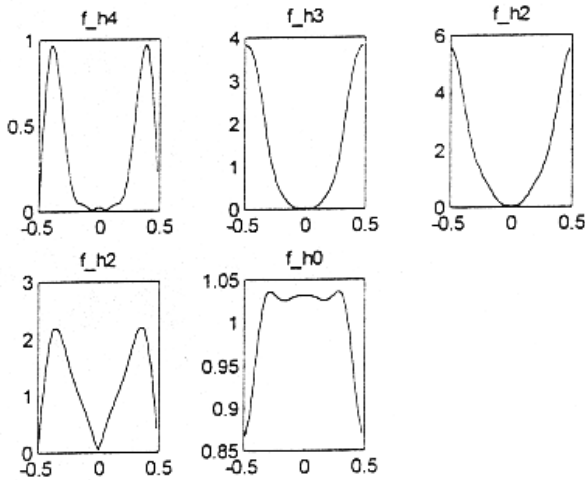
■ Figure 9. Taylor Series Coefficients For Each Column Of Polyphase Partition

In figure 10, these coefficients are reordered by rows to show how they are used in eq (5). A seasoned DSP practitioner may recognize three of

these coefficient sets as an all-pass, a first, and a second order differentiator. Figure 11. presents the magnitude of the frequency response of these filters, Note the constant gain, the linear gain, and the quadratic gain of the first three filters.



■ Figure 10. Row Coefficients Set of Series Expansion



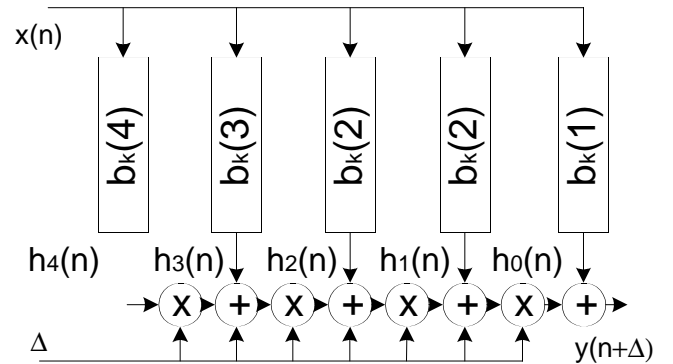
■ Figure 11. Frequency Response of Row Coefficients

The Coefficients shown in Figures 9 and IC are listed in table 1. Here the rows are the fourth degree polynomial (or Taylor Series) expansion of the coefficient sets and the columns are the eight tap differentiating filters whose spectra are shown in figure 11.

X^4	X^3	X^2	X^1	X^0
-----	-----	-----	-----	-----
-0.0596	0.1865	-0.0744	-0.0291	-0.0079
0.1732	-0.5170	0.1845	0.1171	0.0155
-0.2643	1.0740	-0.4190	-0.3206	-0.0266
0.1408	-1.3808	1.2717	0.9230	0.0426
0.1408	0.8350	-1.9481	-0.0116	0.9990
-0.2643	-0.0497	1.2137	-0.9299	0.0131
0.1732	-0.1540	-0.3431	0.3514	-0.0155
-0.0596	0.0445	0.1320	-0.1351	0.0112

TABLE 1: Taylor Series Coefficients

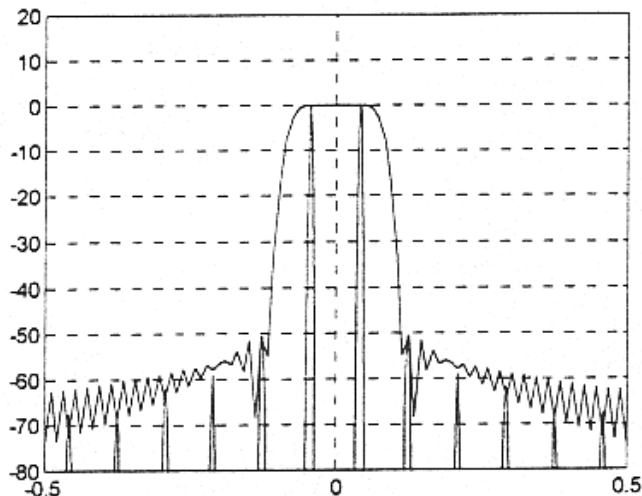
The structure of the Farrow filter is shown in figure 12. Here the data is delivered to each order differentiating filter, the coefficients $b_k(l)$ [or $b(k,l)$] of eq (5). The outputs of these five filters are the Taylor series expansion of the data which evaluates the output at offset Δ , by Horner's rule as shown in eq (6).



■ Figure 12. Farrow Filter: Forms Input Data Taylor Series For Arbitrary Resampling

$$\begin{aligned}
 y(n + \Delta) &= h_0 + h_1\Delta + h_2\Delta^2 + h_3\Delta^3 + h_4\Delta^4 = \\
 &= h_0 + \Delta(h_1 + \Delta(h_2 + \Delta(h_3 + \Delta(h_4))))
 \end{aligned} \tag{6}$$

Figure 13 presents the Spectrum of a sinusoid passed through the Farrow Filter to perform a 1-to-6 upsampling operation. The figure also shows the spectrum of the impulse response of the same 1-to-6 filter. Note sidelobes are suppressed to the 50 dB level for which the prototype filter had been designed.



■ Figure 13. *Spectrum of Farrow Filter Output and Filter Impulse Response For 1-To-6 Upsampling*

CONCLUSIONS

We have rederived the Farrow filter which supports continuously variable resampling. The filter accomplishes this in a two step process by efficiently describing partitions of an oversampled polyphase filter into segments corresponding to columns of the underlying two dimensional mapping. The segments are represented by low order polynomials formed by using the polyfit m-file in MATLAB.

We discovered that the approximating polynomial had to be of fourth degree to sustain artifacts more than 60 dB below the design passband. We tested filter sets with an even number (8) and with an odd number (9) of coefficients per stage. Filters with an odd number of coefficients placed the filter's main lobe in a single column of the partition as opposed to splitting it between two columns. We reasoned there would be better slope match at the boundaries for a given order approximation polynomial. There appeared to be no improvement in approximation error when observed via the spectra of the resultant filter.

The compact representation of the filter segments has system implications when coefficients have to be computed on the fly or in response to a pro-

grammed change in sample rate. Filters operating at low input data rates or with large resampling ratios can be implemented efficiently in the Taylor Series of Data form.

The second step in forming the Farrow filter reorders the two dimensional summation which first forms the filter coefficients for a specified delta and then uses the input data to compute the output for that delta. In the reordered form, the input data is used to compute the Taylor Series expansion in the neighborhood of the current input sample and uses that expansion to compute the output at the desired delta.

We note that when the up-sampling ratio is greater than 1-to-5 the Barrow filter will compute the output samples with fewer operations than the standard polyphase form. A significant advantage of the polynomial form of the polyphase filter sets the savings in memory needed to store the stage coefficients.

BIBLIOGRAPHY

1. **C.W. Barrow**, "A Continuously Variable Digital Delay Element", Proc. IEEE Int. Symp. Circuits Syst.,(ICAS-88), Vol.3, pp 2641-2645, Espoo, Finland, June 6-9, 1988.
2. **f.j. harris**, "Forming Arbitrary Length Windows or Filter Sequences From a Fixed Length Reference Table", Eighteenth Annual Asilomar Conference on Circuits, Systems, and Computers, Pacific Grove CA., Nov. 1984.
3. **T. I Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine**, "Splitting The Unit Delay", IEEE Signal Processing Mag. Jan. 1996, Vol. 13, No. 1, pp. 30-60.
4. **F. M. Gardner**, "Interpolation in Digital Modems Part-I: Fundamentals", IEEE Trans. Comm., Vol. 41, No. 3, pp. 502-508, Mar. 1993.
5. **L. Erup, F. M. Gardner, and F. A. Harris**, "Interpolation in Digital Modems Part-II: Fundamentals and Performance", IEEE Trans. Comm., Vol. 41, No. 6, pp. 998 -1008, June. 1993.