Reducing IIR Filter Computational Workload

By Richard (Rick) Lyons [May 2019]

This document describes a straightforward method to significantly reduce the number of necessary multiplies per input sample of traditional IIR lowpass and highpass digital filters.

Reducing IIR Filter Computations Using Dual-Path Allpass Filters

We can improve the computational speed of a lowpass or highpass IIR filter by converting that filter into a dual-path filter consisting of allpass filters as shown in Figure 1.

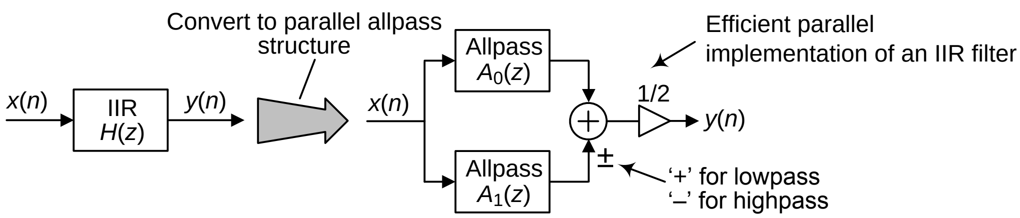


FIGURE 1. IIR filter -to- dual-path allpass filters conversion.

The dual $A_0(z)$ and $A_1(z)$ filter paths in Figure 1 are cascades of simple 1st- and 2nd-order allpass filters. (Allpass filters are a specialized class of IIR filters whose frequency magnitude responses are constant over the full frequency range from zero to the f_s input data sample rate [1].)

Figure 2 shows an example of this standard IIR -to- parallel allpass filter conversion process, where our original IIR filter is a 5th-order lowpass filter.

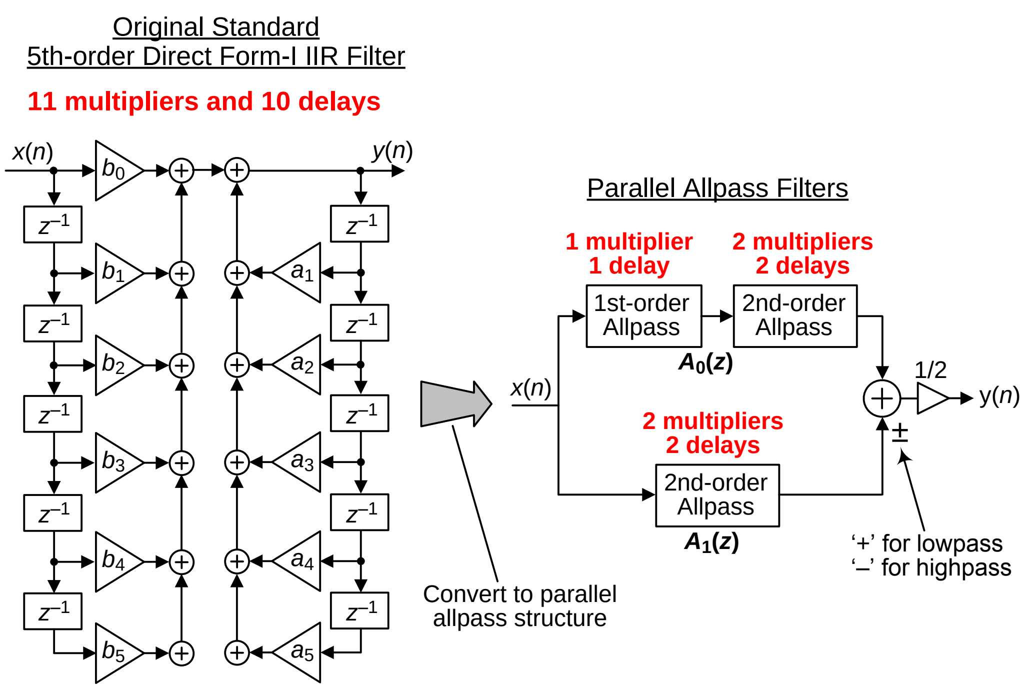
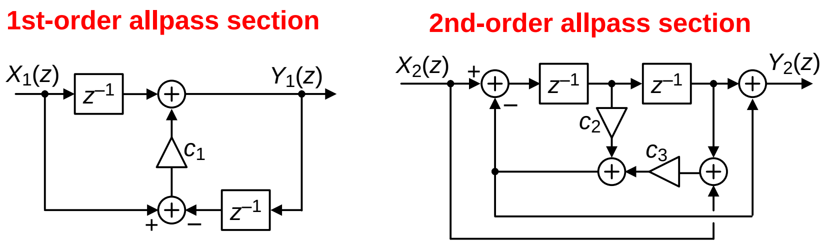


FIGURE 2. Example of "standard" IIR filter -to- parallel-path allpass filter conversion.

The original IIR and the dual-path filters in Figure 2 have identical frequency magnitude responses. **A key point of this document is that the filter on the left side of Figure 2 requires 11 multiplies per input sample whereas the parallel filter on the right side only requires five multiplies per input sample.**

Details of the allpass sections within the 5th-order dual-path filter on the right side of Figure 2 are given in Figure 3. The c_k coefficients are real-valued scalars.



$$H_1(z) = \frac{Y_1(z)}{X_1(z)} = \frac{c_1 + z^{-1}}{1 + c_1 z^{-1}} \qquad H_2(z) = \frac{Y_2(z)}{X_2(z)} = \frac{c_3 + c_2 z^{-1} + z^{-2}}{1 + c_2 z^{-1} + c_3 z^{-2}}$$

FIGURE 3. 1st- and 2nd-order allpass sections and their *z*-domain transfer functions.

We see in Figure 3 that $H_1(z)$ and $H_2(z)$ are indeed allpass filters because their numerator coefficients are in reverse order from their denominator coefficients.

At first glance the notion of implementing a lowpass or highpass IIR filter using allpass filters seems impossible. An explanation of how this is possible is found in Reference [2], graciously supplied online by its author DSP guru fred harris.

As additional examples, we show the result of our standard IIR -to- dual-path allpass conversion for standard 3rd-, 7th-, and 9th-order IIR filters in Figure 4.

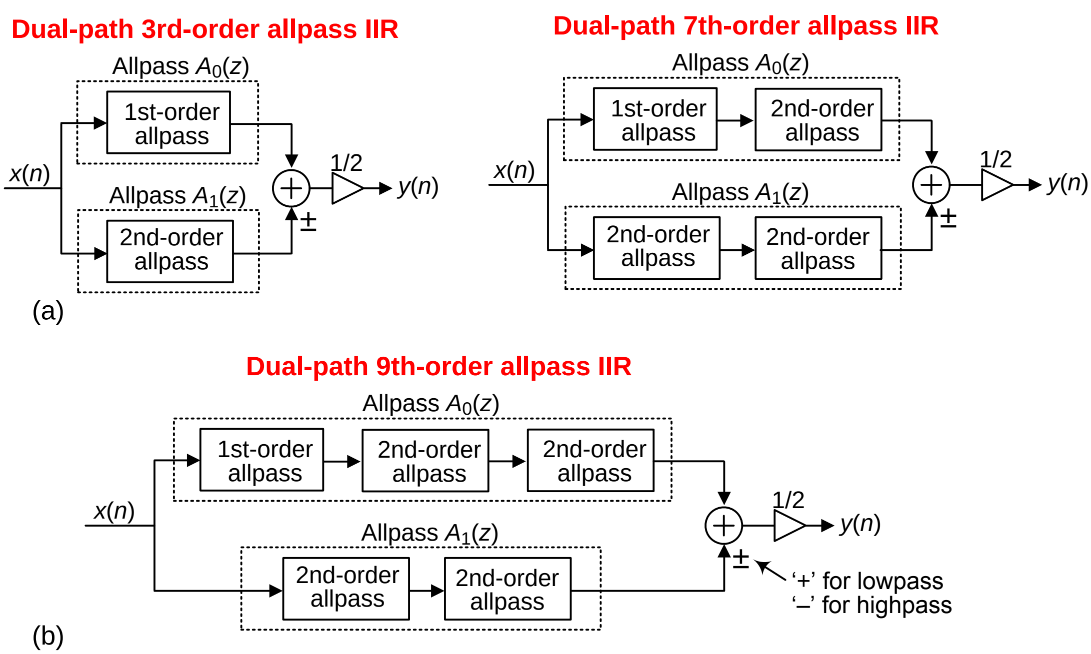


FIGURE 4. Dual-path allpass IIR filters: (a) 3rd- and 7th-order; (b) 9th-order.

Computational Savings When Using the Dual-Path Allpass Filters

Table 1 compares the computational workloads of traditional Direct Form I IIR filters relative to the dual-path IIR filters in Figures 3 and 4. The bottom row of Table 1 illustrates the computational advantage of the dual-path allpass filters.

Table 1. *N*th-Order IIR Filter Computations per output (*N* is odd).

	Multiplies	Adds	Delays
*N*th-order Direct Form I IIR filter	2*N*+1	2*N*	2*N*
*N*th-Order Direct Form I IIR Filter utilizing numerator coefficient symmetry	(3*N*+1)/2	2*N*	2*N*
Proposed *N*th-order dual-path IIR filter (using allpass sections)	***N***	**2*N***	***N*+1**

Additional information regarding our 'IIR –to– dual-path allpass filter conversion' process is provided in the Appendices of this document. That information is:

• Appendix A: Allpass Filter Conversion Implementation Considerations,
• Appendix B: Standard IIR –To– Dual-Path Allpass Filter Conversion Design Example, and
• Appendix C: MATLAB code to compute dual-path allpass filter coefficients.

References

[1] S. Mitra, Digital filter lecture slides (allpass filter discussion starts at slide# 23). Available at: http://www.emba.uvm.edu/~gmirchan/classes/EE275/Handouts_Ed4/Ch07(4e)Handouts/Ch7(1)Handouts_

[2] f. harris, "A Most Efficient Digital Filter: The Two-Path Recursive All-Pass Filter", *Streamlining Digital Signal Processing, A Tricks of the Trade Guidebook* (IEEE Press/Wiley, 2007), Chapter 9, pp. 85-104. Available at: https://www.researchgate.net/publication/278320928_A_Most_Efficient_Digital_Filter_The_Two-Path_Recursive_All-Pass_Filter

[3] P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice-Hall, Upper Saddle River, New Jersey, 1993, pp. 89-90.

[4] S. Mitra, *Digital Signal Processing, A computer-Based Approach, 4th Edition*, McGraw-Hill, New York, New York, 2011, pp. 462-464.

Appendix A: Allpass Filter Conversion Implementation Considerations

A-1: Restrictions on the Original Direct IIR Filter

To use this document's technique for converting a traditional IIR filter to an efficient parallel-path allpass structure, the original IIR filter must:

* Be lowpass or highpass with no poles on or outside the unit cycle.
* Have a transfer function with numerator and denominator polynomials of equal and odd-order. (If the original IIR filter is even order the allpass coefficients will be complex-valued which negates the

computational advantage of using allpass filters.)
* Have a maximum frequency magnitude response of one (unity).
* Have symmetrical or anti-symmetrical numerator coefficients.

Filters designed with MATLAB's ellip(), butter(), cheby1(), and cheby2() commands satisfy the last two of the above restrictions.

A-2: Mathematical Considerations

The final multiplications by 1/2 at the right sides of Figures 1 and 2, that can be implemented with a binary right shift, ensures that the parallel-path filter has a maximum gain of one. Whether the two allpass parallel paths' outputs are added or subtracted depends on whether the original IIR filter is lowpass or highpass.

In floating-point numerical implementations the original IIR and parallel-path allpass filters in Figure 2 have identical frequency magnitude responses.

The good news is that the Figure 3 allpass sections maintain their allpass behavior when, and have low frequency magnitude response sensitivity to, quantized coefficients used as demonstrated on pages 712-713 of Reference [4].

A-3: Software Considerations

Figure A-1 shows the structures of the 1st- and 2nd-order allpass filters, as well as the software commands needed to compute a single output sample. The underlined operations in Figure A-1 perform the data shifts through the delay elements in anticipation of the arrival of the next *x*(*n*) input sample.

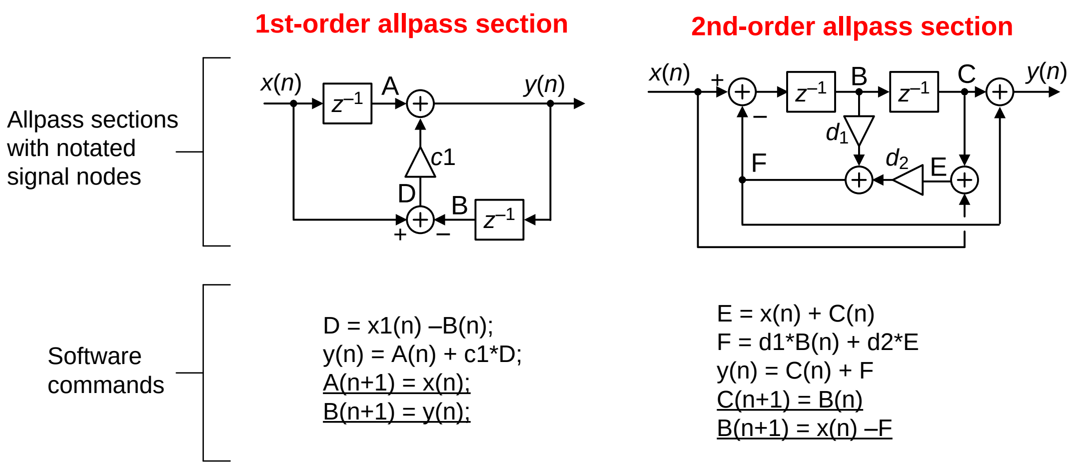


FIGURE A-1: 1st- and 2nd-order allpass filters and their implementation software commands.

Appendix B: How the Standard IIR -To- Dual-Path Allpass Filter Conversion Process Works

Here we show how this 'IIR -to- dual-path allpass filter conversion' process works by way of example. Let's start with a 5th-order lowpass IIR filter defined by the MATLAB command:

```
[b,a] = cheby1(5,0.2,0.15); % 0.2 dB passband ripple, .15 cutoff freq
```

The transfer function of this 5th-order IIR filter is:

$$H_{\mathrm{IIR}}(z) = \frac{Y_{\mathrm{IIR}}(z)}{X_{\mathrm{IIR}}(z)}$$

$$= \frac{0.0002 + 0.0008z^{-1} + 0.0015z^{-2} + 0.0015z^{-3} + 0.0008z^{-4} + 0.0002z^{-5}}{1 - 4.0501z^{-1} + 6.8238z^{-2} - 5.9479z^{-3} + 2.6745z^{-4} - 0.4955z^{-5}} \qquad \text{(B-1)}$$

Next, we compute the filter's five pole locations and plot them in Figure B-1(a).

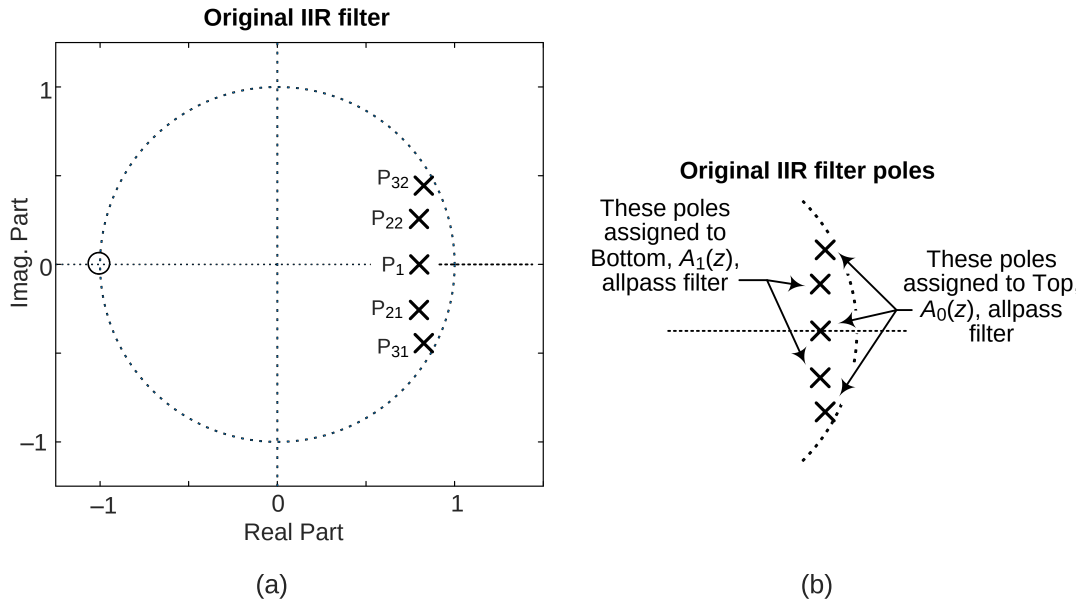


FIGURE B-1: 'Original IIR filter poles.

Those pole values, sorted from their smallest to their largest magnitudes, are:

$P_1 = 0.8005$
$P_{21} = 0.7989 - j0.2566$
$P_{22} = 0.7989 + j0.2566$
$P_{31} = 0.8259 - j0.4439$
$P_{32} = 0.8259 + j0.4439$

The real-valued pole P_1 is assigned to the denominator coefficient of a 1st-order allpass section. Because the original IIR filter is always odd-order, there will always be a real-valued pole assigned to a 1st-order allpass section. The denominator of that 1st-order allpass section's transfer function, in terms of z^{-1}, is $(1 - P_1 z^{-1})$. So the denominator coefficients of the 1st-order allpass filter are

1st-order allpass section denominator coeffs = $[1, -P_1]$ = [1 -0.8005].

The complex-conjugate pole pair P_{21}/P_{22} are assigned to the denominator coefficient of a 2nd-order allpass section. The denominator of that 2nd-order allpass section's transfer function, in terms of z^{-1}, is
$(1 - P_{21}z^{-1})(1 - P_{22}z^{-1}) = 1 - (P_{21}+P_{22})z^{-1} + P_{21}P_{22}z^{-2}$. So the denominator coefficients of this 2nd-order allpass filter are

2nd-order allpass section denominator coeffs = $[1, -(P_{21}+P_{22}), P_{21}P_{22}]$
= [1, -1.5978, 0.7041].

The complex-conjugate pole pair P_{31}/P_{32} are assigned to the denominator coefficient of another 2nd-order allpass section. The denominator of that 2nd-order allpass section's transfer function, in terms of z^{-1}, is $(1 - P_{31}z^{-1})(1 - P_{32}z^{-1}) = 1 - (P_{31}+P_{32})z^{-1} + P_{31}P_{32}z^{-2}$. So the denominator coefficients of this 2nd-order allpass filter are

$$\text{2nd-order allpass section denominator coeffs} = [1, -(P_{31}+P_{32}), P_{31}P_{32}] = [1, -1.6517, 0.8791].$$

So now, for this example, we have computed the denominator coefficients for a single 1st-order allpass filter section and two 2nd-order allpass filter sections. Those denominator coefficients are:

[1, -0.8005] ------------- Single 1st-order for Top-Path $A_0(z)$ section
[1, -1.5978, 0.7041] ----- First 2nd-order for Bottom-Path $A_1(z)$ section
[1, -1.6517, 0.8791] ----- Second 2nd-order for Top-Path $A_0(z)$ section

Using the "pole interlacing property" as shown in Figure B-1(b), based on their angles the alternate poles will be *assigned* to the top $A_0(z)$ and bottom $A_1(z)$ paths of the dual-path allpass filter in this document's Figure 2 [3,4].

The results of using the "pole interlacing property" is shown in Figure B-2. Notice how the dual-path allpass filter sections' numerator coefficients are their denominator coefficients in reverse order. The numerator coefficients produce the *z*-plane zeros shown in Figure B-2. The reversed-order coefficients ensure that the zeros' locations are the reciprocals of the poles' locations. That reciprocal property is necessary for the filter sections to have allpass frequency-domain behavior.

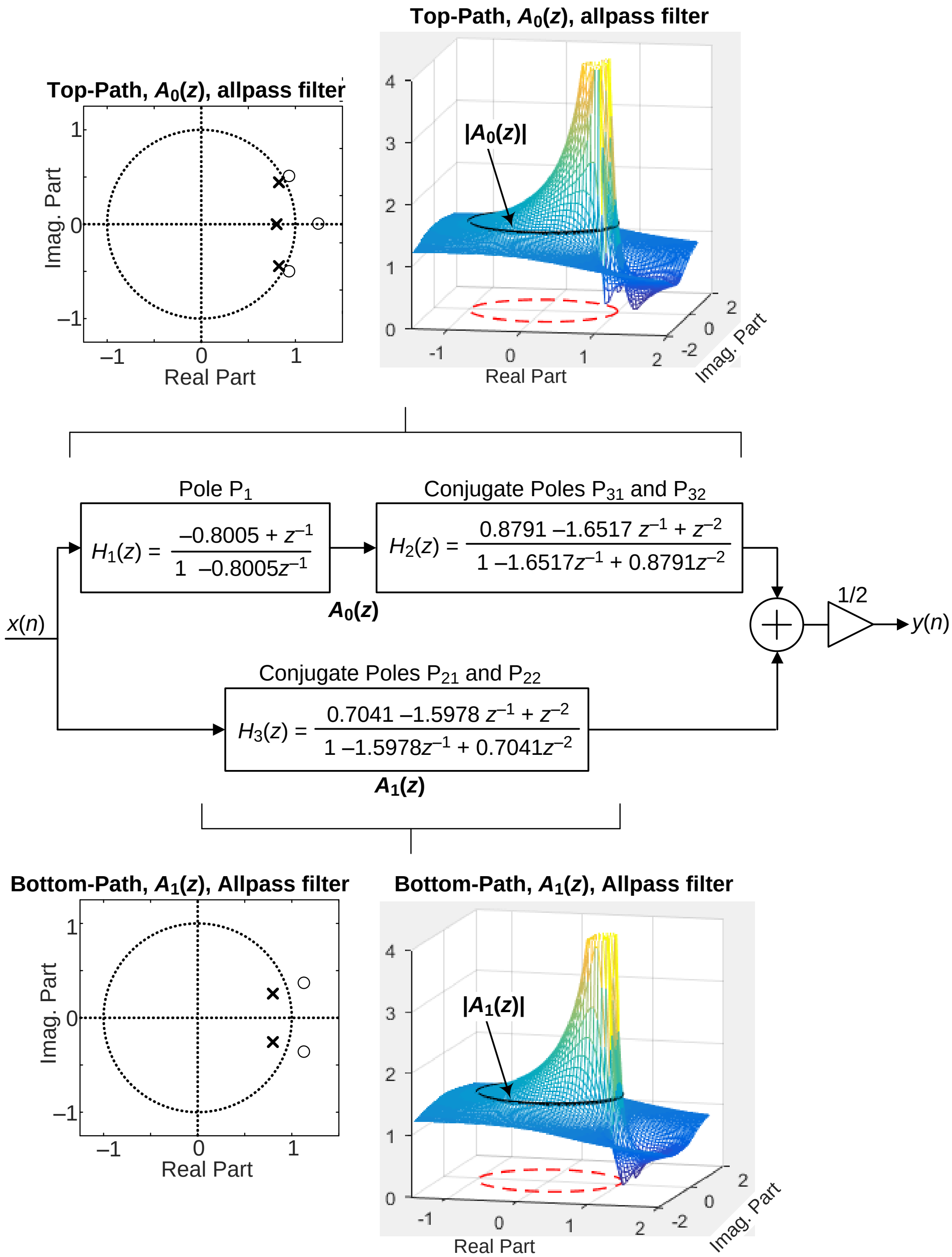


FIGURE B-2: Result of the example 'IIR –to- dual-path allpass filter conversion'.

The frequency magnitude responses of the original IIR filter and the resultant dual-path allpass filter are shown in Figure B-3.

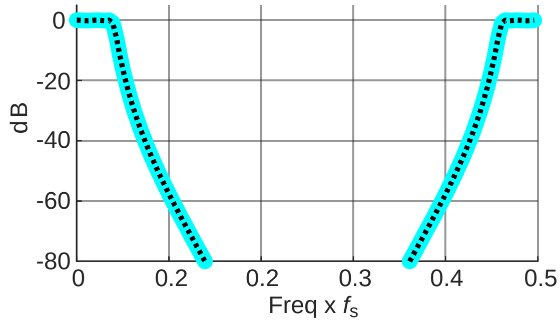


FIGURE B-3: Frequency magnitude responses. Thick shaded curve for the original IIR filter, black dashed curve for the dual-path allpass filter.

Based on the networks in this document's Figure 3, our example dual-path allpass filter in the center of Figure B-2 is implemented as shown in Figure B-4.

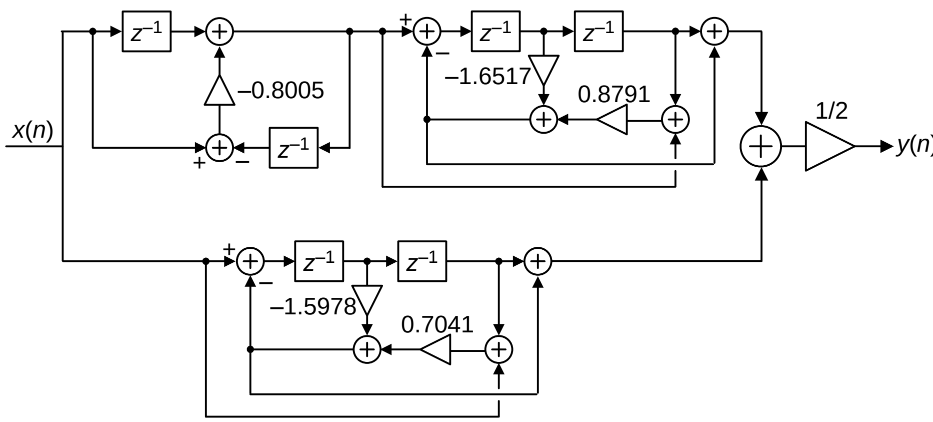


FIGURE B-4: Dual-path allpass filter implementation.

Appendix C: MATLAB Code To Compute Allpass Sections' Coefficients

The process of computing allpass sections' denominator coefficients comprises the following steps:

1. Compute input IIR filter's poles' & zeros locations.
2. Determine if input IIR filter is lowpass or highpass.
3. Sort IIR filter's poles, based on their magnitudes, to compute allpass sections' denominator coeffs.
4. Find the 1st- and 2nd-order sections' denominator coeffs.
5. Sort 'Denoms' matrix into "Bottom_Path_Denoms" & "Top_Path_Denoms" using the "Pole Interlacing Property".
6. Partition the 'Top_Path_Denoms' into 1st- & 2nd-order sections (the Bottom_Path_Denoms will always be the desired 2nd-order sections).
7. Convolve Top filter sections' coeffs to enable User spectral plotting.
8. Convolve Bottom filter sections' coeffs to enable User spectral plotting.

Once we know the dual-path allpass filter sections' denominator coefficients, the dual-path allpass filter sections' numerator coefficients are the denominator coefficients in reverse order.

The MATLAB code to implement the above eight steps is the following:

```
function [Top_1stOrd_Denom,Top_2ndOrd_Denoms,Bottom_Denoms,...
       Filter_Type,Top_Casc_Denoms,Bottom_Casc_Denoms] ...
         = Allpass_Find_Coeffs(Numer_IIR, Denom_IIR)
% Computes dual-path allpass denominator coefficients,
% using the "Pole Interlace Property" of standard IIR filters, as
% described on page 462 of Sanjit Mitra's "Digital Signal Processing,
% A computer-Based Approach" book and page 89 of Vaidyanathan's
% Multirate Systems book.
% The input IIR filter must be an odd-order lowpass or highpass filter.
% The dual-path allpass numerator coefficients are the denominator
% coefficients, computed by this function, reversed in order.

% Inputs:
%   Numer_IIR = numerator coeffs of an IIR lowpass or highpass filter.
%   Denom_IIR = denominator coeffs of an IIR lowpass or highpass filter
% Outputs:
%   Top_1stOrd_Denom = denominator coeffs of top path 1st-ord section
%   Top_2ndOrd_Denoms = denominator coeffs of top path 2nd-ord sections.
%   Bottom_Denoms = denominator coeffs of bottom path sections.
%   Filter_Type = Original IIR filter type: 'Lowpass' or 'Higpass'.
%   Top_Casc_Denom = Top path cascaded (convolved) denominator
%              coeffs (used for allpass filter freq response plotting).
%   Bottom_Casc_Denom = Bottom path cascaded (convolved) denominator
%              coeffs (used for allpass filter freq response plotting).
% ** Example **
% [b, a] = cheby1(5, .2, .15); % 5th-order lowpass IIR filter
% [Top_1stOrd_Denom, Top_2ndOrd_Denoms, Bottom_Denoms,...
%        Filter_Type, Top_Casc_Denoms, Bottom_Casc_Denoms] ...
%          = Allpass_Find_Coeffs(Numer_IIR, Denom_IIR)
% ** Example Results **
% Top_1stOrd_Denom = 1.0000   -0.8005
% Top_2ndOrd_Denoms = 1.0000   -1.6517    0.8791
% Bottom_Denoms = 1.0000   -1.5978    0.7041
% Filter_Type = Lowpass
% Top_Casc_Denom = 1.0000   -2.4523    2.2014   -0.7038
% Bottom_Casc_Denom = 1.0000   -1.5978    0.7041

    % Rick Lyons, May, 2019

% Compute Input IIR filter's poles' & zeros locations
IIR_Poles = roots(Denom_IIR);
IIR_Zeros = roots(Numer_IIR);

% Determine if input IIR filter is lowpass or highpass
if mean(real(IIR_Poles)) > mean(real(IIR_Zeros))
    Filter_Type = 'Lowpass';
else
    Filter_Type = 'Higpass';
end

% Sort IIR filter's poles, based on their magnitudes, to compute
% allpass sections' denominator coeffs
Sorted_Poles = sort(IIR_Poles);
Num_Poles = length(Sorted_Poles);

% Now find the 1st- and 2nd-order sections' denominator coeffs
% based on the single real pole and the complex conjugate pole pairs
    % Compute 1st-order section's denominator coeffs
    Denoms(1,:) = [1, -Sorted_Poles(1), 0];
```

```
    % Compute 2nd-order sections' denominator coeffs
    for K = 1:(Num_Poles-1)/2
        Denoms(K+1,:) = [1, -(Sorted_Poles(2*K)+Sorted_Poles(2*K+1)), ...
                       Sorted_Poles(2*K)*Sorted_Poles(2*K+1)];
    end

% Sort 'Denoms' matrix into "Bottom_Denoms" & "Top_Denoms"
% using the "Pole Interlace Property".
[Number_of_Demon_Rows,Temp] = size(Denoms); % Nu rows in Demons matrix
Top_Denoms = []; % Initializ
Bottom_Denoms = []; % Initialize
    % Perform allpass pole interlacing based on whether the
    % variable 'Number_of_Demon_Rows' is an odd or even number
    if Number_of_Demon_Rows == 2*floor(Number_of_Demon_Rows/2)
        % Number_of_Demon_Rows is even
        Swap = 'N';
        for K = 1:ceil(Number_of_Demon_Rows/2)
            Top_Denoms(K,:) = Denoms(K+(K-1),:);
        end
        for K = 1:ceil(Number_of_Demon_Rows/2)
            Bottom_Denoms(K,:) = Denoms(2*K,:);
        end
    else
        % Number_of_Demon_Rows is odd
        Swap = 'Y';
        for K = 1:ceil(Number_of_Demon_Rows/2)
            Top_Denoms(K,:) = Denoms(K+(K-1),:);
        end
        for K = 1:floor(Number_of_Demon_Rows/2)
            Bottom_Denoms(K,:) = Denoms(2*K,:);
        end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Break 'Top_Denoms' into 1st- & 2nd-order sections
Top_1stOrd_Denom = Top_Denoms(1,1:2); % Eliminate zero coeff
Top_2ndOrd_Denoms = Top_Denoms(2:end, 1:3); % 2nd, 3rd, 4th,... rows

    % Check, and correct, if no Top 2nd-order denominators
    Size_of_Top_Denoms = size(Top_Denoms);
    if Size_of_Top_Denoms(1) == 1 % 'Top_Denoms' has no 2nd-order parts
      disp('Allpass Top-path is 1st-order only (no 2nd-order sections)')
      Top_2ndOrd_Denoms = [0, 0, 0];
    else,end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Convolve Top filter sections' coeffs for spectral plotting
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Top_Casc_Denoms = Top_Denoms(1,1:2);
Top_Casc_Denoms = Top_1stOrd_Denom;
[Top, Temp] = size(Top_Denoms); % Numb of 'Top_Denoms' rows
for P = 2:Top
    %Top_Denoms(P,:);
    Top_Casc_Denoms = conv(Top_Casc_Denoms, ...
                           Top_Denoms(P,:));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Convolve Bottom filter sections' coeffs for spectral plotting.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Bottom_Casc_Denoms = [1];
Bottom_Casc_Denoms = Bottom_Denoms(1,:);
[Bot, Temp] = size(Bottom_Denoms);
for Q = 2:Bot
    Bottom_Casc_Denoms = conv(Bottom_Casc_Denoms, ...
                                 Bottom_Denoms(Q,:));
end
```