Converting Infinite Impulse Response Filters to Parallel Form

Balázs Bank, *Member, IEEE*

***Abstract*—Discrete-time rational transfer functions are often converted to parallel second-order sections due to better numerical performance compared to direct form infinite impulse response (IIR) implementations. This is usually done by performing partial fraction expansion over the original transfer function. When the order of the numerator polynomial is greater or equal to that of the denominator, polynomial long division is applied before partial fraction expansion resulting in a parallel finite impulse response (FIR) path.**

This article shows that applying this common procedure can cause a severe dynamic range limitation in the filter because the individual responses can be much larger than the net transfer function. This can be avoided by applying a delayed parallel form where the response of the second-order sections is delayed in such a way that there is no overlap between the IIR and FIR parts. In addition, a simple least-squares procedure is presented to perform the conversion which is numerically more robust than the usual Heaviside partial fraction expansion. Finally, the possibilities of converting series second-order sections to the delayed parallel form are discussed.

***Index Terms*—digital signal processing, infinite impulse response (IIR) digital filters, partial fraction expansion, parallel second-order sections, least-squares method.**

I. Introduction

IIR digital filters are part of most signal processing algorithms. They are not only used in classic filtering applications (low-pass, high-pass, and so on), but also as tools for approximating any given transfer function, e.g., a measured frequency response that we wish to model in discrete time. Compared to FIR filters, IIR filters typically require lower computational resources for the same modeling accuracy. However, care has to be taken to assure their stability: a theoretically stable IIR filter might become unstable when implemented with finite coefficient precision. The problem becomes pronounced when the filter has high order and/or has poles near the unit circle. As a remedy, IIR filters are often implemented as a series or parallel combination of (typically, second-order) subfilters [1].

The conversion to series second-order sections starts with finding the poles p_n and zeros z_m of the transfer function

$$H(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + z^{-N}}, \quad (1)$$

B. Bank is with the Department of Measurement and Information Systems, Budapest University of Technology and Economics, H-1521 Budapest, Hungary (e-mail: bank@mit.bme.hu).

This is the author's version of the manuscript which differs from the one published by IEEE in its formatting. The version published by IEEE is available at: https://doi.org/10.1109/MSP.2018.2805358

resulting in

$$H(z^{-1}) = K\frac{(1 - z^{-1}z_1)(1 - z^{-1}z_2)\ldots(1 - z^{-1}z_M)}{(1 - z^{-1}p_1)(1 - z^{-1}p_2)\ldots(1 - z^{-1}p_N)}. \quad (2)$$

Finally, the complex-conjugate pairs of poles and zeros are recombined to form second-order sections. Carefully pairing those poles and zeros is of utmost importance, and the ordering of the sections is also critical as it influences the roundoff noise and dynamic range of the filter [1].

Today, the parallel implementation is gaining more and more interest since it provides several advantages compared to series biquads: it has lower quantization noise [2], and even more importantly, it leads to a significant speedup in modern multicore processors that can take advantage of the fully parallel filter structure [3].

While alternative methods are available for direct-to-parallel conversion [4], [5], by far the most common way of converting filters to parallel form is based on partial fraction expansion [1]. Here the first step is converting the transfer function (1) to the residue form

$$H(z^{-1}) = \frac{r_1}{1 - p_1 z^{-1}} + \frac{r_2}{1 - p_2 z^{-1}} + \ldots + \frac{r_N}{1 - p_N z^{-1}}, \quad (3)$$

where r_n are the residues corresponding to the poles p_n. The usual way of determining r_n is the Heaviside cover-up method, which can be formulated mathematically as

$$r_n = (1 - z^{-1}p_n)H(z)\big|_{z=p_n}. \quad (4)$$

Note that (3) and (4) are general only if poles p_n are distinct. In the case of pole multiplicity, higher order terms also appear [6].

The partial fraction expansion requires that the transfer function $H(z)$ is strictly proper, that is, the order of the denominator is larger than the order of the numerator ($N > M$). If this is not the case, polynomial long division has to be performed to result in a FIR part $F(z^{-1})$ and a strictly proper IIR part $B'(z^{-1})/A(z^{-1})$ as

$$H(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{B'(z^{-1})}{A(z^{-1})} + f_0 + f_1 z^{-1} + \ldots + f_K z^{-K}, \quad (5)$$

where $K = M - N$ is the order of the FIR part. Partial fraction expansion, including the polynomial long division, is performed by the MATLAB/Octave function *residuez* included in the Signal Processing Toolbox. The last step of the conversion is combining the complex-conjugate pairs to second-order sections having real coefficients:

$$H(z) = \sum_{l=1}^{L} \frac{b_{0,l} + b_{1,l} z^{-1}}{1 + a_{1,l} z^{-1} + a_{2,l} z^{-2}} + \sum_{k=0}^{K} f_k z^{-k}. \quad (6)$$

To my knowledge, the method discussed so far is the one that is covered in all digital signal processing textbooks. However, this common method, when implemented in real-world processors, may produce several problems. First, the overlap between the FIR and IIR parts lead to a dynamic range problem, and thus, increased quantization noise. Second, the expansion can be numerically sensitive leading to problems when factoring high-order ($N > 100$) transfer functions. These problems are addressed in this article, in addition to covering the case of converting filters given in series or pole-zero form.

All of the MATLAB codes used to create the figures in this article are available for download, which can help the interested reader gain more insight to the presented algorithms (see the "Supplementary Material" section).

II. The Dynamic Range Problem

As can be seen in (6), both the IIR part $B'(z^{-1})/A(z^{-1})$ and the FIR part $F(z^{-1})$ have an impulse response starting at sample zero, meaning that the first $K+1$ samples are determined by the combination of the FIR and IIR parts. This overlap leads to the fact that the individual transfer functions of the filter sections can be significantly larger than the net transfer function, leading to the limitation of practical dynamic range [7].

This is illustrated in Figure 1(a) for an IIR filter designed to model a measured loudspeaker response. The sampling rate is $f_s = 44.1$ kHz in all of the examples in this article. First a direct-form IIR filter is designed by the Steiglitz-McBride method [8] (*stmcb* command in MATLAB) shown by a thick blue line, and then converted to a parallel set of second-order sections plus a FIR part as described in the "Introduction" section. The orders of the numerator and the denominator are both 20, with the short notation: (20/20). This results in 10 second-order sections plus a constant gain section in parallel. The red dashed line shows the net transfer function, overlapping the direct-form transfer function (thick blue line) perfectly. The black dashed line displays the transfer function of the FIR part $F(z^{-1})$, which is now a constant gain, while the thin colored lines correspond to the magnitude responses of the individual second-order sections.

Figure 1(a) shows that some of the individual transfer functions (in this case, the constant gain part displayed by the black dashed line and one second-order transfer function displayed by a thin purple line) are significantly larger than the net transfer function. Figure 1(a) also shows only magnitude responses; therefore it cannot be seen that these two upper curves have almost opposite phase, and the required net response is a result of the phase cancellation of these individual responses. In theory, this does not lead to any difficulties. However, when the filter is implemented with finite word-length, this requires the downscaling of the input signal to avoid overload, which, in turn, will lead to increased quantization noise. (No direct downscaling is required in floating-point implementation, since it is done "automatically" by the number format. However, the dynamic range reduction remains the same.)

This dynamic-range problem becomes even more pronounced if the order of the numerator M is larger than that

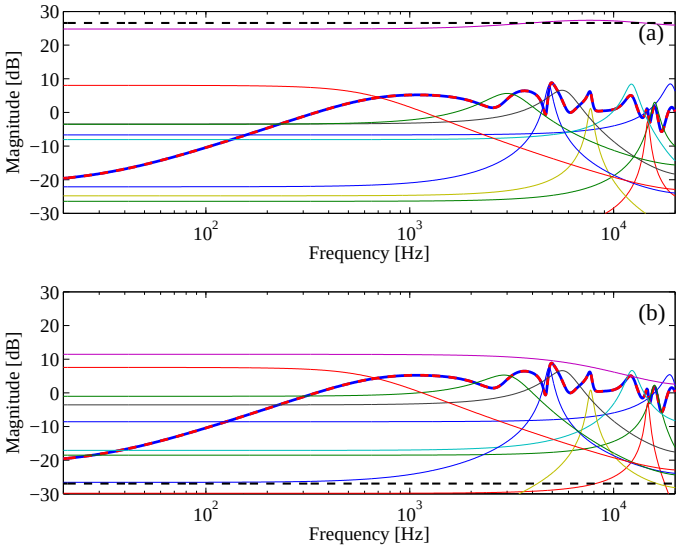

Fig. 1. The parallel implementation of a (20/20) transfer function: (a) The traditional parallel form and (b) the delayed parallel form. The thick blue line shows the original transfer function and the red dashed line displays the net response of the parallel implementation. The thin colored lines display the magnitude responses of the second-order sections, and the black dashed line shows the magnitude response of the constant gain path.

of the denominator N. This is illustrated in Figure 2(a) for numerator and denominator orders of 25 and 20, respectively (25/20). Figure 2(a) shows that now the fifth-order FIR part (black dashed line) is around 70 dB larger than the net transfer function, decreasing the signal-to-noise ratio by 70 dB due to the required downscaling. There is a red line very close to the black dashed line that again corresponds to a second-order transfer function with almost opposite phase compared to the parallel FIR part.

III. The Delayed Parallel Form

The aforementioned dynamic range problem becomes particularly serious in such constellations when the net transfer function is a result of phase cancellations between the FIR and IIR parts. This can be completely avoided if we do not allow any overlap between the FIR and IIR part [7]. This results in the form

$$H(z^{-1}) = z^{-(K+1)} \sum_{l=1}^{L} \frac{\tilde{b}_{l,0} + \tilde{b}_{l,1} z^{-1}}{1 + a_{l,1} z^{-1} + a_{l,2} z^{-2}} + \\ + \sum_{k=0}^{K} \tilde{f}_k z^{-k}, \tag{7}$$

where the first $K+1$ samples of the impulse response are now determined solely by the K-th order FIR part, and the rest of the impulse response by the IIR part.

The coefficients of the delayed form can be obtained in a very similar way as those of the traditional parallel form discussed in the "Introduction". The only difference is that now the FIR part $\tilde{F}(z^{-1})$ and the strictly proper numerator $\tilde{B}'(z^{-1})$ is computed by performing polynomial long division over the reversed numerator coefficients [6], [7] to get

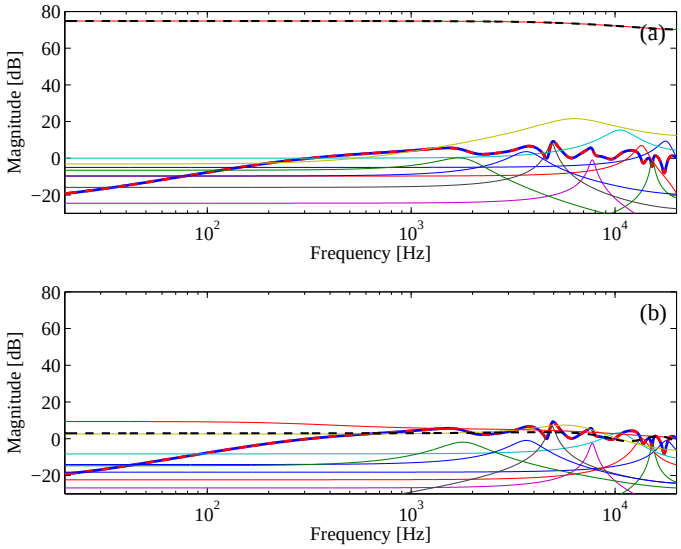

Fig. 2. The parallel implementation of a (25/20) transfer function: (a) The traditional parallel form and (b) the delayed parallel form. The thick blue line shows the original transfer function and the red dashed line displays the net response of the parallel implementation. The thin colored lines display the magnitude responses of the second-order sections, and the black dashed line shows the magnitude response of the parallel FIR path.

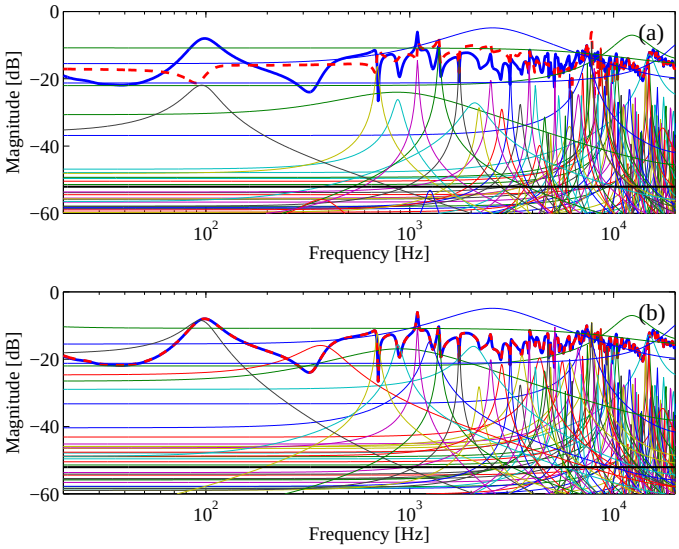

Fig. 3. The delayed parallel implementation of a (200/200) transfer function: conversion done (a) by partial fraction expansion and (b) by a least-squares fit. The thick blue line is the original transfer function and the red dashed line is the net transfer function of the delayed parallel form. The thin colored lines show the responses of the individual second-order sections and the black line displays the transfer function of the constant gain.

$$H(z^{-1}) = z^{-(K+1)}\frac{\tilde{B}'(z^{-1})}{A(z^{-1})} + \tilde{f}_0 + \tilde{f}_1 z^{-1} + \ldots + \tilde{f}_K z^{-K}. \quad (8)$$

Next, the transfer function $\tilde{B}'(z^{-1})/A(z^{-1})$ is expanded to partial fractions. This complete procedure is performed by the *residued* command distributed in Octave. The *residue* function in MATLAB originally intended for converting s-domain transfer functions can also be used for the $M \geq N$ case (note however that it produces wrong results for $M < N$) [7]. Finally, the partial fractions are recombined to second-order sections in the same way as for the nondelayed case.

Note that, if the transfer function is available in the non-delayed parallel form, it is not necessary to convert it back to direct-form to obtain the coefficients of the delayed form: in [7] a simple procedure is proposed to convert between the nondelayed and delayed parallel forms.

Next, we see that such a simple change has a dramatic effect on the dynamic range requirements of the filters: Figure 1(b) shows the same (20/20) transfer function as in Figure 1(a), but now the IIR part is delayed by one sample so that there is no time domain overlap with the constant gain part. Now the individual transfer functions are only approximately 5 dB larger than the net transfer function, which is 15 dB smaller compared to that of the traditional parallel form of Figure 1(a).

Even more pronounced is the difference for the (25/20) transfer function of Figure 2(a): with the delayed form displayed in Figure 2(b), the need for downscaling the input signal by 70 dB is completely eliminated, leading to a drastic improvement in signal-to-noise ratio.

IV. Numerical issues with partial fraction expansion

For moderate filter orders (< 100), obtaining the parallel form of IIR filters by partial fraction expansion is the most practical option. However, for higher filter orders the conversion can lead to numerical errors. This is displayed in Figure 3(a), where the blue line shows a 200th order IIR filter (200/200) designed by the Steiglitz-McBride method [8] (*stmcb* command in MATLAB) to model a measured room response. The red dashed line is the net transfer function of a delayed parallel filter obtained by the procedure already outlined, that is, performing polynomial long division on the reversed numerator polynomial and then partial fraction expansion. It can be seen that the magnitude response of the converted filter does not match that of the original, which is due to numerical errors.

V. Obtaining the parallel form by a least-squares fit

Instead of formulating the problem in terms of converting rational expressions, here we take a different approach where the numerator parameters of the parallel filter are determined in such a way that the impulse response of the parallel implementation is the closest possible to the impulse response of the original filter in the mean square sense.

This least-squares procedure gives the parameters of the delayed parallel form directly and is robust even for large (≈ 1000) filter orders. The method is inspired by the fixed-pole design of parallel filters [9], a methodology used for obtaining IIR filters having uneven (e.g., logarithmic)-frequency resolution. First, the roots of denominator $A(z^{-1})$ are found that are used to form the denominator polynomials of the second-order

sections $A_l(z^{-1})$. Next, the numerators of the sections are obtained via a least-squares fit such that the difference between the impulse responses of the original and parallel structures is minimized. (Alternatively, it is also possible to minimize the difference between the complex transfer functions of the direct form and parallel form filters, leading to practically the same results.) This can be done easily because the transfer function (7) becomes linear in its free parameters $\tilde{f}_k$, $\tilde{b}_{l,0}$, $\tilde{b}_{l,1}$ once the denominator coefficients $a_{l,1}$, $a_{l,2}$ are determined.

While the procedure is also applicable to the traditional, nondelayed parallel form, it will be illustrated for the numerically better-performing delayed version. We can see in (7) for $M > N$ that the first $K + 1 = M - N + 1$ samples of the impulse response are solely determined by the FIR coefficients $\tilde{f}_k$, and thus the problem reduces to finding the parameters of the second-order sections such that the resulting impulse response is the closest to the original impulse response starting from sample $K + 1$.

The steps of the conversion are given next. The procedure is outlined for the case of no pole multiplicity. In the case of repeated poles, terms of higher than second-order must also be included, similarly to the case of partial fraction expansion.

1) Compute the roots p_n of the denominator $A(z^{-1})$, flip the unstable poles $|p_n| > 1$ inside the unit circle by replacing them with $1/p_n$, find the complex-conjugate pairs and recombine the denominators of the second-order sections $A_l(z^{-1})$.
2) Compute the impulse response $h(i)$ of the filter $H(z^{-1}) = B(z^{-1})/A(z^{-1})$ for samples $i = 0 \dots I$.
3) For $M \geq N$, the coefficients of the FIR part equal to the first $K = M - N + 1$ samples of the filter impulse response, that is, $\tilde{f}_k = h(k)$ for $k = 0 \dots K$.
4) Compute the impulse responses $u_l(i)$ of the numerators $1/A_l(z^{-1}) = 1/(1 + a_{l,1}z^{-1} + a_{l,2}z^{-2})$.
5) Find the numerator coefficients $\tilde{b}_{l,0}$, $\tilde{b}_{l,1}$ by a least-squares fit such that the resulting impulse response

$$\tilde{h}(i) = \sum_{l=1}^{L} \tilde{b}_{l,0} u_l(i) + \tilde{b}_{l,1} u_l(i-1) \tag{9}$$

is closest to the impulse response of the original filter $h(i)$ starting from sample $i = K + 1$.
Since (9) is linear in its free parameters $\tilde{b}_{l,0}$, $\tilde{b}_{l,1}$, it can be written in a matrix form

$$\tilde{\mathbf{h}} = \mathbf{U}\tilde{\mathbf{b}}, \tag{10}$$

where $\mathbf{U}$ contains the impulse responses $u_l(i)$ and their delayed versions $u_l(i-1)$ in its columns, and $\tilde{\mathbf{b}}$ is a column vector composed of the corresponding $\tilde{b}_{l,0}$ and $\tilde{b}_{l,1}$ values. Now the resulting impulse response vector $\tilde{\mathbf{h}}$ should be the closest possible to the target $\mathbf{h}$ vector containing the samples $h(i)$ from $i = K+1$ in the least-squares sense. This is a standard linear least-squares problem (overdetermined set of equations) and can be solved by the *mldivide* function in MATLAB/Octave by using the syntax $\tilde{\mathbf{b}} = \mathbf{U}\backslash\mathbf{h}$.

Figure 3 (b) shows the net transfer function of the delayed parallel form when the conversion is done by the aforementioned least-squares fit. Now the conversion is much more accurate compared to the one obtained by using partial fraction expansion shown in Figure 3(a). As for the size of the least-squares problem, the impulse response fit was made for $I = 2M = 400$ samples, to have more equations than unknowns (we have $M = 200$ free parameters).

Filter order	Error with Partial Fraction Expansion Conversion	Error with Least-Squares Conversion
(50/50)	2.36×10^{-10} dB	3.86×10^{-10} dB
(100/100)	7.97×10^{-4} dB	5.52×10^{-8} dB
(200/200)	2.84 dB	6.78×10^{-8} dB
(500/500)	4.45 dB	7.02×10^{-8} dB
(1000/1000)	NaN	1.70×10^{-7} dB
(1500/1500)	NaN	1.34×10^{-6} dB

TABLE I
THE MEAN ABSOLUTE DECIBEL ERRORS OF THE CONVERTED TRANSFER FUNCTIONS.

Table I lists the mean absolute dB errors computed between the original and converted transfer functions in the range of 20 Hz and 22.05 kHz for various filter orders, including the (200/200) example of Figure 3. The significantly better accuracy of the least-squares procedure is apparent starting from order 100. For the orders of 1000 and 1500, some of the extracted poles p_n are outside the unit circle, thus, the partial fraction expansion based method leads to an unstable filter. On the other hand, the proposed procedure still produces accurate results since it starts with stabilizing the poles by flipping them inside the unit circle in step 1.

The reason for the significantly better performance compared to partial fraction expansion is that the numerical errors in finding the poles are compensated by the numerators of the second-order sections: the least-squares fit will give the best possible impulse response match for the given (slightly inaccurate) denominators. Besides simplicity, the ability to correct the numerical errors of pole finding (even unstable poles) is a great benefit of the LS design also compared to other conversion methods proposed previously [4], [5].

VI. CONVERSION FROM SERIES OR POLE-ZERO FORM

In some situations, the transfer function we are converting to a parallel form is given as a series of second-order sections, or, equivalently, in pole-zero form. Examples include classic low-pass, band-pass filters such as Butterworth, Chebyshev, etc. [1]. The *butter*, *cheby1*, etc. commands in MATLAB/Octave can give the pole-zero versions of the filters making the implementation possible for such a high-order/low-cutoff-frequency filters where the direct form implementation is unfeasible due to numerical reasons. Other examples can be series graphic or parametric equalizers [10] and equalizer filters iteratively designed directly in the series form [11] or obtained from a warped IIR design [12].

For strictly proper transfer functions ($M < N$) the partial fraction expansion works well since the poles are either known (pole-zero form) or computed from second-order polynomials (series second-order form), and the numerator can also be evaluated in its factored form.

On the other hand, when $M \geq N$, we need to perform polynomial long division to reduce the order of the numerator, and for that the denominator and numerator have to be recombined from the poles and zeros. At this is point we would loose all the numerical benefits coming from the fact that pole-zero form of the filter is known instead of the rational form. Therefore, this procedure is not recommended.

A. Least-squares fit

One possibility is obtaining the delayed parallel form by the least-squares fit as we have seen before for the direct-to-parallel conversion case. Of course, the target impulse response $h(i)$ is computed by running the series version of the filter, and we are not converting the series or pole-zero form to direct form. Also, the numerically problematic root finding is avoided, since either the poles, or the second-order denominators, are known.

Such a conversion example is presented in Figure 4. The measured transfer function of an average living room is the starting point to design a warped IIR filter. Warped filter design is one possible methodology to obtain filters with logarithmic-like frequency resolution to fit the resolution in human hearing, a desirable property in audio applications. However, warped filters require a complicated all-pass-based structure for implementation. This can be avoided when they are converted series second-order sections [12]. To show the robustness of the conversion method, in the example of Figure 4 a 1000th order warped IIR filter is designed by the *prony* function in MATLAB based on the warped room response (the warping parameter is $\lambda = 0.8$). Next, the warped IIR filter is converted to series second-order sections [12]. The net transfer function of the 500 series second-order sections is displayed by blue line. Finally, the filter is converted to the delayed parallel form by the least-squares method, displayed by a red line. The two transfer functions in Figure 4 match perfectly: the mean absolute dB error from 20 Hz to 22.05 kHz is 5.70×10^{-12} dB.

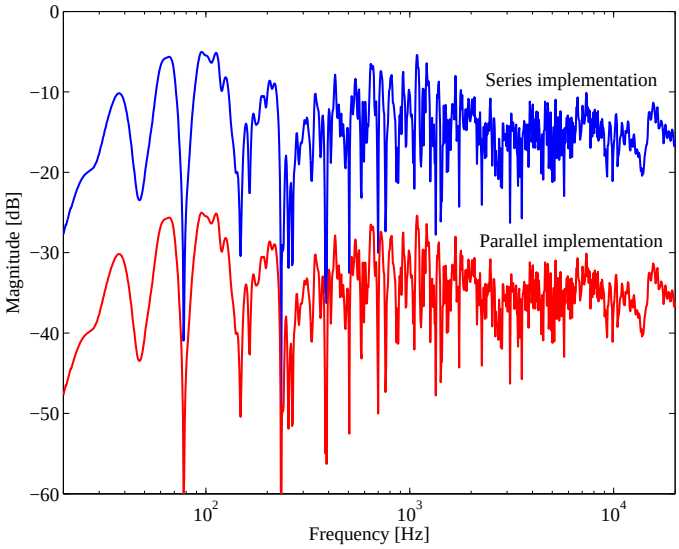

Fig. 4. The parallel implementation of a (1000/1000) transfer function. The blue line shows the response of the 500 series second-order sections, which is then converted to 500 parallel second-order sections by the least-squares fit, displayed by red line. The red curve is offset by -20 dB for clarity.

B. Factoring out zeros before partial fraction expansion

If we are willing to give up the full parallelization of the filter structure, there is another possibility for converting the series form to the parallel form: factoring out zeros from the numerator until we reach $M < N$. For the typical scenario of $M = N$, this means factoring out one real zero, or, if there are no real zeros, a complex conjugate pair. Now the filter structure is a set of parallel second-order sections (without additional delay) and a first- or second-order FIR filter in series. For the case of factoring out one real zero, we obtain:

$$H(z^{-1}) = (1 - z^{-1}z_1)\left(\sum_{l=1}^{L} \frac{\hat{b}_{0,l} + \hat{b}_{1,l}z^{-1}}{1 + a_{1,l}z^{-1} + a_{2,l}z^{-2}}\right). \quad (11)$$

The conversion is illustrated in Figure 5 for an 8th order high-pass Butterworth filter designed by the *butter* command in MATLAB. The thick blue line shows the magnitude response of the series second-order implementation obtained from the pole-zero form (the direct-form implementation is unstable, thus, not shown). Since all the zeros of the Butterworth filter equal to unity $z_m = 1$ for $m = 1 \ldots 8$, we simply factor out one of them before performing partial fraction expansion. Thus, the resulting filter structure is composed of four second-order IIR filters in parallel and an FIR filter $\hat{F}(z^{-1}) = 1 - z^{-1}$ in series. Obviously, the four second-order sections can be computed in parallel, but the first-order FIR filter cannot, which might lead to a slight increase in computational time in parallel architectures.

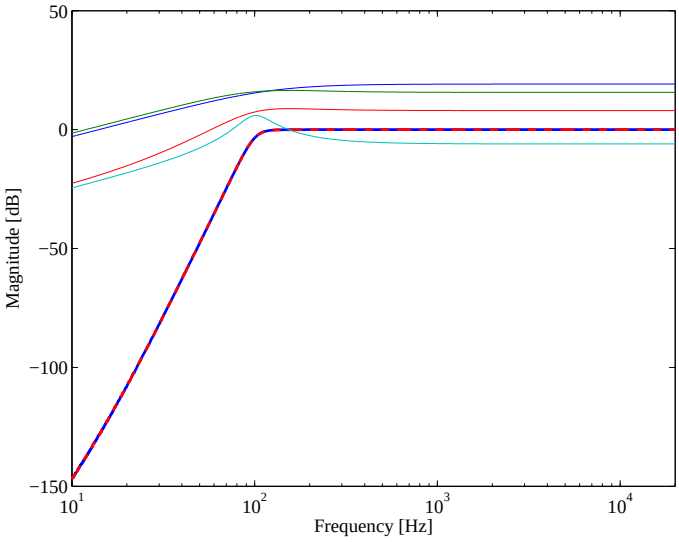

Fig. 5. The factored parallel implementation of an 8th order Butterworth high-pass filter. The cutoff frequency is $f_c = 100$ Hz, while the sampling rate is $f_s = 44.1$ kHz. The thick blue line shows the response of the 4 series second-order sections, and the red dashed line displays the response of the factored parallel implementation. The dotted lines display the magnitude responses of the parallel second-order sections in series with the response of the first-order FIR filter that has been factored out.

The benefit of the factored conversion is the significantly

reduced design time. Since no root finding is required, the partial fraction expansion requires very few (though complex) operations. This allows computing the parallel form on the fly when the user is manipulating the parameters of the series transfer function in real time, e.g., by changing the cutoff frequency of the high-pass filter. Another typical example would be varying the parameters of a parametric or graphic equalizer in series form and converting it to the more efficient parallel form in real time.

We note that, for high-order filters such as in the example of Figure 4, partial fraction expansion can lead to numerical errors even when converting from the series form. Indeed, the example of Figure 4 cannot be converted by the factored partial fraction expansion method discussed in this section, and the least-squares method should be used.

VII. Conclusion

In this article, the common approach of converting direct form IIR filters to parallel form has been revisited: the method based on partial fraction expansion. If the order of the numerator M is greater or equal to that of the denominator N, the responses of the individual transfer functions can be significantly larger than the net transfer function leading to reduced dynamic range and increased quantization noise. The use of an alternative parallel form is suggested to avoid the dynamic range problem: in the delayed parallel filter the response of the second-order sections is delayed such that there is no overlap with the parallel FIR path. The parameters of the delayed parallel form can also be computed by the usual partial fraction expansion; the only difference is that the order of the numerator polynomial is reduced by performing polynomial long division over the reversed numerator polynomial.

For high (> 100) filter orders, conversion via partial fraction expansion can be numerically sensitive. Therefore, a simple least-squares procedure is proposed to obtain the delayed parallel form directly. The denominators of the second-order sections are computed by recombining the roots of the original denominator. The coefficients of the parallel FIR part simply equal to the first $M - N + 1$ samples of the original impulse response, and the numerators of the second-order sections are estimated by a least-squares fit such that the resulting impulse response is the closest possible to the impulse response of the original filter. Since we are directly optimizing the error of the impulse response, this guarantees optimal filter performance (the frequency response error will be also minimal due to Parseval's theorem). Indeed, filters in the order of 1000 can be factored with very high accuracy.

Finally, the case in which the original transfer function is available in a pole-zero form or, equivalently, in a series combination of second-order sections has been tackled. In addition to the least-squares fit, an alternative method has been presented that starts with factoring out zeros from the numerator until we reach $M < N$ so that partial fraction expansion can be performed in the pole-zero form. The factored partial fraction expansion method has a very low computational complexity requirement, thus, it is ideal for cases when the conversion must be done on the fly. This comes at a price of a low-order FIR filter in series with the parallel sections.

The goal of this article is to raise awareness about the numerical issues related to the common way of converting filters to the parallel form, with the hope that the reader will find the alternative methods appealing and useful in many practical situations.

Acknowledgment

I am thankful to Prof. László Sujbert, Prof. Julius Smith, and Prof. Vesa Välimäki for their helpful comments. This work was supported the ÚNKP-16-4-III New National Excellence Program of the Hungarian Ministry of Human Capacities.

Supplementary material

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author. The material includes MATLAB functions required for converting IIR filters to parallel form and specific scripts to generate the example figures of the article. The files can also be downloadaed at http://www.mit.bme.hu/∼bank/parconv.

Author

Balázs Bank (bank@mit.bme.hu) is an associate professor in the Department of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest, Hungary. From 2013 to 2016 he was an Associate Editor of IEEE Signal Processing Letters. His research interests include physics-based sound synthesis and filter design for audio applications.

References

[1] A. V. Oppenheim, R. W. Schafer, and J. R. Bruck, *Discrete-Time Signal Processing*. Englewood Cliffs, New Jersey, USA: Prentice-Hall, 1975.

[2] W. Chen, "Performance of cascade and parallel IIR filters," *J. Audio Eng. Soc.*, vol. 44, no. 3, pp. 148–158, 1996.

[3] J. A. Belloch, B. Bank, L. Savioja, A. Gonzalez, and V. Välimäki, "Multi-channel IIR filtering of audio signals using a GPU," in *Proc. IEEE Int. Conf. Acoust. Speech and Signal Process.*, Florence, Italy, May 2014, pp. 6692–6696.

[4] M. Price, S. Holden, and M. Sandler, "Accurate parallel form filter synthesis," *IEE Electronics Letters*, vol. 32, no. 22, pp. 2066–2067, Oct. 1996.

[5] A. Krukowski, I. Kale, and G. D. Cain, "Decomposition of IIR transfer functions into parallel arbitrary-order IIR subfilters," in *Proc. IEEE Nordic Signal Processing Symposium (NORSIG'96)*, Espoo, Finland, Sep. 1996.

[6] J. O. Smith, *Introduction to Digital Filters with Audio Applications*. http://ccrma.stanford.edu/˜jos/filters/, Sep. 2007, online book.

[7] B. Bank and J. O. Smith, "A delayed parallel filter structure with an FIR part having improved numerical properties," in *Proc. 136$^{\mathrm{nd}}$ AES Conv., Preprint No. 9084*, Berlin, Germany, Apr. 2014.

[8] K. Steiglitz and L. E. McBride, "A technique for the indentification of linear systems," *IEEE Trans. Autom. Control*, vol. AC-10, pp. 461–464, Oct. 1965.

[9] B. Bank, "Perceptually motivated audio equalization using fixed-pole parallel second-order filters," *IEEE Signal Process. Lett.*, vol. 15, pp. 477–480, 2008.

[10] V. Välimäki and J. D. Reiss, "All about audio equalization: Solutions and frontiers," *Appl. Sci.*, vol. 6, no. 5, p. 129, May 2016.

[11] G. Ramos and J. J. López, "Filter design method for loudspeaker equalization based on IIR parametric filters," *J. Audio Eng. Soc.*, vol. 54, no. 12, pp. 1162–1178, Dec. 2006.

[12] M. Tyril, J. A. Pedersen, and P. Rubak, "Digital filters for low-frequency equalization," *J. Audio Eng. Soc.*, vol. 29, no. 1–2, pp. 36–43, Jan. 2001.