Charles M. Rader

The Rise and Fall of Recursive Digital Filters

The explosive growth in digital signal processing began in the decade of the 1960s, when researchers discovered how to use recursive digital filters to simulate analog filters. I was one of those researchers. The other (older) digital filters were nonrecursive. The two kinds of filters have different characteristics, which are often complementary. At that time, the advantages of recursive filtering over nonrecursive filtering crowded the latter technology off the technological stage. But by the end of the decade, the pendulum began to swing the other way, and recursive digital filtering was no longer very popular. In what follows, I will recall how I began to use and then design recursive filters. I will also compare their properties and characteristics with those of nonrecursive filters and review the reasons for their declining popularity. Finally, I will argue that many of the supposed problems of recursive filters can be easily overcome.

HOW I BEGAN USING RECURSIVE FILTERS

My work on recursive digital filtering in the early 1960s was prompted by a need to simulate speech bandwidth compression devices. Most of the hardware of a 1960s vocoder consisted of low-pass and bandpass analog filters, and there were dozens of them. Such filters were a problem: each weighed over 1 kg and needed to be separately tuned. We didn't understand how the quality of vocoder synthesized speech depended on the filter characteristics, so we needed to try a variety of filter banks and compare the vocoder performances by listening. Of course, it was not practical to build a large number of different filter banks for such a comparison. Since our group was already using a computer to experiment with different pitch detection methods, I decided to use the same computer to simulate an entire vocoder, including the filters.

I knew, of course, that a filter output was related to its input by a convolution integral and that I could approximate a convolution integral of two continuous time functions by a convolution sum of the sampled filter input and the sampled filter impulse response. Today we would refer to that as nonrecursive filtering, or finite impulse response (FIR) filtering. But the impulse responses of the filters I wanted to simulate were very long, taking hundreds of samples to decay. Simulating each filter would have required several hundred multiplications per sampling instant, and there were several dozen filters in a vocoder. With 1960s computer multiplication speed, it would have taken more than an hour of computer time to process each second of speech.

Then I realized that a real filter didn't use a convolution integral. Instead, it stored the current in each inductor and the voltage across each capacitor, five or six quantities in all. Then it computed their changed values an infinitesimal time later. I could approximate the derivatives by differences and derive a set of equations to update the state of a sampled time system that imitated the

EDITORS' INTRODUCTION

Our guest in this issue is a familiar presence: you have already met Charles Rader (charlesmrader@verizon.net) in the "DSP History" column of the May 2004 issue of *IEEE Signal Processing Magazine*. What you may not know is the reason why he opted at the time for an interview rather than a contributed article. He was writing a manuscript on recursive digital filters (which was planned to be published in this column) but Charlie realized that it would not be ready in time to meet our deadlines. So we worked together and published an interview instead. A few years and many editing stages later, the article that presented arguments in favor of recursive digital filters has grown, has expanded its coverage, and has been updated. It is with pleasure that we include it in this issue.

As for Charlie Rader, you may remember from our 2004 interview that he was planning to retire. Indeed, he has retired in Chatham, Massachusetts, but continues to pursue professional interests. He still keeps an office at Lincoln Lab to work on digital signal processing problems and has been a consultant for a modem testing company. He has developed a substantial interest in genetic engineering in agriculture and sometimes gives talks about this topic. In Chatham, Charlie and his wife Carol live right across from "the best clam beds in the world." He can often be found raking clams (even in winter, when he wears a rubber suit) and always welcomes recipes for clams. He continues to learn Swedish and Chinese and is making slow progress. Overall, he tells us that he is so busy that "I wonder how I ever had any time to go to work."

—Adriana Dumitras and George Moschytz
"DSP History" column editors
adrianad@ieee.org
moschytz@isi.ee.ethz.ch

1053-5888/06/$20.00©2006IEEE

filter. This gave me recursive difference equations, which could be implemented as a computer program. The program would run a hundred times faster than a convolution sum. At that point, I had no way of knowing how closely my approximations would match the real performance of the filters that I needed to simulate. Therefore, I needed to learn how to analyze recursive difference equations.

In college, I had heard a lecture about the z-transform in a context completely unrelated to filtering, but it had involved solving difference equations; so I dug out my notes and tried to relearn z-transform theory. I also explained what I was thinking to Ben Gold, a much more experienced engineer. Ben remembered an article by Hurewicz [1], which showed how a z-transform could describe a sampled data transfer function. We also got a lot of help from Dr. Joe Levin, a colleague at Lincoln Labs. Joe gave us several lectures on z-transforms and provided us with lots of references and with mathematical rigor. That gave us the tools we needed. For the next few years, we performed hundreds of different vocoder simulations to learn how to improve vocoder speech quality. We could go from having a design idea to hearing processed speech in a single day. (Joe was one of the smartest engineers I ever met, and one of the nicest. Tragically, he died in an auto accident soon after we began working together.)

HOW I BEGAN DESIGNING DIGITAL FILTERS

Once we understood the z-transforms, it was rather straightforward to design digital filters. These were what we would today call recursive filters or infinite impulse response (IIR) filters. There was substantial literature on the approximation problem for analog filters, which gave formulas for finding the filters' poles and zeros in the Laplace domain. We would find those poles and zeroes and map them into the z-plane. Analog filter designers needed a second step, figuring out how to realize those poles and zeroes by connecting up resistors, capacitors and inductors. But for digital filters, that second step was unnecessary. We also didn't have to worry about component tolerances since digital arithmetic could be arbitrarily accurate.

Ben and I knew right from the start that digital filters were not just simulations of analog filters. We understood that if a signal could be sampled, then real time waveform processing (which was at that time done by analog circuits) could be performed instead by computation. We knew that this would fundamentally change how signals were processed. For example, analog signals are evanescent and must be processed on the fly, but once a signal is in a computer memory, there are many more ways to process it. It was just a matter of faster and cheaper digital hardware. (We never dreamed though that digital hardware would improve so fast or by so many orders of magnitude.)

If digital filters were to be used without computers, we had to understand the effects of short arithmetic word length [2]. Some previous analyses had found expressions for the worst-case accumulation of roundoff error, but these upper bounds were very pessimistic. Instead, we modeled each roundoff operation by adding white noise to what an ideal filter would have produced. The expressions for expected noise power were very easy to derive. On the other hand, for some kinds of inputs, roundoff noise could be far from uncorrelated. For example, if the input were zero, the filter could reach a steady state with each roundoff error identical to the last, which was the exact opposite of uncorrelated. The roundoff noise's component was amplified by the dc gain of the poles of the filter. That steady state output could take any value in a range of values called the deadband. But once this effect was understood, it was easily avoided.

To describe our work, Ben and I wrote the paper "Digital Filter Design Techniques in the Frequency Domain" [3]. We submitted it to *Proceedings of the IEEE*, but the reviewers recommended that it be rejected. We revised the paper only very slightly and resubmitted. The paper was accepted and published in 1967. In the meantime, the book *System Analysis by Digital Computer* appeared, including a comprehensive chapter [4] on digital filters written by Jim Kaiser. Very little in our paper was really new, but along with Kaiser's book chapter, it brought digital filtering to the attention of a large number of engineers. (In parallel with our work on understanding and designing filters, there were people at Bell Laboratories doing much the same thing. Ben and I considered people at Bell Labs, who were doing similar work with ours at the time, as competition; however, I'm pretty sure that Jim Kaiser, who was one of them, never felt that way about us.)

About that same time, Ben and I decided to write a book about digital signal processing. Nothing was written for months and months, until Al Oppenheim persuaded Ben, Tom Stockham, and me to teach a summer course about digital signal processing at the Massachusetts Institute of Technology. Ben and I said that we would use our proposed book as the class notes. This decision gave us a deadline and got us working on the book. *Digital Processing of Signals* [5] didn't actually get published until 1969, but almost all the writing work was done during the summer of 1967.

Whenever you teach, you learn. I had designed and used elliptic filters before 1967, but for teaching about them, I needed to learn the theory of Jacobian elliptic functions, which was a very elegant piece of mathematics. Filter design uses the elliptic functions in two different ways. The transfer function specification is written as an equiripple elliptic function in a design space. Then its poles and zeroes are mapped from the design space to a realization space using a second elliptic function. The mapping preserves the equiripple response. Once the concept is clear, everything is reduced to formulas and the design technique is analytic and closed form. However, the method can only be used to design recursive filters. [SP]

On the other hand, to the best of my knowledge, nobody so far has found an analytic closed form design technique for nonrecursive filter design. In 1967, there wasn't any way to even design optimum FIR filters. Today they can be designed using iterative successive approximation techniques, such as McClellan-Parks method [7]. In this computational age, the usage of iterative methods is not a serious limitation of FIR filter design.

FIR AND IIR COMPARISONS

If we had listed the general characteristics and properties of recursive and nonrecursive filters in the 1970s, in the 1980s and today, the three lists would not be the same. In the 1970s, the major application of digital filters was frequency selective filtering. Therefore, as Table 1 shows, in the 1970s, the recursive filters offered the most advantages.

Then things began to change. Digital filters began to be used for other purposes besides frequency selectivity, like matched filtering and adaptive filtering. For such applications, closed form optimal IIR design techniques didn't work; therefore, FIRs were a natural choice. In particular, the theory and the design techniques of adaptive filtering were appropriate only for FIR filters. Digital filters also began to be used in applications for which linear phase was very important, since having linear phase preserves the waveshape of the desired output. Again, FIR filters were advantageous. Overall, the following factors contributed to the wider usage of nonrecursive filters: 1) new filtering applications; 2) requirements for linear phase; 3) some optimization methods became available for the first time. The resulting FIRs were still less efficient than IIR filters, but the difference between their efficiency was reduced; 4) techniques to handle very long FIR filters that resulted from given design specifications emerged, such as Stockham's method of high speed convolution [6]; 5) the cost of memory and multiplication operations decreased; and 6) parallel computation became practical. Both FIR and IIR filters offer some opportunities for parallel computation, but for IIR filters the parallelism is limited if a multiplication cannot be completed between samples.

As a result of the above evolution, by the 1980s, the preference had tipped decisively toward FIR filtering. But as Table 1 also shows, some of the features that favored FIR filters during the 1980s have become neutral since the 1990s. What changed for the better is that IIR filters can now be designed to be always stable and to have linear phase. Their implementations also feature many more opportunities for parallelism than before. This means that some of the older problems of IIR filters are now overstated.

OVERCOMING THE PROBLEMS OF RECURSIVE FILTERS

Many digital signal processing practitioners mention the following problems of recursive filters to justify their rare usage nowadays: instability, nonlinear phase, limited parallelism, and unsuitability of IIRs for some applications. However, most of these difficulties can be overcome if we agree to run the filtering iteration in blocks and to accept the negligible error of truncating the block. In what follows, I will explain this idea in more detail.

[TABLE 1] A COMPARISON OF THE NONRECURSIVE (FIR) AND RECURSIVE (IIR) FILTERS OVER THE YEARS. COLORED BOXES INDICATE WHEN ONE TECHNIQUE OFFERS AN ADVANTAGE (IN GREY) OVER THE OTHER (IN ORANGE).

	1970		1980		NOW	
PROPERTY TRANSFER FUNCTION	**FIR ZEROS ONLY**	**IIR POLES AND/ OR ZEROS**	**FIR ZEROS ONLY**	**IIR POLES AND/ OR ZEROS**	**FIR ZEROS ONLY**	**IIR POLES AND/ OR ZEROS**
DESIGN METHODS FOR FREQUENCY SELECTIVITY	SUB-OPTIMAL USING WINDOWS	OPTIMAL ANALYTIC, CLOSED FORM	OPTIMAL USING ITERATIVE METHODS	OPTIMAL ANALYTIC, CLOSED FORM	OPTIMAL USING ITERATIVE METHODS	OPTIMAL ANALYTIC, CLOSED FORM
MULTIPLICATIONS/REGISTERS NEEDED FOR SELECTIVITY	MANY	FEW	MORE	FEWER	MORE	FEWER
CAN BE EXACTLY ALLPASS	NO	YES	NO	YES	NO	YES
UNSTABLE	NEVER	FOR POLES p_i, $\|p_i\| > 1$	NEVER	FOR POLES p_i, $\|p_i\| > 1$	NEVER	NEVER
DEADBAND EXISTS	NO	YES	NO	YES	NO	NO
CAN BE EXACTLY LINEAR PHASE	YES	NO	YES	NO	YES	YES
CAN BE ADAPTIVE			YES	DIFFICULT OR IMPOSSIBLE	YES	DIFFICULT OR IMPOSSIBLE
OPPORTUNITIES FOR PARALLELISM			MANY	SOME	MANY	MANY
HILBERT TRANSFORMER	INEFFICIENT	IMPRACTICAL BECAUSE NOT CAUSAL	INEFFICIENT	IMPRACTICAL BECAUSE NOT CAUSAL	INEFFICIENT	EFFICIENT

Let us recall that a recursive filter can be viewed as an efficient way to achieve an approximation to a desired impulse response. In theory, the impulse response of a recursive filter lasts forever, but in practice, it decays exponentially fast toward zero. We could truncate the impulse response after a few hundred samples, when it has decayed to insignificant values. The result would be an FIR filter, but one whose very long impulse response can be achieved with much less work per sample than would be needed for a general FIR filter of the same length.

Let us partition the input signal into blocks of length M. This length should be much larger than the effective length L of the impulse response. Each block can be filtered using the recursive difference equation, but the output of the block will be L samples longer than the input block. Successive blocks of the output must be then pieced together using the overlap-add idea. The first L samples computed by the filter for each block must be added to the last L samples produced by the filter for the previous block. Other samples are used directly. If we use blocks of length $M = 1{,}000$ and a filter whose effective impulse response length is $L = 100$, then the filter recursion is executed 1,100 times to produce 1,000 output samples, which means that there is a 10% penalty. If the alternative were to use a nonrecursive filter that required more than 10% more arithmetic, the block recursive method would be a clear winner. If the penalty cannot be accepted, then longer blocks can be used.

STABILITY

Once we allow ourselves to use recursive filters in blocks and to piece together the blocks, a new possibility opens up. Recursive filters no longer need to be causal recursive filters. We used to be afraid of a pole outside the unit circle because it meant instability. If we use a filter iteration that runs backward in time, then poles that are outside the unit circle are not unstable. So we can have stable recursive digital filters with poles anywhere except directly on the unit circle.

LINEAR PHASE

The same idea of using recursive filters in blocks allows us to design and implement practical recursive digital filters with linear phase. For example, suppose $H(z)$ is the z-transfer function of any filter. We apply that filter to the data first in one time direction and then again in the opposite time direction. The resulting filter's transfer function is $H(z)H^*(1/z)$, which has real values everywhere on the unit circle. If $H(z)$ has any nontrivial poles inside the unit circle, $H^*(1/z)$ has matching poles outside the unit circle. A real transfer function means one with zero phase. So, it is not true that recursive filters can't have linear phase. It's only causal recursive filters that cannot have linear phase. If we process a signal in blocks, we can easily use noncausal filters—each output block will be longer than the input block at each end, but we can still piece successive output blocks together by overlap-add.

The above idea was presented early in a paper by Powell and Chau [8], but it did not receive much attention. Running a causal filter in cascade with an anticausal filter to obtain zero phase when the input is of finite length is a very old idea. MATLAB users may also recognize it implemented as the FILTFILT operator.

PARALLELISM

With the recursive filtering in blocks, we have another opportunity for parallel computation. Different instantiations of the filtering hardware can be devoted to different blocks. Also, the deadband effect no longer exists.

SUITABILITY FOR APPLICATIONS

Hilbert transformers and equalization are two of the applications that can use recursive filters that perform filtering in blocks. Unfortunately, the adaptive filtering problem has not been solved so far as to make use of recursive filtering.

The Hilbert transformer is a filter whose frequency response is $+j$ for negative frequencies and $-j$ for positive frequencies. We can start with a 90° phase-splitter, which is a pair of stable causal all-pass filters, $H_1(z)$ and $H_2(z)$, whose two transfer functions differ by 90°. If the input signal is played in the forward direction through $H_1(z)$ and the resulting signal is played in the backward direction through $H_2(z)$, the final output is a Hilbert transform approximation. This approximation will substantially outperform that of an optimal FIR design, using probably five to ten times fewer multiplications for the same specification.

In equalization and other applications with arbitrary frequency response specifications, there is no a priori reason to expect that the best filter to compensate a channel will be a nonrecursive filter. Good design methods for nonrecursive filters are available, but a recursive filter might achieve equivalent compensation performance to that of a nonrecursive filter with many fewer multiplications per sample. The downside of recursive filtering would be in the adaptation stage, i.e., the process of discovering what filter to use. It is much harder to adapt recursive filters than it is to adapt nonrecursive filters. But a part of the difficulty is because we want to avoid adapting to an unstable filter, a difficulty that no longer needs to concern us. I consider this an area for further investigation.

Thirty years after its apparent demise, perhaps recursive digital filtering deserves a comeback.

REFERENCES

[1] H.M. James, M.B. Nichols, and R.S. Phillips, "Theory of servomechanisms," in *M.I.T. Radiation Lab. Ser. 25*, 1947, ch. 5, pp. 231–261.

[2] J.F. Kaiser, "Some practical considerations in the realization of linear digital filters," in *Proc. 3rd Allerton Conf. Circuit System Theory*, 1965, pp. 621–633.

[3] B. Gold and C.M. Rader, "Digital filter design techniques in the frequency domain," *Proc. IEEE*, vol. 55, no. 2, pp. 149–171, Feb. 1967.

[4] *System Analysis by Digital Computer*, J.F. Kaiser and F. Kuo, Eds. New York: Wiley, 1966, pp. 218–277.

[5] C.M. Rader and B. Gold, *Digital Processing of Signals*. New York: McGraw Hill, 1969.

[6] T.G. Stockham, Jr., "High speed convolution and correlation," in *Proc. AFIPS*, no. 28, pp. 229–233, 1966.

[7] T.W. Parks and J.H. McClellan, "A program for the design of linear phase finite impulse response digital filters," *IEEE Trans. Audio Electroacoustics*, AU-20, no. 3, pp. 195–199, Aug. 1972.

[8] S.R. Powell and P.M. Chau, "Time reversed filtering in real-time," in *Proc. IEEE Int. Symp. Circuits Systems*, May 1990, vol. 2, pp. 1239–1243.