Digital Filters with Adaptive Length for Real-Time Applications

Nicolas Perrin
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Bonnie Heck Ferri
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250

Abstract
This paper explores the use of imprecise computing to develop digital filters with adaptive length in order to meet real-time constraints. A large-order filter typically provides better filtering but takes longer to compute than a low order filter. An adaptive length filter can provide a resource manager with alternatives that can be used if resources become scarce. This paper discusses the use of standard IIR filters used in controls applications and how these filters can be implemented in prioritized stages. Examples are given to show how to implement a 60 hertz notch filter using imprecise computing.

I Introduction

Real-time digital filtering is used in many applications such as control systems, signal conditioning in real-time measurement systems, and target tracking [Stergiopoulos, 2001]. An example of a common digital filter is a 60 hertz notch filter, which is often used in measurements of electromechanical systems to eliminate standard line voltage prior to feeding back the measurement (or 50 hertz notch filters in parts of the world) [Heck, 2001]. Typically, the larger the order of the filter, the better its performance, but the longer it takes to compute.

The use of resource management schemes in order to meet real-time constraints or to improve processor utilization has become a topic of research in the last few years [Murthy and Manimaran, 2001]. However, it is not typically employed in control systems. This paper proposes a method of doing imprecise computing [Natarajan, 1995; Liu et al, 1994] for digital filters to facilitate resource management. The algorithm for the filter is formulated as an *anytime algorithm*, thus, we call the filters *anytime filters.*

II General Procedure

Consider an IIR filter:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}}{a_0 + a_1 z^{-1} + \ldots + a_N z^{-N}}$$

This can be implemented in direct form using the corresponding difference equation:

$$y[n] = b_0 x[n] + b_1 x[n-1] + \ldots + b_M x[n-M] - a_1 y[n-1] - \ldots - a_N y[n-N]$$

To improve numerical accuracy. this filter can be factored into cascaded first (or second) order filters.

$$H(z) = \frac{\prod_i b_{0i} + b_{1i} z^{-1} + b_{2i} z^{-2}}{\prod_i a_{0i} + a_{1i} z^{-1} + a_{2i} z^{-2}}$$

and implemented such that the output of the first filter becomes the input to the second and so on. Thus, you can consider the whole filter to be broken into stages made up of first (or second) order filters. The difference equation for the i^{th} stage (assuming second-order) would be:

$$y[n] = b_{0i} x[n] + b_{1i} x[n-1] + b_{2i} x[n-2] - a_{1i} y[n-1] - a_{2i} y[n-2]$$

For the first stage, x is the actual measured signal. For the subsequent stages, x is the output of the previous filter stage.

Using imprecise computing to reduce computation time is accomplished by prioritizing the stages so that the latter stages can be skipped if resources become scarce. Each stage will filter the signal a given amount, and the more stages that you use the better the filtering. Setting up the filter stages for this type of computation is done by prioritizing the stages in terms of their desired frequency responses and by putting all the gain in the first stage(s). The latter stages should have approximately unity gain. Different types of filters have different characteristics and their adaptive length counterparts behave differently as well.

Butterworth filters have a magnitude frequency response that is monotonic in the pass band and in the stop band. To match given constraints, this kind of filter usually requires higher order than other approximations such as Chebyshev or Elliptic filters. However, what is helpful in the considered application is that the Butterworth filter can be broken easily into second-order stages that have very similar frequency characteristics. Chebyshev filters have a magnitude frequency response that either has ripple in the pass band and is monotonic in the stop band for Type 1 Chebyshev filters, or has ripple in the stop band and is monotonic in the pass band for Type 2 Chebyshev filters. For this application, we considered a Type 1 Chebyshev filter. Elliptic filters have a magnitude frequency response that has ripples in the stop band and in the pass band, and the order of such a filter is then lower than a Chebyshev filter for the same constraints. However, the example will demonstrate a disadvantage of this kind of filter for our application.

The advantage of imprecise computing lies in the ability to skip some of the stages to give more flexibility to the resource manager. A potential problem arises when going from a low order filter to a larger one since the filters are recursive. If a stage is skipped, the output of the corresponding filter will not be computed and we have to reinitialize it for the next use of the filter. Poor initialization could induce large transients the next time that the filter is used.

To mitigate transients when a filter stage is added back into the operation, a simple initialization is to set the previous values of that stage equal to the previous values of the last stage that was actually computed. Suppose the computation is stopped after the i^{th} stage in one iteration (at time index n-1), then at the next iteration (at time index n) the resource manager allows the filter to complete all the stages. Let y_i denote the output of the i^{th} stage. (Note, y_i is also the input of the $i+1^{th}$ stage.) Initialize the $i+1^{th}$ stage with values

$$y_{i+1}[n-m] = y_i[n-m] \quad for \quad m = 1,2$$

This provides an almost "bumpless transfer" for the output signal. The same initialization could be done for subsequent stages that are added in the same iteration.

$$y_{i+k}[n-m] = y_i[n-m] \quad for \quad m = 1,2$$

where $k > 1$. To be conservative, the resource manager might add stages one at a time. Thus, to go from a 1 stage filter to a 3 stage filter, it may first add the second stage in one iteration and the third stage in the next iteration. However, we have found from simulations that going from a 1 stage directly to a 3 stage filter has negligible difference in the transient when compared to the transition from a 2 stage to a 3 stage filter.

III Application to a Notch Filter

For illustration sake, we choose to consider a 6^{th} order filter to remove the 60 Hz frequency component of a signal. We want to implement this filter in the framework of imprecise computing. Factor the sixth-order filter into three second-order filters to be implemented sequentially in three stages. The resource manager can choose to skip either the last two filter stages or only the last one if there is not enough time available for processing.

We will present different implementations with different types of filters. In each case, the filter will be designed according to specifications shown in the characteristic frequency response of a notch filter given on Figure 1.

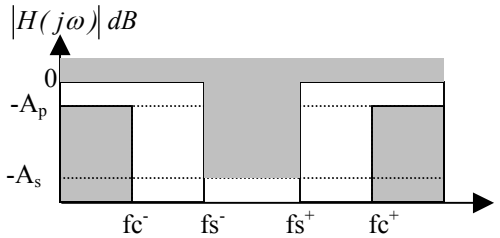

Figure 1: Stop band or notch filter frequency characteristics.

The test signal used to evaluate the filters is:

$$x[n] = \cos(2\pi \frac{30}{F_s} n) + \cos(2\pi \frac{60}{F_s} n) + \cos(2\pi \frac{90}{F_s} n)$$

where F_s is the sampling frequency (chosen here to be 600 Hz). Note that this signal contains one component at 60 Hz in the stop band, one component at 30 Hz in the low frequency part of the pass band and one component at 90 Hz in the high frequency part of the pass band.

The anytime filter is implemented by always computing the first stage of the filter, and then having the choice to disgard the second or third stage, corresponding to having a 2^{nd} , 4^{th} or 6^{th} order filter. The initialization needed when going from a smaller-order filter to a larger one mentioned in the last section is used here.

Elliptic Filter: A 6^{th} order Elliptic filter was synthesised that satisfies the following constraints:

- Loss in the stop band: $A_s = 30$
- Loss in the pass band: $A_p = 1$
- Frequencies of the stop band: $f_{s-} = 50$, $f_{s+} = 70$
- Frequencies of the pass band: $f_{c-} = 40$, $f_{c+} = 80$

The 6^{th} order filter is then divided into three stages, each consisting of a second-order filter with corresponding frequency responses plotted on Figure 2.

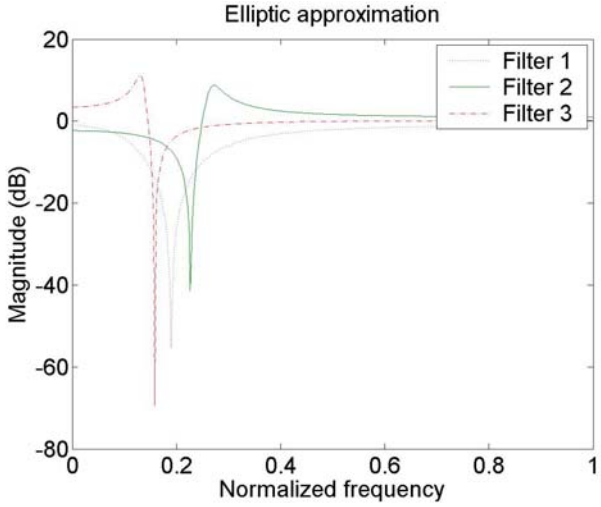

Figure 2: Frequency response of the second-order filters comprising the filter stages (Elliptic design).

Observe that the three second-order filters are all band stop filters with different stop bands. Therefore, if one filter stage is skipped, one band of frequency in the stop band may be filtered very little. This kind of behavior is not desirable in imprecise computing; we would rather filter the entire stop band with every filter stage so that skipping one or more filter stages will result in approximately the same frequency characteristic.

Now consider the Fourier transforms of the input and output signals of two filters, one being the full 6^{th} order filter (shown in Figure 3) and the other being the anytime (shown in Figure 4). Figure 3 shows clearly how the full-order filter is able to

filter out the 60Hz signal with small attenuation of the 30Hz and 90Hz components.

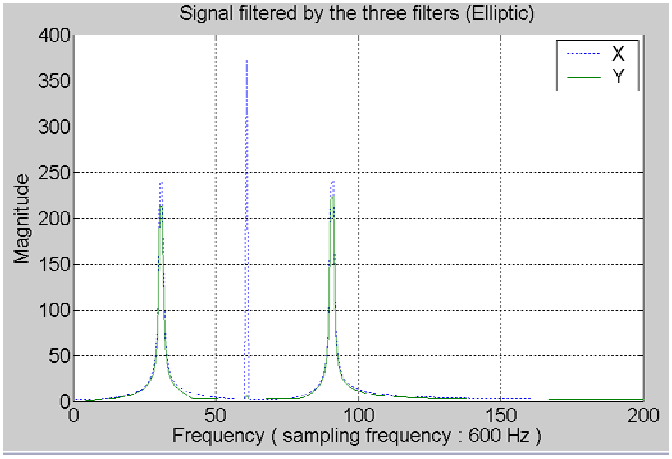

Figure 3: Input (x) and output (y) of the 6th order Elliptic filter.

The results for Figure 4 were generated by running the anytime filter and changing the filter length using the following schedule:

- 0 - 0.25 seconds: 6th order filter
- 0.25 – 0.5 seconds: 4th order filter
- 0.5 – 0.75 seconds: 2nd order filter
- 0.75 -1.0 seconds: 4th order filter
- 1.0 - 1.25 seconds: 6th order filter

The results in Figure 4 show that the anytime filter implemented with imprecise computing gives very similar results when compared to the full-order filter.

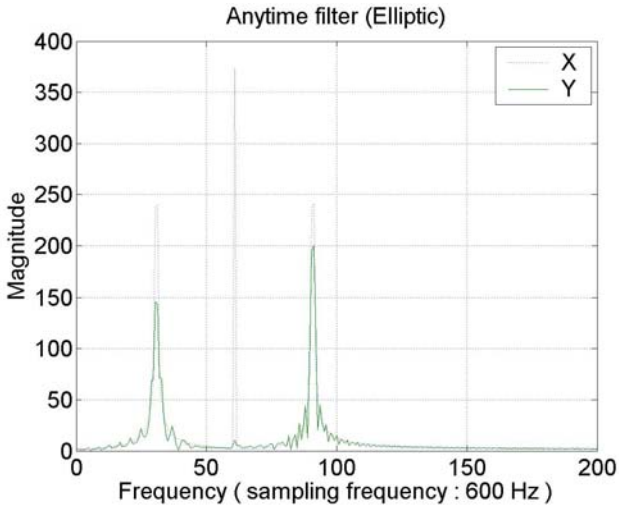

Figure 4: Input (x) and output (y) of the Elliptic filter implemented using imprecise computing.

The Fourier transform of the output is noisy compared to the one obtained by using the full-order filter. Furthermore, the 30 Hz frequency is smaller than it should be and the 90 Hz component is larger.

This behaviour is explained by examining the frequency responses of the three second-order filters in Figure 2. Since each second-order filter is a band stop filter with different stop bands, the ordering of the filters influences the responses of the filter implemented with variable length. In this example, the third filter stage (filter 3 in Figure 2) is the one that has the highest gain at the 30Hz frequency (normalized to 0.1). It is also the filter that is the most likely to be skipped; hence, the 30Hz component is filtered more with this ordering. Filter 1 as defined in Figure 2 should be the first stage since it does the best job on its own of filtering 60 Hz (at 0.2 normalized frequency). Thus, the Elliptic filter is not well-suited to imprecise computing since the second-order filters have very different frequency responses.

Butterworth Filter: Consider, now, the design of a Butterworth filter. The constraints of the filter are now the following:

- Loss in the stop band: $A_s = 20$
- Loss in the pass band: $A_p = 2$
- Frequencies of the stop band: $f_{s-} = 50$, $f_{s+} = 70$
- Frequencies of the pass band: $f_{c-} = 30$, $f_{c+} = 90$

Observe that to keep sixth-order for the filter we have to choose weaker constraints than used for the Elliptic filter.

The sixth-order filter is decomposed into three second-order filters with the magnitudes of the frequency responses plotted in Figure 5.

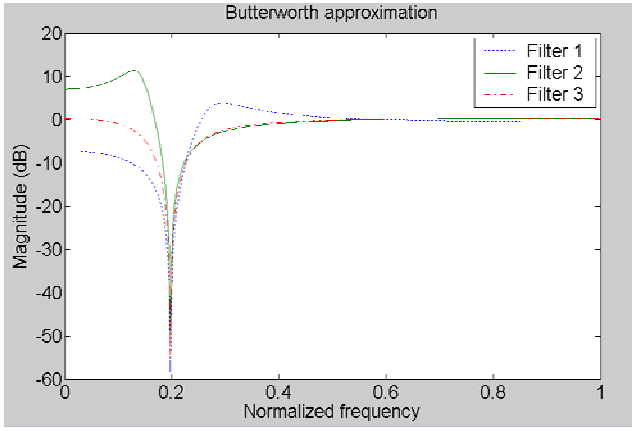

Figure 5: Frequency responses of the second-order filters comprising the filter stages (Butterworth design)

Observe that the magnitude frequency responses of the three filters are quite similar. In particular, they all have the same stop band frequency, unlike the Elliptic filter. So these filters appear to be an iterative refinement of the filtering of the signal, and thus better suited for an imprecise computing application.

Furthermore, notice that we have chosen the filter with the flattest magnitude to be the third stage of the full-order filter. This ordering minimizes the impact of skipping the third filter, which is the stage skipped the most often. In addition, to further minimize the relative importance of the last filter, we have also set the gain of the last filter to be 0 dB in the pass band, and put the main gain on the first filter.

Examine the Fourier transform of the input and output signals when we use either the full-order filter (Figure 6) or the anytime filter implemented with imprecise computing (Figure 7). As in the last example, the size of the anytime filter is determined according to a timed schedule. The 60 Hz component is filtered in both cases. Observe in Figure 7 that the 90 Hz component is now a little amplified, and the 30 Hz component is more attenuated when compared to Figure 6. This result could be predicted from examination of the frequency responses of the second-order filters in Figure 5. Note that reversing the order of Filter 1 and Filter 2 would make the 90 Hz component more attenuated and the 30 Hz component less attenuated. We still observe some noise appearing in the Fourier transform of the output signal of the anytime filter, as we saw for the elliptic filter. However, notice that the overall result of the anytime appears to be very similar to the signal obtained when the full-order filter is used.

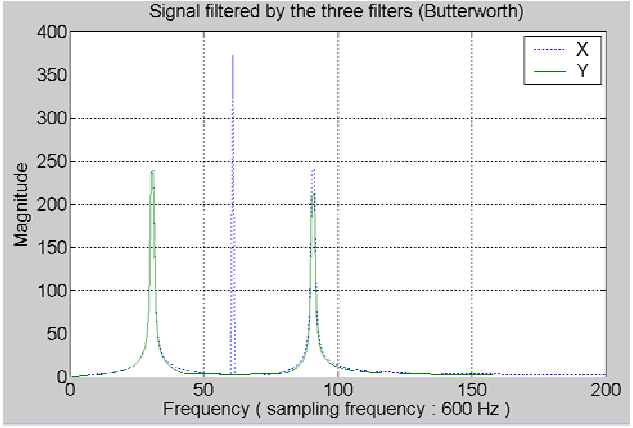

Figure 6: Input and output of the full-order Butterworth filter.

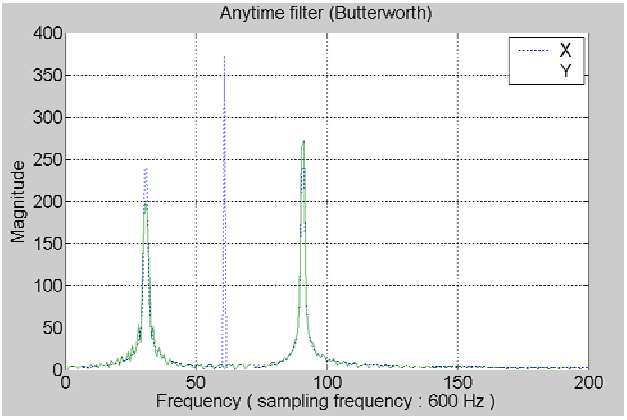

Figure 7: Input (x) and output (y) of the anytime Butterworth filter when imprecise computing is used

It is interesting to examine the effect of the initialization method described in Section II for the case when going from a small-order filter to a larger order one. The time trace of the anytime Butterworth filter following the present schedule (transitions occurring every 0.25 seconds) is shown in Figure 8. Note that the 4th order filter responds similarly to the 6th order filter (some difference in phase accounts for the difference in signal appearance). There is some degradation in the response of the 2nd order filter (between 0.5 to 0.75 seconds). Note also that the transient induced by changing filter size is very small, about 0.02 seconds, when going from a smaller order filter to a larger one. There is negligible difference in the transient response for a transition directly from a 2nd order filter to a 6th order filter.

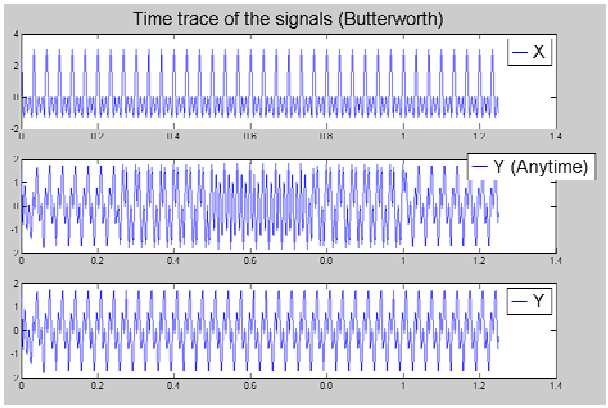

Figure 8: Time trace of signal showing transients.

IV Conclusions

It is shown in this paper that Butterworth filters are better suited for use with imprecise computing than the Elliptic filter (the observation holds with Chebyshev filters as well). Moreover, the ordering of the filters also affects the performance of the anytime filter. An alternative design is to design a second-order filter and replicate it for the other stages. In this way, each filter stage is exactly an iteration of the previous stages. The resulting full-order filter will likely not perform as well as the full-order filter designed using one of the standard methods, but this would likely have better performance if a significant amount of time is spent using the reduced-order filters.

Acknowledgement

This work was funded by NSF grant number CCR-0209179.

References

Advanced Signal Processing Handbook, ed. by S. Stergiopoulos, CRC Press, 2001.

"Digital Signal Processing for Mechatronic Applications," B. Heck and T. Kurfess, in *The Mechatronics Handbook*, ed. by R.H. Bishop, CRC Press, 2002.

C.S.R. Murthy and G. Manimaran, *Resource Management in Real-Time Systems and Networks,* MIT Press, Cambridge, MA, 2001.

Imprecise and Approximate Computation, ed. by S. Natarajan, Kluwer Academic Publishers, 1995.

J. Liu, W-K Shih, K-J Lin, R. Bettati, and J-Y Chung, "Imprecise Computations," *Proceedings of the IEEE*, vol. 82, no. 1, January 1994, pp. 83-93.