

Ultra narrowband filtering with Prism signal processing: design and simulation

Manus Henry^{a,b}

^aDepartment of Engineering Science
University of Oxford
Oxford, OX1 3PJ, UK.
manus.henry@eng.ox.ac.uk

^bSchool of Electrical Engineering and Computer Science
South Ural State University
Chelyabinsk, Russia

Abstract—Prism signal processing is a new FIR filtering technique offering a fully recursive calculation and simple filter design. Its low design and computational cost are particularly suited to the autonomous signal processing requirements for the Internet of Things. This paper describes how arbitrarily narrow bandpass filters may be designed and implemented using a chain of six Prisms. In one simulation, a design with 40 MHz sampling rate, 1 MHz central frequency and 0.1 Hz bandwidth results in an FIR filter of length 771 million samples. This filter can be evaluated in 0.39 μ s per sample on a desktop computer: to achieve this update rate using a conventional non-recursive FIR calculation would require supercomputer resources.

Keywords— *Industrial cyber-physical systems, Recursive FIR filtering, bandpass filtering.*

I. INTRODUCTION

Prism Signal Processing (PSP) [1] is a new FIR technique particularly suited to the requirements of autonomous computing [2] and for intelligent, adaptive components in Cyber-Physical Systems and the Internet of Things (IoT). The Prism (Fig. 1) is an FIR filter generating one or two outputs. Its properties are defined by its characteristic frequency m and harmonic number h [1]. It offers a unique combination of desirable properties: the calculation is FIR, and hence robust; the outputs have linear phase delay; the calculation is fully recursive so that the computational cost per sample is low and fixed, irrespective of the length of the filter; and its design is straightforward, given desired values of m and h , requiring only the evaluation of linearly spaced sine and cosine values.

This design simplicity enables new Prism networks to be created in real time in response to changes in signal processing

requirements. Ref [3] outlines a fault detection scheme in which the detection of an unwanted frequency component in a signal results in the construction of a new Prism network to filter out that component. Refs [4] and [5] describe a condition monitoring scheme for rotating machinery, with two stages. The first stage monitors rotation startup. Once the steady rotation speed is established, in the second stage a set of Prism networks, including filtering elements, are instantiated to simultaneously track six harmonics of the rotation frequency.

The computational efficiency of PSP is illustrated in [6], which describes a fuel injection monitoring application where flow measurements are generated at 48 kHz using modest computation hardware. The calculation, which includes the removal of unwanted signal components, enables the tracking of fuel pulses as short as 1 ms.

Here we present a new example of how PSP can provide flexible functionality in autonomous systems – through the design and instantiation of bandpass filters. As discussed in [1], conventional filter design is resource intensive, and may be a significant impediment to autonomous signal processing in the field. We outline a simple design procedure whereby a chain of six Prisms can be used to implement a bandpass filter with arbitrary central frequency and bandwidth. As will be shown, the procedure can be used to design and implement even ultra-narrowband filters. The same PSP characteristics apply to bandpass filtering: the design is simple even for long filters, while the computational cost is low and independent of the filter length. The only constraint on filter design is available memory. The new technique makes possible the design and instantiation of new bandpass filters in real time on autonomous devices with modest computing resources.

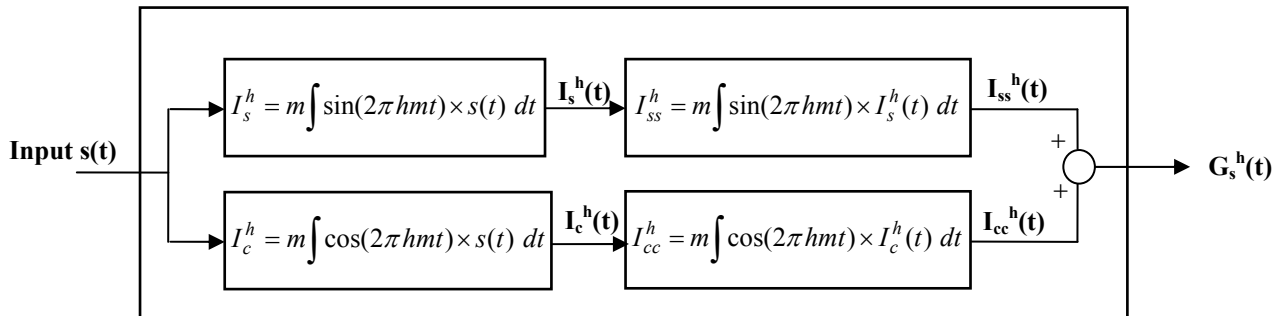


Fig. 1. Structure of Prism to generate a single output, G_s^h .

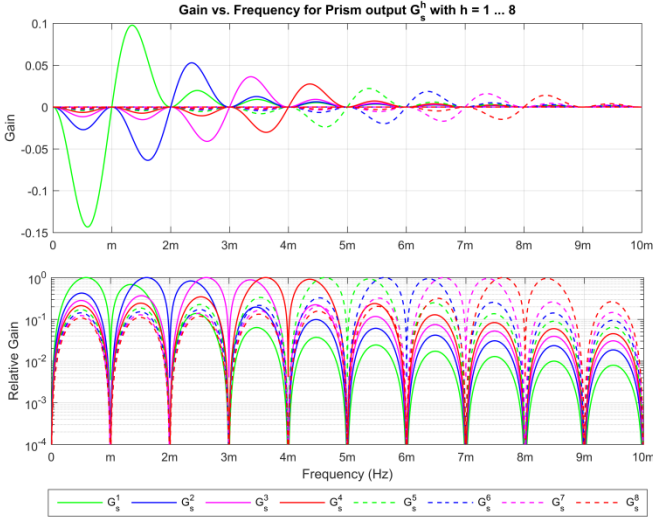


Fig. 2. Gains of Prism output G_s^h for $h = 1, \dots, 8$.

Section II outlines the theory of Prism bandpass filter design. Section III gives examples of Prism-based bandpass filters, including a 770 million tap filter which would require supercomputing resources to evaluate using conventional FIR filtering but which can be evaluated on a desktop computer. Section IV combines a bandpass filter with a Prism-based tracker to demonstrate successful tracking of a signal of known frequency with a signal-to-noise ratio of -60dB.

II. BANDPASS FILTER DESIGN

The bandpass filter design uses a sequence of Prisms where each generates only one output, G_s^h (Fig. 1). Given m , the characteristic frequency of a Prism, and h its harmonic number, the gain of G_s^h at frequency f is given by ([1])

$$\text{gain}(f, m, h) = \text{sinc}^2(r) \frac{r^2}{r^2 - h^2}, \text{ where } r = f/m \quad (1)$$

Fig. 2 shows how the gain of G_s^h varies with frequency (relative to m) and h . In brief, the peak negative value occurs just below hm Hz and the peak positive value occurs just about this value. The first step towards constructing a bandpass filter is illustrated in Fig. 3, which shows a pair of Prisms having a common value of h but with different values of m , selected so that peak negative and positive value of G_s^h align.

Fig. 4 shows the outputs of three such Prism pairs. Three h values are used: 250, 333, and 500 respectively. Higher h values result in a narrower high gain region, which in each case here occurs within ± 0.5 Hz of the centre frequency.

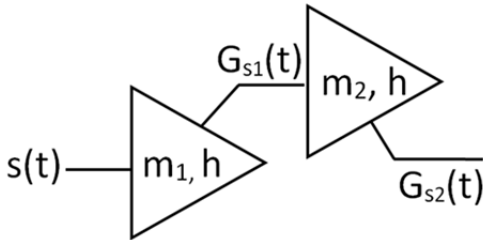


Fig. 3. A pair of Prisms in sequence, forming one stage of a bandpass filter.

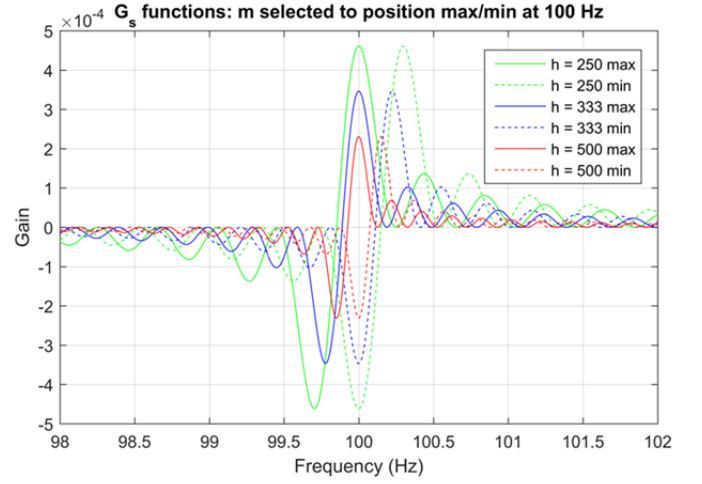


Fig. 4. G_s^h functions designed for maxima or minima occurring at 100 Hz.

For each h , two curves are plotted, where the m_i values are selected so that either the maximum or the minimum value of the corresponding frequency response occurs at 100 Hz.

The main obstacle to be overcome in the creation of efficient Prism-based bandpass filters is the presence of large side-lobes. Here a lobe is defined as any region between notches (i.e. zero values) in the frequency response. The ideal bandpass filter would comprise simply of a single lobe around the central frequency. However, each G_s^h function has two relatively high gain lobes (one positive and one negative) centred on hm Hz, as well as many other lower gain lobes. The arrangement in Fig. 4 both illustrates this problem and suggests a solution. All the curves with a positive peak at 100 Hz have another peak of similar magnitude at a lower frequency. All the curves with a negative peak at 100 Hz have another peak of similar magnitude at a higher frequency. However, a combination of each pair (using the concatenation of two Prisms shown in Fig. 3) may be used to create more symmetric bandpass filters, each with a dominant central lobe.

The result of combining each pair of frequency response curves is shown in Fig. 5, where the y axis has been changed to show the gain in decibels relative to the peak value (ignoring the sign of the gain). Each Prism pair filter has a

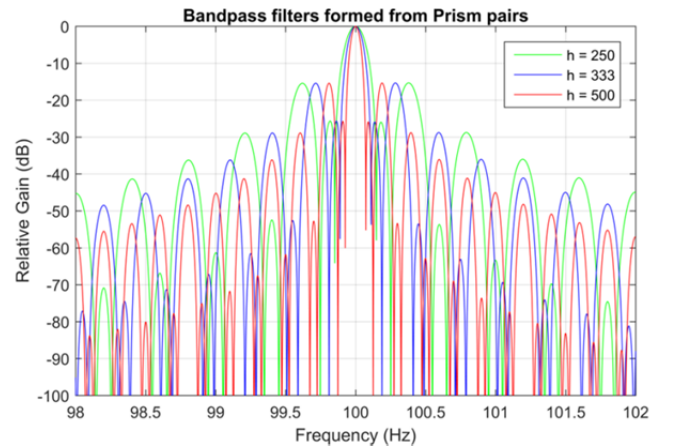


Fig. 5. Bandpass filter responses, where each bandpass filter is created by combining the two frequency responses for each value of h in Fig. 4.

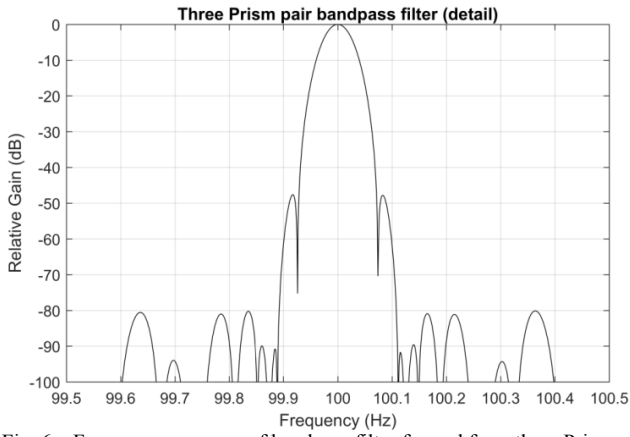


Fig. 6. Frequency response of bandpass filter formed from three Prism pairs.

peak at the desired central frequency and has a reasonably symmetric and declining gain with distance from the peak.

Fig. 6 shows the results of combining all 3 filters together, through a concatenation of six Prisms in series, and where the resulting gain is the product of the gains of each individual Prism. The resulting frequency response shows a rapid decline in gain with distance from the central frequency. The selected ratio of the three h values results in effective cancellation of side lobes. No frequency outside the range $100 \text{ Hz} \pm 0.1 \text{ Hz}$ has a relative gain in excess of -70 dB .

This filter structure, consisting of three pairs of Prisms, with the given ratios between the three h values, is the basic template used for the remainder of the paper. Alternative filter structures, with different numbers of Prisms (singles or pairs), or different ratios between h values, could be used to achieve different bandpass performances, with corresponding trade-offs in terms of the computational resources required.

This filter structure is now generalized as follows. We assume the desired filter has central frequency c , and bandwidth (defined below) b . Simple formulae are used to provide the corresponding values of m and h for each of the six Prisms in order to match this performance requirement. Once the values of m and h are determined, the subsequent design of each individual Prism is readily achieved.

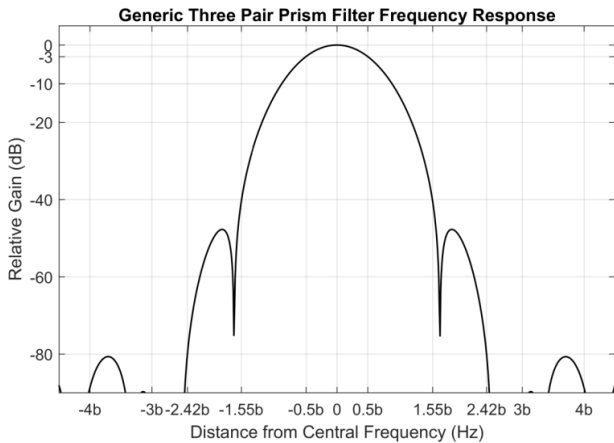


Fig. 7. Frequency response of bandpass filter formed from three Prism pairs.

Fig. 7 shows the frequency response of the generalized three pair Prism filter with arbitrary central frequency c and bandwidth b . Here the bandwidth is defined such that the relative gain in the region $[-b/2, +b/2]$ remains within the range $-3 \dots 0 \text{ dB}$. In other words, the region of length $b \text{ Hz}$, centred on $c \text{ Hz}$, has a relative gain always in excess of -3 dB . Given this definition of b , other characteristics of the filter are readily defined, as follows:

- The gain drops to -40 dB at $\pm 1.55 b \text{ Hz}$
- The gain drops to -80 dB at $\pm 2.42 b \text{ Hz}$

These are the specific characteristics of this filter template. Other filter designs, for example using more Prisms, may result in more rapid drops in gain at the expense of additional computational load and/or higher memory requirements.

Given desired values of c and b , the corresponding values of m and h for each Prism are calculated as follows. Firstly, the three values of h are calculated, in the approximate ratio $6:4:3$, (i.e. $500:333:250$) while allowing that only integer values are permitted. The desired values are found using:

$$h_vals = \text{round}(0.0371 \times c/b \times [6, 4, 3]) \quad (2)$$

where round is the integer rounding function and 0.0371 is a scaling constant to achieve -3 dB gain at $\pm b \text{ Hz}$. For example, if $c = 1000 \text{ Hz}$ and $b = 1 \text{ Hz}$, the desired h values are $223, 148$ and 111 .

For sufficiently large h (say $h > 20$), it is observed that the peak positive and negative values of the corresponding frequency response function G_s^h occur at approximately $m \times (h + 0.371)$ and $m \times (h - 0.371)$ respectively. Accordingly, to make these peak values correspond to the central frequency c , the two values of m for each value of h are given by:

$$\begin{aligned} m_1 &= c / (h + 0.371) \\ m_2 &= c / (h - 0.371) \end{aligned} \quad (3)$$

Using equations (2) and (3), the values of h and m are calculated for each of the six Prisms in the filter. Continuing the example of $c = 1000 \text{ Hz}$ and $b = 1 \text{ Hz}$, the corresponding desired values of m are as follows.

- For $h = 223$, $m_1 = 4.4768 \text{ Hz}$, $m_2 = 4.4917 \text{ Hz}$.
- For $h = 148$, $m_1 = 6.7398 \text{ Hz}$, $m_2 = 6.7737 \text{ Hz}$.
- For $h = 111$, $m_1 = 8.9789 \text{ Hz}$, $m_2 = 9.0392 \text{ Hz}$.

In practice, the values of m are constrained by the requirement that each Prism integral length must be a whole number of samples; this may lead to small variations in the realised values of m with corresponding minor variations in the actual frequency response of the filter.

Continuing the same example, assuming a sample rate f_s of 1 MHz , then the total length of the filter is $1,928,000$ samples. As an approximate rule, the filter length is $1.93 \times f_s/b$. Fig. 8 compares the theoretical response of the filter with its numerical performance when filtering white noise, based on the power spectrum obtained using 100 seconds of data. Here the filter has been implemented using a single threaded C++ function and executed on a desktop PC.

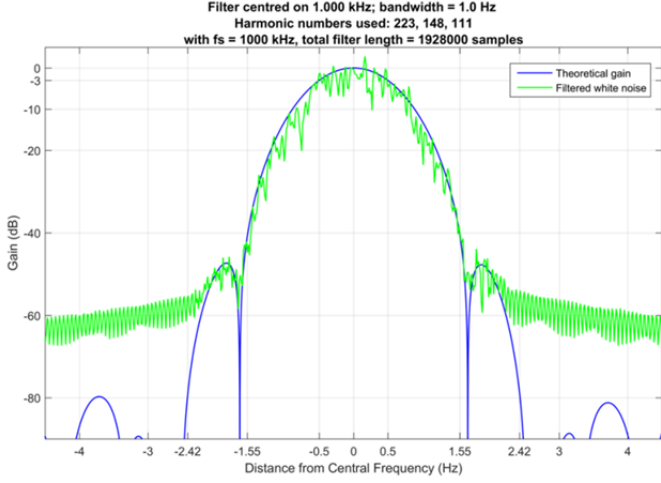


Fig. 8. Theoretical and numerical performance of bandpass filter.

III. BANDPASS DESIGN EXAMPLES

We now present a series of examples that demonstrate the generality of the design procedure given in the preceding section. For the first three examples the sample rate f_s is constant at 40 MHz and the central frequency c is 1 MHz, but the bandwidth is reduced from 10 Hz (Fig. 9) to 1 Hz (Fig. 10) to 0.1 Hz (Fig. 11). This sequence illustrates the relationship between the bandwidth b and h , and the resulting filter length.

In Fig. 9, the highest value of h is 22,260 and the total filter length is 7,716,800 samples. The numerical implementation of the filter shows a good match with the theoretical frequency response. In Figs. 10 and 11, as the filter bandwidth is reduced by an order of magnitude each time, the corresponding values of h and the total filter length in samples both increase by approximately tenfold, so that in Fig. 11 the total filter length is 771,677,600 samples. As the passband frequency shortens, the frequency discretization of the FFT used to analyze the numerical results becomes more apparent. Examples in the next section provide alternative means of verifying that the filters are performing as expected.

Fig. 12 shows another filter with the same f_s but a lower central frequency, with $c = 100$ Hz and $b = 1$ Hz. The resulting performance is broadly similar to that shown in Fig. 10, demonstrating independence from the selection of c .

The memory required to implement a Prism-based bandpass filter is proportional to the filter length. However, the compute time per sample is constant for a particular processor. The filters of Figs. 8–12 all require 0.39 μ s per sample, using a single thread on an Intel Xeon E5-2630 processor, running at 2.3 GHz, and with 32 Gb RAM. The evaluation of a conventional, non-recursive, 771 million sample FIR filter in 0.39 μ s would require a computational load (based on $2 \times$ filter length/computer time per sample) of 3.9 Petaflop/s, equivalent to the 28th fastest supercomputer [7], which uses 126,468 Xenon E5-2695 chips running at 2.1 GHz. This illustrates the computational efficiency provided by Prism signal processing.

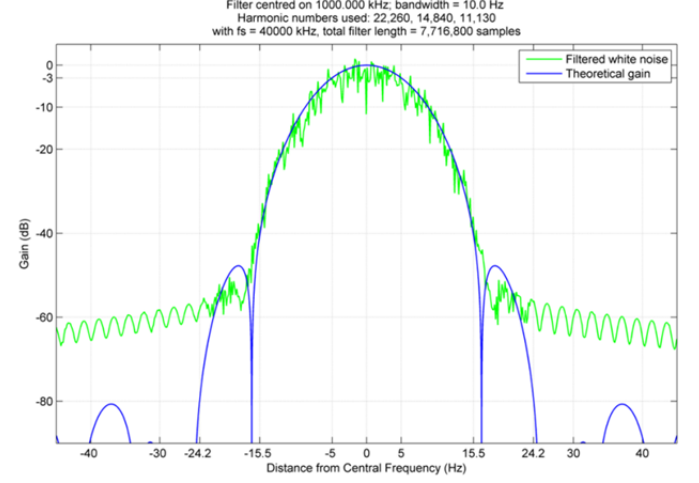


Fig. 9. Theoretical and numerical gain: $f_s = 40\text{MHz}$, $c = 1\text{MHz}$, $b = 10\text{Hz}$

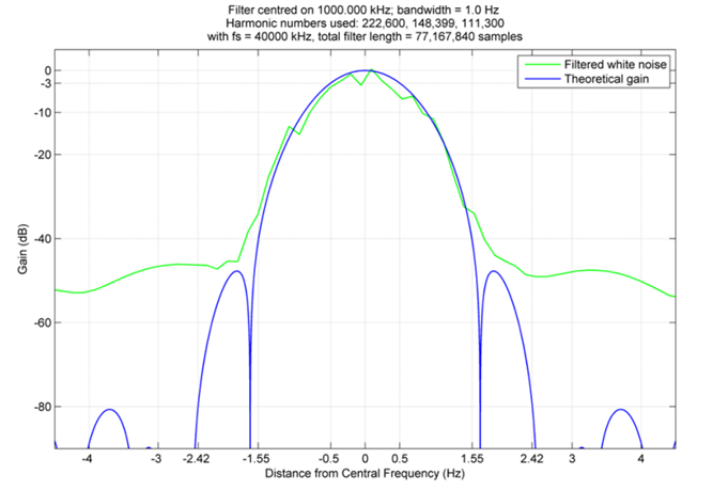


Fig. 10. Theoretical and numerical gain: $f_s = 40\text{MHz}$, $c = 1\text{MHz}$, $b = 1\text{Hz}$

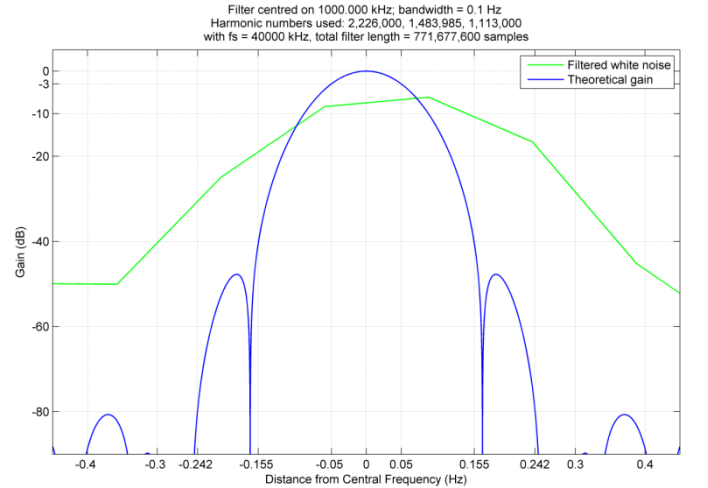


Fig. 11. Theoretical and numerical gain: $f_s = 40\text{MHz}$, $c = 1\text{MHz}$, $b = 0.1\text{Hz}$

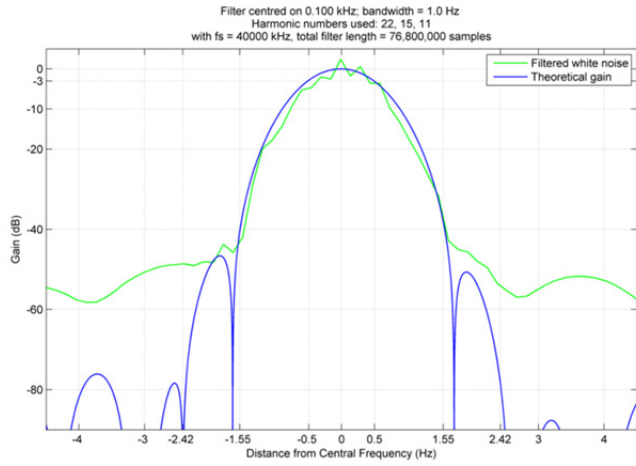


Fig. 12. Theoretical and numerical gain: $f_s = 40\text{MHz}$, $c = 100\text{ Hz}$, $b = 1\text{ Hz}$

IV. TRACKING LOW SNR SIGNALS VIA PRISM FILTERING

Bandpass filters may be used to provide effective pre-filtering for a Prism-based tracker, for example a Recursive Signal Tracker (RST) ([6]), which calculates frequency, amplitude and/or phase estimates of an input sinusoid. Such bandpass pre-filtering can facilitate the tracking of signal components which otherwise would not be possible, due to high levels of noise and/or the presence of other frequency components. Fig. 13 illustrates the basic scenario – a three stage bandpass filter passes its output to an RST which then tracks the sinusoidal parameters of the filter output.

Figs. 14-19 show the tracking performance achieved in the following simulation. A sinusoid with frequency 100 kHz and amplitude 1 mV is mixed with high amplitude white noise so that the signal-to-noise ratio is -60 dB. The resulting noisy signal is sampled at 20 MHz, filtered and tracked. As the input signal frequency is already known to high precision, the bandpass filter is designed using $c = 100\text{ kHz}$ and $b = 0.1\text{ Hz}$.

Fig. 14 shows the power spectra of the raw and filter signals respectively. Given the high level of noise, the signal at 100 kHz is not visible in the power spectrum. However, the filtered signal has a strong peak at 100 kHz.

Figs. 15 – 18 show the tracked frequency, amplitude, phase and the recovered signal (constructed using the tracked amplitude and phase) respectively. In each case, the tracked parameter and its true value are shown in the upper plot, while the residual error is shown in the lower plot. In Fig. 15, the true frequency is constant at 100 kHz; this value is tracked with a root mean square error of approximately $1\text{e-}3\text{ Hz}$.

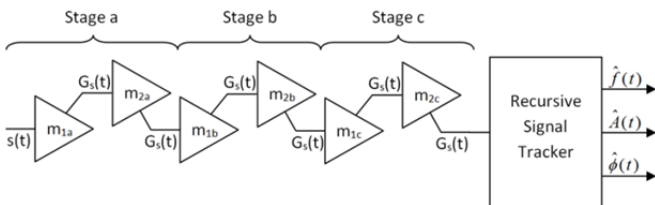


Fig. 13. Bandpass filter and tracker

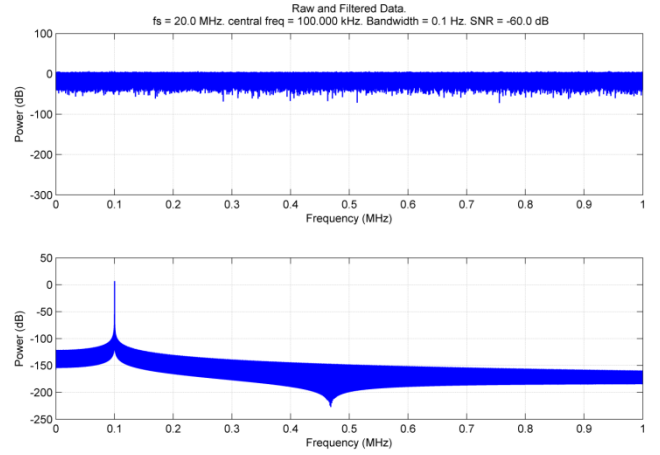


Fig. 14. Filter and track example: (above) power spectrum of unfiltered data, with 100 kHz component not visible above noise floor; (below) filtered signal with 100 kHz component visible.

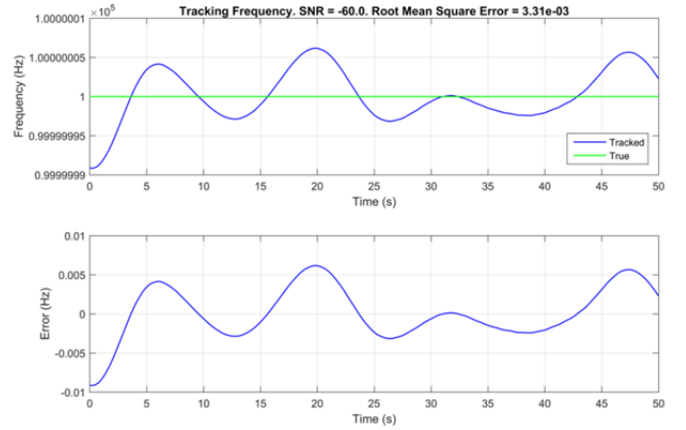


Fig. 15. Filter and track example: frequency tracking performance.

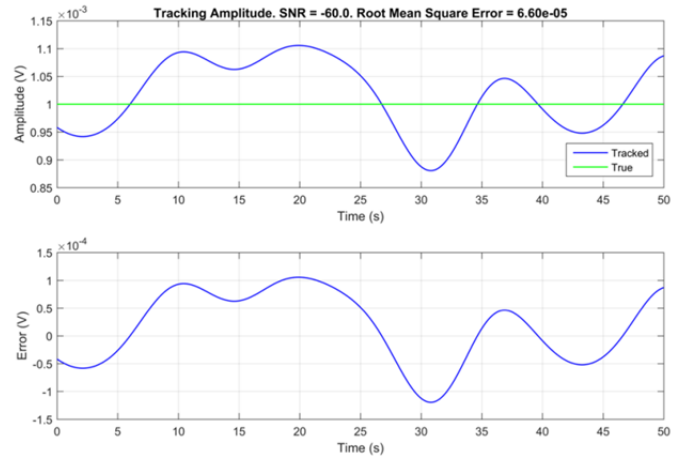


Fig. 16. Filter and track example: amplitude tracking performance.

In Fig. 16, the true amplitude is constant at 1 mV; this value is tracked with a root mean square error of 0.066 mV.

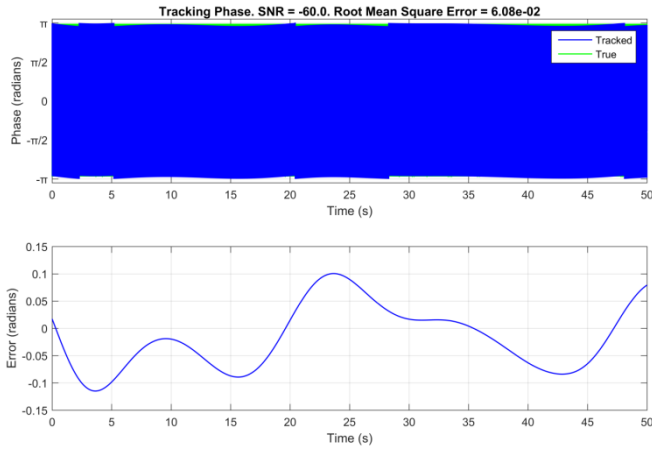


Fig. 17. Filter and track example: phase tracking performance.

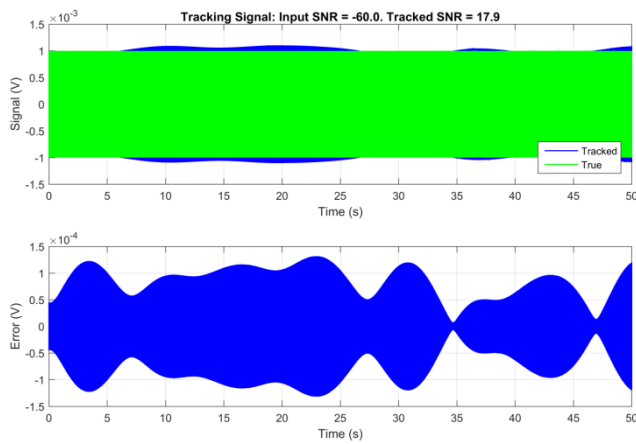


Fig. 18. Filter and track example: signal tracking performance.

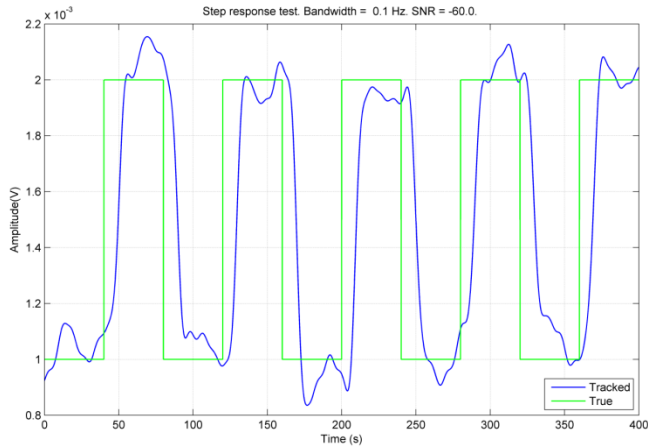


Fig. 19. Tracking periodic step changes in amplitude.

In Fig. 17, the true phase varies between $\pm \pi$ radians; this value is tracked with a root mean square error of 0.06 radians. In Fig. 18, the recovered signal (calculated from the tracked amplitude and phase) is compared with the original. The root mean square error is approximately 0.1 mV, resulting in an

overall change in SNR from -60 dB to +17.9 dB as a result of the filter and track procedure. The amplitude and phase parameters of the tracked signal are compensated for the pre-filtering to enable the reconstruction of the original signal.

Fig. 19 shows a second filter and track example, demonstrating the dynamic response of the system. The sample rate is 20 MHz, the central frequency c is 250 kHz, the bandwidth b is 0.1 Hz, and the signal to noise ratio is -60 dB. The true signal amplitude undergoes a series of step changes between 1 mV and 2 mV: this pattern is essentially recovered by the filter and track mechanism.

The filtering and tracking scheme is straightforward to design and implement and can deliver useful results even with low SNR inputs. The only requirement is knowledge of the frequency range (i.e. the values of c and b) to be monitored, which may be determined at a local level in real time. These techniques offer a powerful toolset for autonomous devices to adapt signal processing schemes to match changing local conditions and evolving goals.

V. SUMMARY

This paper has outlined a scheme for constructing Prism-based bandpass filters using simple design rules, where the only requirement is knowledge of the desired central frequency and bandwidth. The resulting filters share the computational efficiency of individual Prisms. Combining a bandpass filter with a Prism-based tracker facilitates the tracking of a signal component even with low SNR.

A companion paper to this conference describes a bandpass filtering demonstrator, which delivers the equivalent DSP performance of up to 384 TMAC/s, using an Artix-7 FPGA which has a nominal rating of approximately 1 TMAC/s.

The examples in both papers illustrate the suitability of the design scheme to create long filters, which would require supercomputing resources if calculated using the conventional non-recursive FIR convolution. However, the technique is equally well-suited to shorter filter designs.

VI. REFERENCES

- [1] MP Henry, F Leach, M Davy, O Bushuev, MS Tombs, FB Zhou, and S Karout, "The Prism: Efficient Signal Processing for the Internet of Things", IEEE Industrial Electronics Magazine, pp 2–10, December 2017. DOI: 10.1109/MIE.2017.2760108.
- [2] JO Kephart and DM Chess, "The vision of autonomic computing", IEEE Computer, vol. 36, no. 1, pp. 41–50, Jan. 2003.
- [3] MP Henry, "An Introduction to Prism Signal Processing applied to Sensor Validation", Measurement Techniques, pp 1233 – 1237, Mar 2018. DOI: 10.1007/s11018-018-1345-1
- [4] MP Henry, V Sinitsin, "Prism Signal Processing for Machine Condition Monitoring I: Design and Simulation", 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS-2018). May 2018.
- [5] MP Henry, V Sinitsin, "Prism Signal Processing for Machine Condition Monitoring II: Experimental Data and Fault Detection", 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS-2018). May 2018.
- [6] F. Leach, S. Karout, F.B. Zhou, M.S. Tombs, M. Davy, and M.P. Henry, "Fast Coriolis mass flow metering for monitoring diesel fuel injection", Flow Measurement and Instrumentation, 58 (2017), pp 1–5.
- [7] Top500 List. <https://www.top500.org/list/2017/11/?page=1> accessed April 11, 2018