The Prism: recursive FIR signal processing for instrumentation applications

Manus P. Henry, *Member, IEEE*

***Abstract*— This paper provides the mathematical background for the Prism (precise, repeat integral signal monitor), a signal processing network node used in a variety of sensing and instrumentation applications. The key operation is a double Fourier-style integration, which can be implemented recursively using sliding windows and precisely using Romberg Integration. The Prism generates one or two outputs; if two are generated they are orthogonal, analogous to an analytic signal, from which sinusoid properties such as frequency, phase and amplitude can readily be derived. The Prism outputs are finite impulse response (FIR) but the calculation is recursive, resulting in a low computational cost which is independent of filter length. The paper compares the Prism's computational efficiency with both a least squares FIR filter design and an equivalent Prism filter implemented as a conventional convolution. The advantages of the Prism include design simplicity, low computational cost, and a linear phase response, making it a useful network node for a wide range of instrumentation and signal processing tasks.**

Keywords*— *signal processing, FIR filtering, recursive FIR filtering, Prism, Romberg Integration.

I. INTRODUCTION

This paper provides the mathematical background to the Prism (precise, repeat integral signal monitor), which is used as a signal processing network node in a variety of applications reported in the recent literature. The key mathematical operation is a Fourier-style double integration, which can be implemented recursively using sliding windows and precisely using Romberg Integration. The Prism generates one or two outputs; if two are generated they are orthogonal, analogous to an analytic signal, from which sinusoid properties such as frequency, phase and amplitude can readily be derived. The Prism calculation is finite impulse response (FIR); its benefits include design simplicity, a linear phase response, and a low computational cost which is independent of window length, a significant advantage over conventional FIR filters. Prisms can be cascaded in serial and/or parallel to create networks for carrying out more complex tasks, such as notch, bandpass and dynamic notch filtering, or the tracking of one or more sinusoidal components within a signal.

A general overview of Prism Signal Processing (PSP) as a potential signal processing networking technology for Internet of Things applications is given in [1]. In [2], PSP is applied to sensor validation, whereby Prism networks can be instantiated and/or extended in real time, in response to changing signal properties and/or diagnostic states. PSP has been applied to pressure sensor validation [3], condition monitoring of rotating machinery [4], [5], and frequency estimation in power systems [6]. In [7], [8], PSP is applied to Coriolis mass flow metering, whereby rapid measurement updates (at 48 kHz) enable for the first time direct mass flow measurement of fuel injection pulses as short as 1 ms in a laboratory diesel engine. Prism networks have also been used to design and implement ultra-narrowband filters [9], [10].

The central claim to novelty and utility for the Prism is that it operates as a fully recursive FIR filter while having an inherently simple design procedure. These properties represent an advance over previous work where a partially recursive calculation is introduced into FIR filters, at the expense of a more complex design procedure.

Previous authors have reported a number of approaches for developing recursive FIR filters, typically entailing specific mathematical structures to support efficient calculation. These approaches include: adapting IIR filters to approximate FIR behaviour [11], [12], for example the switching and resetting of multiple IIR filters [13]; the Cascade Integrator Comb (CIC) [14], [15], which cascades the moving average filter (arguably the only 'naturally' recursive FIR filter); the construction of piecewise polynomial approximations to the impulse response [16] – [19]; complex filtering over finite rings [20] – [22]; cyclotomic resonators [23]; exploiting linear phase symmetry [24]; impulse response rounding [25]; and combining FIR filtering with state-space estimation [26]. In addition, specific techniques have been developed for two-dimensional recursive FIR filtering in image processing applications [27] – [29].

This paper explains the mathematical development behind the Prism. The analysis is given in terms of continuous time, even though the Prism operates on sampled (i.e. discrete time) data. The step from continuous to discrete time is effected without further mathematical development via the use of Romberg Integration (RI) [30]. Numerical simulations reported in this paper demonstrate that results obtained using discrete time numerical calculation match the corresponding continuous time analytical results with close to double precision accuracy. A future paper will explain in detail how RI has been adapted to operate on time series data in the Prism.

Section II provides the mathematical development. Section III calculates the impulse response of the Prism outputs and constructs the corresponding non-recursive convolutional filters. Section IV compares the design and computation costs of the Prism with those of a conventional least squared low-pass FIR filter and with the convolution form of the Prism. Section V considers stability and implementation issues, particularly for low cost, low precision systems. Sections VI and VI provide discussion, conclusions, and future work.

II. MATHEMATICAL DEVELOPMENT

As stated in the introduction, previously reported recursive FIR filtering techniques typically have a particular application and/or mathematical approach that inform their development. This is also the case for the Prism technique: it was developed with a view to instrumentation applications where the intention is to track essentially sinusoidal sensor signals, applying Fourier-style integration techniques.

Consider a noise-free sinusoidal signal taking the form:

$$s(t) = A\sin(2\pi ft + \phi)\,, \tag{1}$$

where t (or τ) is time in seconds, A is the amplitude, f is the frequency in Hertz and ϕ is the phase offset in radians at time $t = 0$. In practice A and/or f may vary with time, but are assumed constant in this initial analysis. The basic mathematical operation used here is to calculate double integrals of the form:

$$I^{h}_{[s|c][s|c]} = \tag{2}$$
$$m^2\int_{-\frac{1}{m}}^{0}[\sin\,|\,\cos](2\pi hmt)\left(\int_{t-\frac{1}{m}}^{t}[\sin\,|\,\cos](2\pi hm\tau)\cdot s(\tau)d\tau\right)dt$$

The subscript notation [s | c] indicates the selection of one alternative between s (sin) and c (cosine). The superscript h is the harmonic number, a positive integer, usually small. m is the characteristic frequency, where, in any digital implementation, its period must be a whole number of sample durations. Usually m is higher than f, the frequency of the input. For clarity, unlike in other Fourier-based techniques, m is fixed at design time while f is usually unknown, although expected to fall within a certain range. For example, in a typical instrumentation application, using a fixed sampling rate in the range 5 – 50 kHz, a sinusoidal signal with a frequency in the range 50 – 180 Hz might be tracked using double integrals with values of m around 200 Hz. The length of each integral is typically not less than around 30 samples; as will be shown, the greater the length in samples, the greater the computational advantage over conventional FIR filtering.

The integral limits are selected so that ϕ, the phase of $s(t)$ at $t = 0$, is also the phase of the most recent data value. Hence, calculating ϕ gives the phase at the end of the data window, rather than (for example) at the mid-point of the double integral. Accordingly, the double integral as a whole extends in time from $-2/m$ to 0. The harmonic number h indicates how many whole cycles of each modulating sinusoid take place within each integration period. For example:

$$I^{1}_{ss} = m^2\int_{-\frac{1}{m}}^{0}\sin(2\pi mt)\left(\int_{t-\frac{1}{m}}^{t}\sin(2\pi m\tau)\cdot s(\tau)\,d\tau\right)dt \tag{3}$$

$$I^{2}_{sc} = m^2\int_{-\frac{1}{m}}^{0}\sin(4\pi mt)\left(\int_{t-\frac{1}{m}}^{t}\cos(4\pi m\tau)\cdot s(\tau)\,d\tau\right)dt\,. \tag{4}$$

Analytically, these integrals are equivalent to the following expressions:

$$I^{1}_{ss} = A\,\mathrm{sinc}^2(r)\frac{-3r^2}{(r^2-1)(r^2-4)}\sin(\phi - 2\pi r) \tag{5}$$

$$I^{2}_{sc} = A\,\mathrm{sinc}^2(r)\frac{2r(r^2+8)}{(r^2-4)(r^2-16)}\cos(\phi - 2\pi r)\,. \tag{6}$$

Here r is the frequency ratio of the input sinusoid, defined as:

$$r = f/m\,, \tag{7}$$

so that $r = 0$ when $f = 0$ Hz and $r = 1$ when $f = m$ Hz. Typically, $r < 1$ for a sinusoid being tracked (it will be shown in future papers that optimal tracking performance is achieved when $r = 0.5$), but values of r exceeding unity often arise in Prism filtering applications. sinc(x) is the normalised sinc function defined as follows:

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}\,. \tag{8}$$

The utility of the double integral approach begins to emerge from these equations. By passing the signal $s(t)$ through different double integrals in parallel, a set of algebraically related functions can be calculated, from which the values of r (and hence f), A and ϕ might be deduced. Most usefully, some integrals in the family result in sine functions, while others produce cosine functions, offering the strategy of computing orthogonal pairs of integral values (comparable to the analytic function) from which to derive the sinusoid parameters. Both sine and cosine terms have a common delay term $2\pi r$. This is the linear phase delay characteristic of an FIR filter where the time delay is $1/m$, i.e. half the double integral length.

Next, we address the issue of evaluating the numerical values of the double integrals in a computationally efficient (i.e. recursive) manner. This results in a simplification of the algebraic form of the equations, and leads to the development of the Prism concept. We further assume integration is to be applied numerically to a time series of data $s(t_k)$ with fixed sample rate fs, and where, with each new sample, new integral values are to be calculated, based on a window of data consisting of the most recent $2fs/m + 1$ samples. Accordingly, fs is included as an additional parameter of the Prism.

On initial consideration, the numerical evaluation of this family of double integrals should be computationally expensive. For example, if equation (3) is evaluated in a conventional manner, the number of multiplications of a sample by a modulating sine value is given by the square of the number of samples in each single integral. For, say 100 sample points, this would require 10,000 multiplications, even before further calculations were performed to accumulate the single and double integral values. Clearly in many applications this would constitute an unacceptably large computational burden.

A common technique for reducing the computational load of calculations over a window of data is to use a recursive approach. Here the algorithm is arranged so that the calculation at time step $k + 1$ is based primarily on results from the calculation at time step k, typically by removing the contribution of the oldest value in the previous data window and including the contribution of the newest value which has been added to form the current data window. Applying this technique to an integration (or indeed a double integration) calculation is potentially straightforward. However, the

difficulty arises in (re)calculating the products of the input signal $s(t_k)$ and the characteristic sine or cosine function values. Equation (2) implies that the initial phase of each modulating function is zero for each integral. This in turn implies recalculating the product of signal and modulation function across the whole integral at each new time step. To apply a recursive, sliding window approach to equation (2) it would be necessary for the modulating functions to 'slide with' the input signal, so that all (but the oldest) of the signal/modulation function products at time step k remain valid at time step $k + 1$.

This requirement can be expressed mathematically by introducing an additional variable, q, representing the initial phase of the modulating functions, which is no longer assumed to be zero. As both integrals have the same length, it can be assumed that the phase offset is the same for each. Thus, the definition of I_{ss}^1 given in equation (3) may be generalised to

$$I_{ss}^1 = m^2 \int_{-\frac{1}{m}}^{0} \sin(2\pi mt + q) \left(\int_{t-\frac{1}{m}}^{t} \sin(2\pi m\tau + q) \cdot s(\tau)\, d\tau \right) dt \quad (9)$$

where, in any practical implementation, q is known and incremented by $2\pi \times mh/fs$ radians (modulo 2π) each time step.

Unfortunately, introducing q into the definitions of the integral family (2) results in analytic expressions (provided for the case $h = 1$ below) which are not particularly tractable. Fortunately, however, the combination or grouping of certain *pairs* of integrals yields simple analytic results which are independent of q. These groups facilitate the use of sliding windows to significantly reduce the computational effort required to numerically evaluate the double integrals.

Given the revised definition of I_{ss}^1 in equation (9), and the corresponding definition of I_{cc}^1:

$$I_{cc}^1 = m^2 \int_{-\frac{1}{m}}^{0} \cos(2\pi mt + q) \left(\int_{t-\frac{1}{m}}^{t} \cos(2\pi m\tau + q) \cdot s(\tau)\, d\tau \right) dt \quad (10)$$

Then the integral group G_s^1 formed from their sum has a simple analytic expression which is independent of q.

$$G_s^1 = I_{ss}^1 + I_{cc}^1 = A\,\mathrm{sinc}^2(r) \frac{r^2}{r^2 - 1} \sin(\phi - 2\pi r) \quad (11)$$

Similarly, defining integrals I_{sc}^1 and I_{cs}^1 as follows:

$$I_{sc}^1 = m^2 \int_{-\frac{1}{m}}^{0} \sin(2\pi mt + q) \left(\int_{t-\frac{1}{m}}^{t} \cos(2\pi m\tau + q) \cdot s(\tau)\, d\tau \right) dt \quad (12)$$

$$I_{cs}^1 = m^2 \int_{-\frac{1}{m}}^{0} \cos(2\pi mt + q) \left(\int_{t-\frac{1}{m}}^{t} \sin(2\pi m\tau + q) \cdot s(\tau)\, d\tau \right) dt \quad (13)$$

then these integrals can be combined into a group G_c^1:

$$G_c^1 = I_{cs}^1 - I_{sc}^1 = A\,\mathrm{sinc}^2(r) \frac{r}{r^2 - 1} \cos(\phi - 2\pi r)\,. \quad (14)$$

So, by accepting the overhead of calculating two double integral pairs, simple analytic expressions are obtained for the combined results, where the numerical values may be calculated recursively using sliding windows. Note also that the pair G_s^1 and G_c^1 nearly form an analytic function – they differ only by a factor r. Similar results are obtained for the integrals for higher harmonic numbers:

$$G_s^h = I_{ss}^h + I_{cc}^h = A\,\mathrm{sinc}^2(r) \frac{r^2}{r^2 - h^2} \sin(\phi - 2\pi r) \quad (15)$$

$$G_c^h = I_{cs}^h - I_{sc}^h = A\,\mathrm{sinc}^2(r) \frac{hr}{r^2 - h^2} \cos(\phi - 2\pi r) \quad (16)$$

These results have been verified directly using the MATLAB Computer Algebra toolbox for all $h \le 500$, but they appear to be valid for all h and have been used successfully in narrowband filtering applications with h as high as 2,000,000 [9], [10]. Note that for each harmonic number the close analogy between the two integral groups and an analytic function is preserved: in general the ratio between G_s^h and G_c^h (excluding the orthogonal sine/cosine functions) is r/h.

These analytic expressions for the integral groups form the basis of the Prism, a signal processing object that receives a time series signal as an input, and which generates, via recursive numerical integration, one or two time series outputs, corresponding to G_s^h and/or G_c^h. It has three primary configuration parameters, m, h and fs. Other parameters determine whether one or both of the outputs are generated, and how the numerical integration is to be executed, as outlined below.

Fig. 1 shows the structure of the Prism, which consists of six single integration blocks, arranged in a cascade of two layers. Each integration block accepts an input time series and generates an output time series, formed from the input signal multiplied by the selected modulation function and then integrated over the last $1/m$ seconds. The integration is implemented efficiently using a recursive sliding window arrangement where the q term (not shown in the diagram equations) is common across all integration blocks and incremented each time step. A factor m is included at each stage to effect the multiplication by m^2 required in equations (12) and (13). The time series are routed between the integration blocks, and the second stage integral outputs are combined to form the Prism output(s) G_s^h and/or G_c^h.

The properties of the Prism outputs, considered as FIR filters, are discussed in [1]. The gain characteristics facilitate the use of Prism networks to instantiate low pass, band pass and notch filters. Examples are provided in [1] – [3], [9] – [10]; more systematic treatments will be given in future publications.

A numerical discrete time simulation has been developed to demonstrate the internal operation of the Prism. The resulting output of each integration block shows good agreement with the corresponding continuous time analytical expression, thus demonstrating the correspondence between (continuous time) theory and (discrete time) numerical operation.

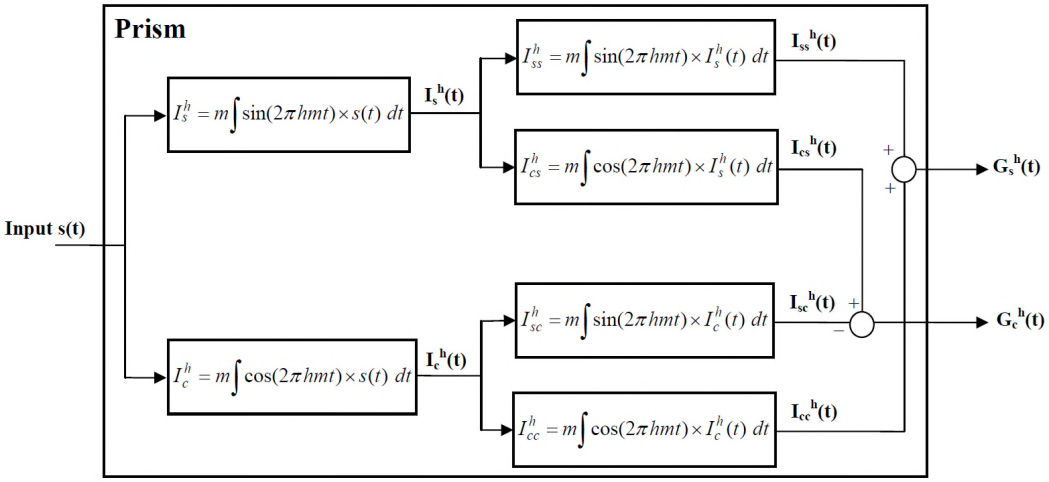

Fig 1: Prism structure.

The Prism input is the pure sinusoidal input given in eqn. (1). Assuming $h = 1$, the superscript notation may be dropped and analytic results for each of the integration blocks outputs can be derived, as follows:

$$I_s = \frac{A}{2\pi(r^2-1)}\times\Big[\big(\cos(2\pi r)-1\big)\big(r\cos(p)\sin(q)-\sin(p)\cos(q)\big) + \sin(2\pi r)\big(r\sin(p)\sin(q)+\cos(p)\cos(q)\big)\Big] \quad (17)$$

$$I_c = \frac{A}{2\pi(r^2-1)}\times\Big[\big(\cos(2\pi r)-1\big)\big(r\cos(p)\cos(q)+\sin(p)\sin(q)\big) + \sin(2\pi r)\big(r\sin(p)\cos(q)-\cos(p)\sin(q)\big)\Big] \quad (18)$$

$$I_{ss} = \frac{A\,\mathrm{sinc}^2(r)}{4(r^2-1)(r^2-4)}\times\big[\ 2\sin(p-2\pi r)\big(r^2-4\big) - \sin(p+2q-2\pi r)(r-1)(r-2) - \sin(p-2q-2\pi r)(r+1)(r+2)\ \big] \quad (19)$$

$$I_{cc} = \frac{A\,\mathrm{sinc}^2(r)}{4(r^2-1)(r^2-4)}\times\big[\ 2\sin(p-2\pi r)\big(r^2-4\big) + \sin(p+2q-2\pi r)(r-1)(r-2) + \sin(p-2q-2\pi r)(r+1)(r+2)\ \big] \quad (20)$$

$$I_{cs} = \frac{A\,\mathrm{sinc}^2(r)}{4r(r^2-1)(r^2-4)}\times\big[\ 2\cos(p-2\pi r)\big(r^2-4\big) - r\cos(p+2q-2\pi r)(r-1)(r-2) + r\cos(p-2q-2\pi r)(r+1)(r+2)\ \big] \quad (21)$$

$$I_{sc} = \frac{A\,\mathrm{sinc}^2(r)}{4r(r^2-1)(r^2-4)}\times\big[\ -2\cos(p-2\pi r)\big(r^2-4\big) - r\cos(p+2q-2\pi r)(r-1)(r-2) + r\cos(p-2q-2\pi r)(r+1)(r+2)\ \big] \quad (22)$$

While all these functions are individually dependent on the modulation phase q, nevertheless it can be observed that, algebraically, all the q terms cancel out when the groups G_s and G_c are formed as defined by equations (11) and (14). For example, equations (19) and (20) give analytic expressions for the terms I_{ss} and I_{cc} respectively. The terms containing q are equal but of opposite sign, so that when I_{ss} and I_{cc} are summed to form G_s the dependency on q is eliminated.

For the simulation, values of fs = 48 kHz and m = 100 Hz are selected. The input signal is a 'pure' (to double precision accuracy) sinusoid with fixed amplitude A = 0.5. In the first example (Figs. 2 – 4), f = 50 Hz, so that r = 0.5. A second example (Fig. 5 only) illustrates the change in behaviour for f = 25 Hz, so that in this case r = 0.25.

The Prism implementation is designed to demonstrate the precise agreement between continuous time analytic results and discrete time numerically calculated values, and so four stages of RI are used. The modulation functions are calculated to double precision. Each product term is also calculated in double precision, and then converted to a 64-bit signed integer, to eliminate the accumulation of rounding error. Integration totalizers are 96-bit fixed point. Given that the true values of all the parameters in equations (17) – (19) are known, it is possible to compare the numerically calculated output values of each Prism integration block with the corresponding analytic values, as they vary with time. Section VIII discusses alternative, lower-cost Prism implementations. Figs. 2 – 4 illustrate the resulting behaviour for r = 0.5. In each case the upper plot shows how Prism integration block outputs vary over time, while the lower plot shows the corresponding error (i.e. the difference between the calculated numerical value and the 'true' analytical value). The magnitude of the numerical error is in all cases less than 3e-14. Fig. 2 shows the behaviour of the first level integrals I_s and I_c. Fig. 3 shows I_{ss}, I_{cc} and their sum G_s. While I_{ss} and I_{cc} have periodic waveforms containing several frequency components, G_s, is a pure sinusoid with frequency 50 Hz. Similarly, in Fig. 4, I_{cs} and I_{sc} have periodic waveforms, while G_c, formed from their difference, is a pure sinusoid orthogonal to G_s. Finally, Fig. 5 is the equivalent of

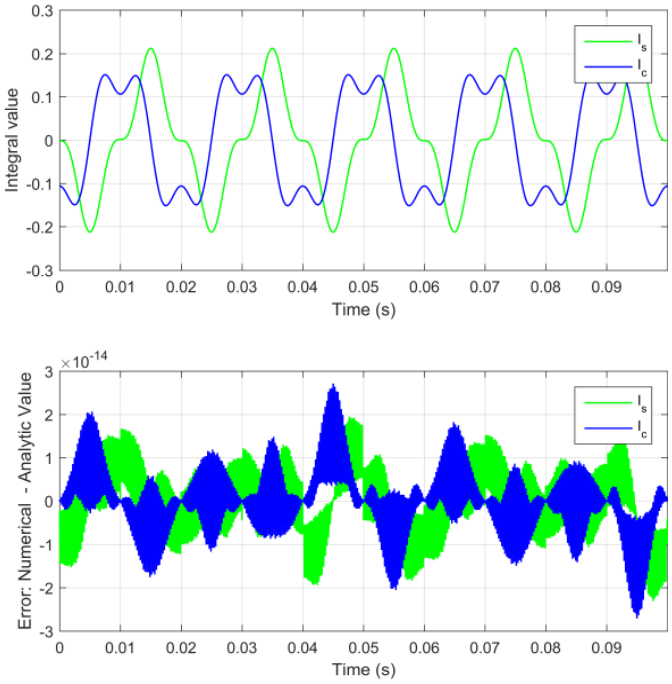

Fig 2: Time series of I_s and I_c outputs (f = 50 Hz, r = 0.5)

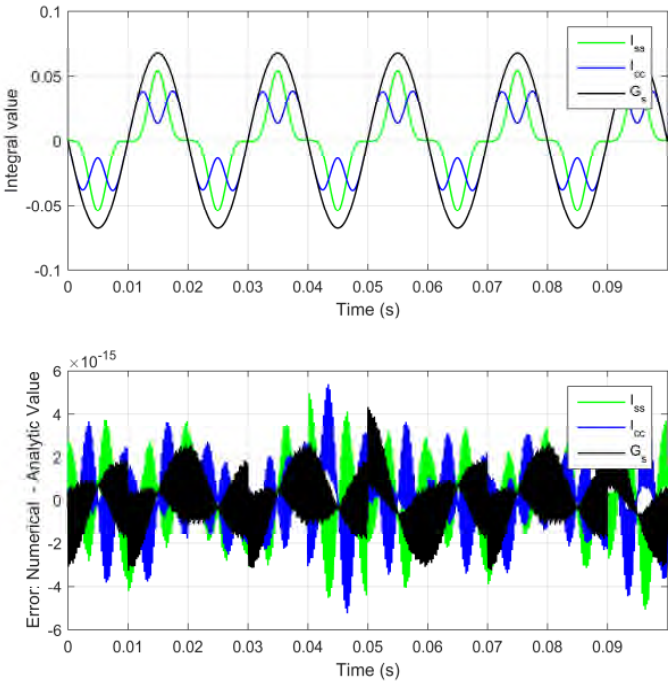

Fig 3: Time series of I_{ss}, I_{cc} and G_s outputs (f = 50 Hz, r = 0.5)

Fig. 4 but using f = 25 Hz and hence r = 0.25. Again, while I_{ss} and I_{cc} contain multiple components, G_s, remains a pure sinusoid with frequency 25 Hz.

Thus, in the absence of noise in the input signal, the numerically integrated values of the Prism block outputs match the corresponding analytic results to high precision when several stages of Romberg Integration are applied.

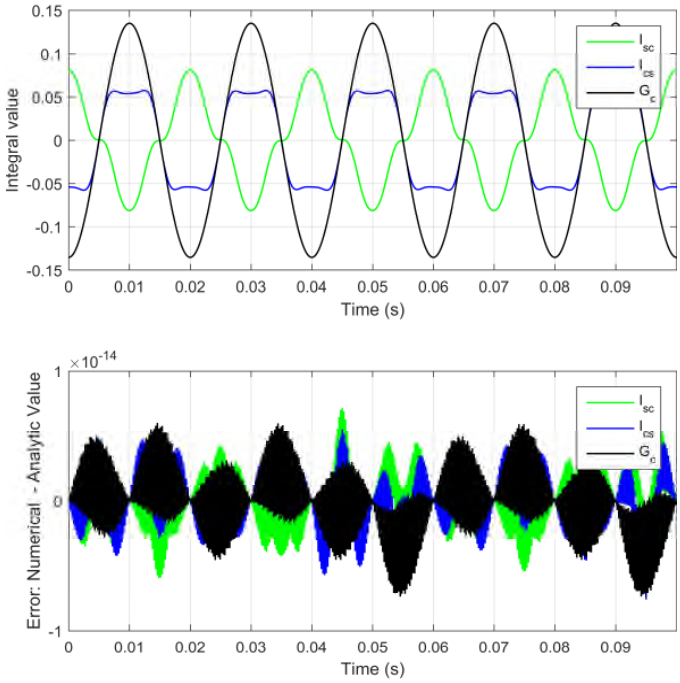

Fig 4: Time series of I_{sc}, I_{cs} and G_c outputs (f = 50 Hz, r = 0.5)

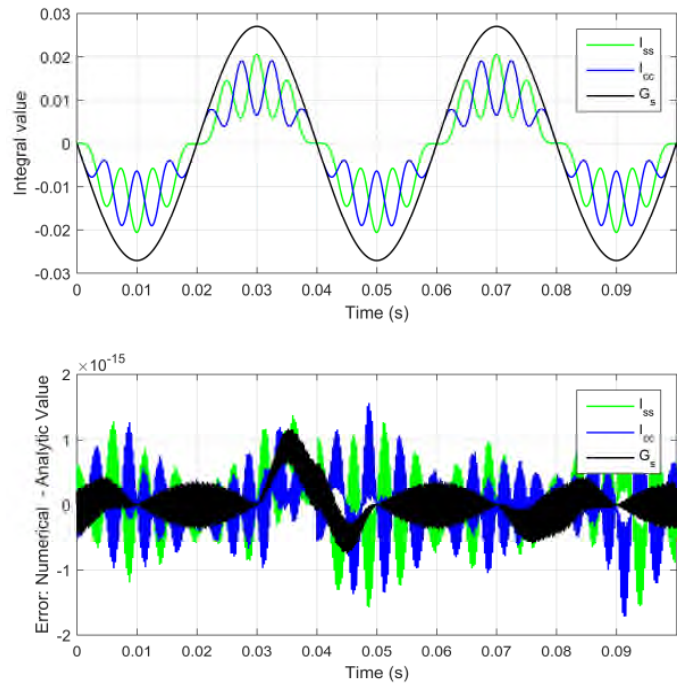

Fig 5: Time series of I_{ss}, I_{cc} and G_s outputs (f = 25 Hz, r = 0.25)

III. Non-Recursive Prism-Equivalent Filters

In this section the impulse response of each of the Prism outputs is determined, from which the coefficients of the corresponding conventional FIR filters are derived. Simulations demonstrate the equivalent outputs are obtained from the Prism and convolution forms of filter. The computational efficiencies are compared in the next section.

Figure 6 shows the impulse responses for G_s and G_c for h = 1, 2 and 3, via numerical simulation. Here the time axis has been scaled in terms of the characteristic frequency m. As would be expected for an FIR filter with a data window of duration $2/m$, each impulse response is zero beyond this time limit. Given the sinusoidal nature of each of the impulse responses, it is straightforward to deduce the corresponding analytic expressions. Denoting the impulse responses for G_s and G_c as H_s and H_c respectively, these are given by:

$$H_s(t) = \frac{1}{2h\pi}\sin(2hm\pi t), \qquad 0 \le t \le 1/m \tag{23}$$

$$H_s(t) = \frac{-1}{2h\pi}\sin(2hm\pi t), \qquad 1/m \le t \le 2/m \tag{24}$$

$$H_c(t) = \frac{1}{2h\pi}[\cos(2hm\pi t) - 1], \quad 0 \le t \le 1/m \tag{25}$$

$$H_c(t) = \frac{-1}{2h\pi}[\cos(2hm\pi t) - 1], \quad 1/m \le t \le 2/m \tag{26}$$

where the difference between the simulated impulse response and the values derived from these equations are of the order of 1e-5 or less. Note that H_s is symmetric about $t = 1/m$ while H_c is anti-symmetric.

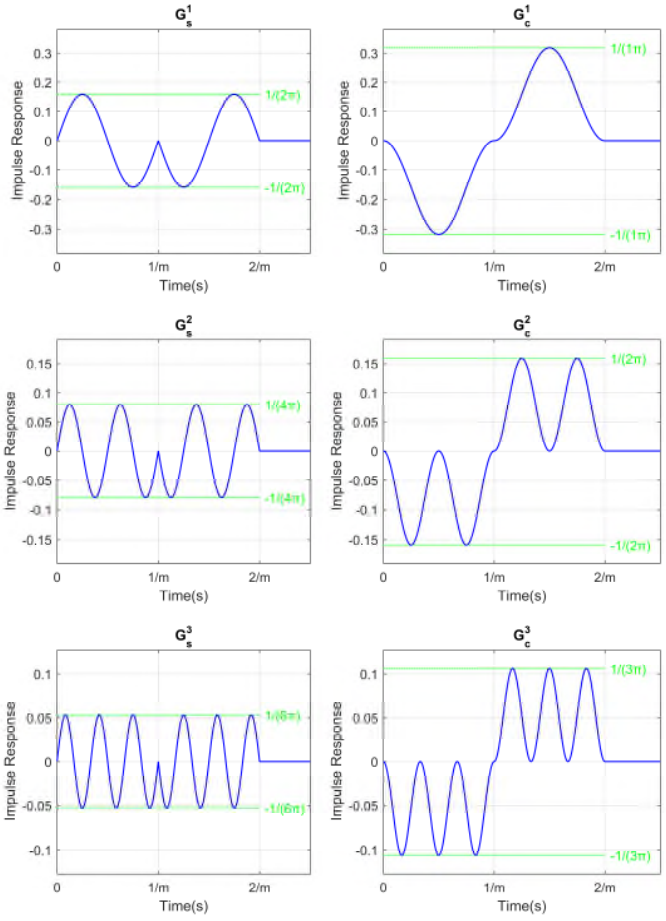

Fig 6: Impulse response of Prism outputs G_s and G_c for h = 1, 2, 3

To calculate the coefficients of the corresponding convolution form of the Prism filters, it is necessary to select the sampling rate and determine the order of the filter. Defining $n = fs/m$, which must be an integer value (as discussed in section II above), then the Prism equivalent filters are of order $2n$. The total number of filter coefficients required is $2n$+1, and their values are given by the following equations:

$$C_s[i] = \frac{1}{2hn\pi}\sin(2h\pi i/n), \qquad 0 \le i < n \tag{27}$$

$$C_s[i+n] = \frac{-1}{2hn\pi}\sin(2h\pi i/n), \qquad 0 \le i \le n \tag{28}$$

$$C_c[i] = \frac{-1}{2hn\pi}[\cos(2h\pi i/n) - 1], \quad 0 \le i < n \tag{29}$$

$$C_c[i+n] = \frac{1}{2hn\pi}[\cos(2h\pi i/n) - 1], \quad 0 \le i \le n \tag{30}$$

where $C_s[i]$ and $C_c[i]$ are the i^{th} coefficients of the equivalent filters of G_s and G_c respectively. Given the input time series $s(t_k)$ these coefficients are used to perform convolution calculations as follows:

$$G_s^*(t_k) = \sum_{i=0}^{2n} C_s[i]\cdot s(t_{k-i}) \tag{31}$$

$$G_c^*(t_k) = \sum_{i=0}^{2n} C_c[i]\cdot s(t_{k-i}) \tag{32}$$

where $G_s^*(t_k)$ and $G_c^*(t_k)$ are the convolution filter outputs at time step k.

Fig. 7 shows the results of applying the non-recursive FIR filters of equations (31) and (32) to the same pure sinusoidal input used in Figs. 3 and 4. Here fs = 48 kHz and m = 100 Hz, so the filter order is 960. The upper plots show the convolution outputs G_s^* and G_c^* while the lower plots show the deviations from the analytical results for G_s and G_c given in equations (11) and (14). All plots show good agreement with the analytic values of the Prism outputs, while manifesting sinusoidally-varying errors. It has not been investigated why, for this example, the errors for G_s^* (of order 1e-7) are significantly larger than those for G_c^* (of order 1e-12): whatever the cause, neither matches the low errors (1e-14 or less) shown by the direct Prism calculation in Figs. 4 and 5. Furthermore, as shown in the next section, the computational cost of using these non-recursive convolution forms is significantly higher than for the Prism. Accordingly, other than for the purposes of directly demonstrating the equivalence of the Prism to a pair of conventional FIR filters, there appears to be limited utility for this non-recursive form of filter design.

IV. Design and Filtering Efficiency

In [1], a set of simulations was used to compare a series of conventional FIR Equiripple and Prism filters, in terms of the computational effort required for filter design and operation. For each filter type, a series of designs was generated by

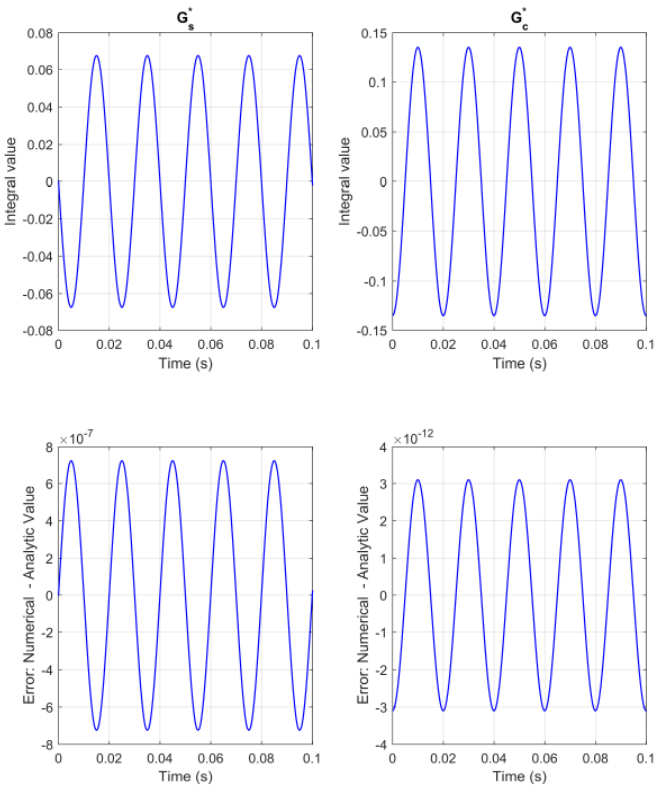

Fig 7: Convolution-form equivalent of Prism outputs G_s^* and G_c^*

increasing the sampling rate *fs* while retaining the same passband and other filter characteristics.

Here another conventional FIR filter type is used – the Least Squares filter –while the sampling rate is fixed at 48 kHz and the filter bandwidth is reduced successively by an order of magnitude, resulting in a corresponding rise in the order of each filter type. Thus for the Prism filters, the value of *m* is reduced from 1 kHz down to 100 Hz, 10 Hz and finally 1 Hz. This results in an equivalent order of the Prism (i.e. its total length in samples over the two stages of integration, equal to $2fs/m$) increasing from 96 to 960, 9,600 and finally 96,000 respectively. Using a high sample rate compared to the value of *m* has practical applications, for example when constructing narrowband Prism filters [9], [10]. The computational requirement of each Prism is also compared with that of the non-recursive FIR filter equivalent developed in section V.

For any Prism, given its length in samples (derived from *fs* and *m*) and harmonic number *h*, the design task consists simply of calculating linearly spaced sine and cosine values (the modulating functions of equation (2)) so that exactly *h* periods of each function occur over each integration data window. This calculation is essentially trivial, so that, as asserted in [1], any computing device capable of evaluating a Prism should also be capable of designing the Prism, or indeed creating a new Prism to match changing signal processing requirements. The design task for the non-recursive equivalent filter, given in equations (29) – (32) above, is equally trivial.

By contrast, the Least Squares design (i.e. the calculation of the filter coefficients given parameters such as the start and stop frequencies, filter order etc) requires dedicated design software. Here the MATLAB filter design toolbox is used, but other design tools could equally be employed. To provide a fair basis for comparison, each Least Squares filter has been designed with the same order as the corresponding Prism filter, with a pass frequency equal to the corresponding *m* and a stop frequency of 1.5 times the pass frequency. All other Least Squares filter design parameters use the MATLAB defaults: the purpose is to find a reasonable basis for comparison rather than fitness for any particular signal processing task.

Table I shows the design time required for the Least Squares filters. All results were obtained using a 2.5 GHz i7-2860 laptop with 16 GB of RAM, running 64 bit Windows 7. Between the 960 and 9,600 order filter there is almost a 100-fold increase in design time, while the 96,000 order filter design could not be completed. Here MATLAB generated an exception due to the construction of a 96k x 96k matrix, requiring 17 Gbyte of memory. While it remains possible that other design software and/or other parameter choices would result in a successfully completed filter design, this example illustrates the very significant resources that may be required to design conventional FIR filters, particularly for high orders.

TABLE I. LEAST SQUARES FIR FILTER DESIGN TIME

Pass frequency (Hz)	Stop frequency (Hz)	Filter order (samples)	Design time (s)
1,000	1,500	96	8.08e-03
100	150	960	4.96e-02
10	15	9,600	4.52e+00
1	1.5	96,000	N/A

Table II compares the Least Squares, the Prism filter and the non-recursive Prism equivalent filter in terms of the number of operations and the average compute time required to process each sample as the filter order increases. Note that for the Prism calculation, only the G_c output is generated, and only a single stage of RI is applied. Similarly, for the non-recursive Prism-equivalent filter, only the G_c output is calculated.

The calculation times for the Least Squares and the non-recursive Prism equivalent filters are similar, as they are performing convolutions with identical filter lengths and differ only in the filter coefficients used. While the Least Squares filter of order 96,000 could not be designed (and so its calculation time per sample is estimated), the Prism equivalent filter was constructed straightforwardly using equations (31) and (32). With both non-recursive techniques, the calculation time per sample increases linearly with filter length.

Even for the smallest order case, the Prism calculation is more than three times faster than the other filters. While the computational overhead for the two non-recursive techniques increases linearly with filter order, the Prism calculation time remains approximately constant. Hence the Prism of order 96,000 requires about 3,000 times less computation time per sample than both the corresponding non-recursive equivalent and (by extrapolation) the Least Squares filter.

TABLE II. LEAST SQUARES AND PRISM FILTER CALCULATION TIME

Pass frequency or *m* (Hz)	Filter order (samples)	Convolution ops per sample		Prism ops per sample		Filter time per sample (s)		
		Multiply	Addition	Multiply	Addition	Least Squares	Non-recursive Prism equivalent	Prism
1,000	96	97	97	18	33	8.79e-08	8.99e-08	2.57e-08
100	960	961	961	18	33	8.22e-07	8.25e-07	2.65e-08
10	9,600	9,601	9,601	18	33	8.17e-06	8.19e-06	2.66e-08
1	96,000	96,001	96,001	18	33	(8.00e-05)	8.22e-05	2.66e-08

Figs. 8 – 11 demonstrate the similarities of the frequency responses for the Least Squares and Prism filters. Each graph shows both the theoretical frequency response and the actual numerical performance when filtering white noise sampled at 48 kHz. Good agreement is shown between theoretical and numerical performance in each case. The Least Squares filters shown have pass frequencies of 1 kHz (Fig. 8) and 10 Hz (Fig. 9). Other than the frequency scaling (indicated by the different ranges of their respective x-axes), the two filters have similar characteristics, with a flat passband and a steadily dropping stopband with regular notches.

The Prism filter outputs are shown in Figs. 10 and 11, with m = 1 kHz and 10 Hz respectively. The frequency ranges on the x-axes correspond to those of Figs. 8 and 9 respectively, while the gain shown is relative to the absolute maximum gain value. Notches occur at all multiples of m Hz, including 0 Hz.

The Prism filter G_c outputs have a low pass characteristic similar to that of the Least Squared filters, other than a more rapid rate of attenuation in the stopband, and the absence of steady gain in the 'passband' for frequencies below m Hz.

While the theoretical developments in Section II consider only a pure sinusoidal input, figs. 10 and 11 illustrate how the Prism filters a signal containing multiple frequency components, for the special case of pure white noise. As would be expected for a linear filter, the output consists of the summation of each input frequency component to which the corresponding Prism gain and phase delay has been applied. Further illustrations are provided in [1], where dynamic notch filtering is applied to 'split' a two-tone signal into its respective components, and in [10], which demonstrates tracking a sinusoidal signal in white noise with an SNR of -60dB, through the use of ultra-narrow Prism band pass filtering. In considering the filtering 'efficiency' of the Prism, the Cramér-Rao Lower Bound (CRLB) [31] is widely used in signal processing to determine the 'best' (i.e. smallest variance) performance theoretically achievable when extracting a parameter value from a data set contaminated by white noise. The Prism-based Recursive Signal Tracker (RST) [7] calculates frequency, phase and amplitude values for an input sinusoid, and performs close to the CRLB limit for each of these parameters. A more complete analysis of the noise performance of the RST, which requires a full analysis of the mathematical technique, will be given in a future paper.

V. STABILITY AND IMPLEMENTATION CONSIDERATIONS

In this section we consider numerical stability and implementation issues, particularly for low cost platforms. As an FIR filter, the Prism has inherent numerical stability. It is straightforward to show that for the non-recursive, convolutional form given in equations (27) - (32), a bounded input time series (say $|s(t_k)| < K$) will generate a bounded output series with lower magnitude. As sine and cosine values have a magnitude no greater than unity, the Prism outputs are constrained as follows:

$$|G_s^*(t_k)| \le \sum_{i=0}^{2n} |C_s[i]| \cdot |s(t_{k-i})| \le \frac{2n+1}{2hn\pi} K \le K \qquad (33)$$

$$|G_c^*(t_k)| \le \sum_{i=0}^{2n} |C_c[i]| \cdot |s(t_{k-i})| \le \frac{2n+1}{hn\pi} K \le K \qquad (34)$$

When implementing the Prism in its default recursive form, errors may also accumulate from the use of a sliding window calculation. An efficient sliding window implementation requires totalizers to which the latest product value is added and from which the oldest product value is removed, which is liable to the accumulation of rounding errors. This issue can be addressed in a number of ways, for example:

1. A fixed point implementation is immune from rounding error accumulation. As shown in section IV, fixed point integration totalizers can nevertheless generate high precision results if the bit lengths are sufficiently long.
2. Two sets of totalizers may be used, with only one set in active use at any time. On a regular basis, the inactive set is zeroed and its value recalculated from the current data window, thereby flushing out rounding errors, before becoming the new active set.
3. The examples given so far in this paper have been selected to demonstrate high precision, and in particular the close agreement between continuous time analytical results and discrete time numerical values, as a means of explaining the mathematics underlying the Prism. However, a further claim made for the Prism (for example in [1]) is that it is suited to low-cost instrumentation applications, for example in the Internet of Things. A demonstration of a 'low cost' recursive Prism implementation is shown in Fig. 12. For ease of comparison, the same parameters are used from earlier examples (e.g. Figs. 3, 4, and 7): *fs*

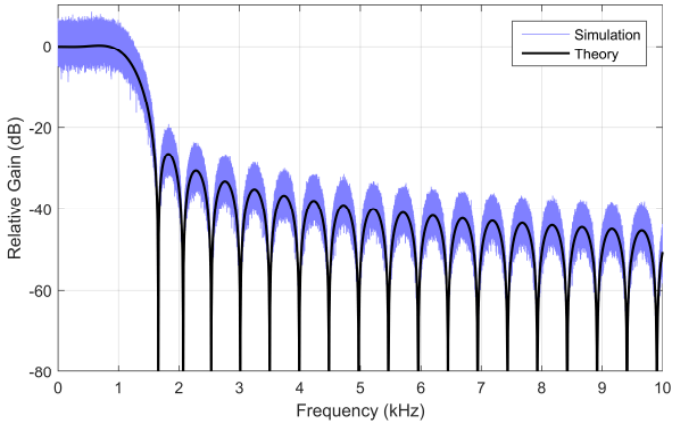

Fig 8: Least Squares FIR filtering performance with f_{pass} = 1 kHz

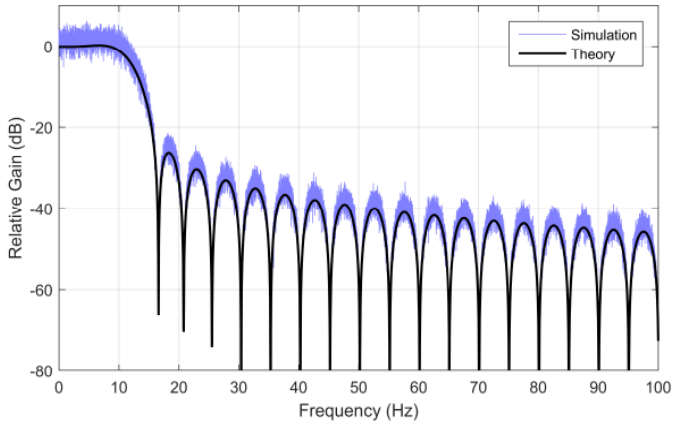

Fig 9: Least Squares FIR filtering performance with f_{pass} = 10 Hz

= 48 kHz, m = 100 Hz, so that the filter order is 960. Here all calculations are done in floating point (32 bit) precision, including the integration totalizers. The corresponding errors in G_s and G_c are of the order ± 1e-6, compared with the ± 1e-14 results obtained using the 4 RI stage, double precision implementation used to generate Figs. 3 and 4.

The number of floating point operations per sample is a function of the number of Romberg Integration (RI) stages. Here a single RI stage is used. To generate both Prism outputs (i.e. including all 6 Prism integration blocks in Fig. 1) requires 26 multiplications and 50 additions per sample; to generate only one Prism output (either G_s or G_c, requiring the two first stage integration blocks of Fig. 1 and only two of the second stage blocks) requires 18 multiplications and 33 additions. This computational requirement is fixed, irrespective of filter length, and contrast favourably with the 961 each of multiplications and additions required for a convolution calculation of the same filter order (Table II).

VI. Discussion

In the Prism, the linear phase and numerical stability of conventional FIR filtering are combined with the low computational cost conventionally associated with IIR filtering.

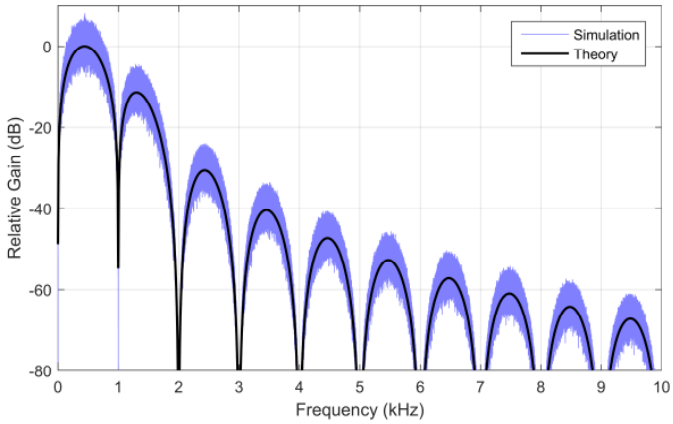

Fig 10: Prism filtering performance (G_c output only) with m = 1 kHz

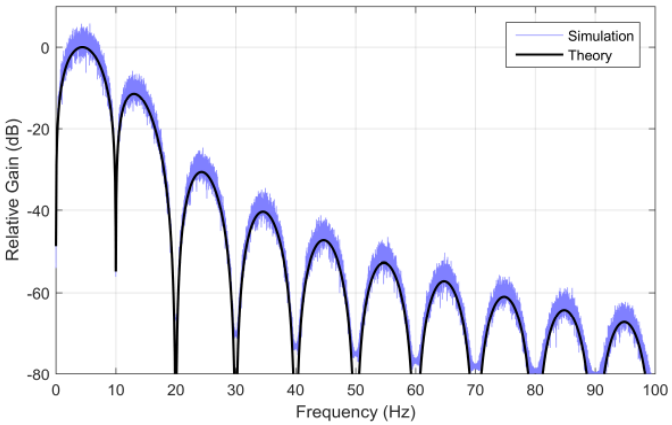

Fig 11: Prism filtering performance (G_c output only) with m = 10 Hz

Where two Prism outputs are generated they are orthogonal, facilitating the tracking of amplitude, phase and frequency properties, which is particularly useful in in instrumentation applications. These benefits have been achieved by adopting an entirely new approach to filter construction.

In any non-trivial FIR filter, the contribution of each sample to the filter output must vary as it passes through the filter's data window. The conventional means of achieving this, given some desired filtering performance, is to directly calculate the weighting associated with each position in the data window. These positional weights constitute the filter coefficients, as conventionally understood. This design calculation is usually sophisticated (e.g. [32]) and may require significant computational resources. Once the design is complete, the filter output is calculated by convolution i.e. by accumulating the sum, over the full data window, of the products formed from each sample value and the coefficient corresponding to the weighting of its current position. However, when a new sample arrives, all the data are shifted one position along in the data window, so that each sample must be weighted by a different coefficient associated with its new position. Accordingly, the convolution must be repeated in full across the entire data window for each new sample.

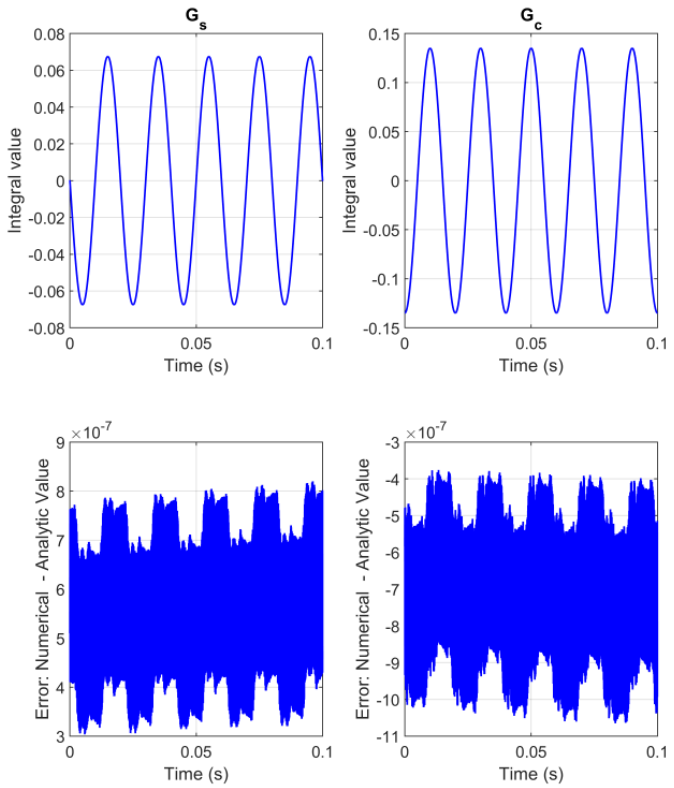

Fig 12: Prism outputs G_s and G_c using 32-bit floating point implementation

In the Prism, each sample retains a fixed weighting as it passes through a first stage integral, thereby eliminating the need to recalculate all products each time step and so facilitating recursive calculation. This fixed first stage weighting is determined only by the phase q of the modulation functions when the sample first enters the Prism. However, the Prism structure implicitly generates variable weighting at the second stage, as the weighting of a sample in each second stage integral is in part a function of its residence time in the first stage integrals. Furthermore, combinations of the second stage integrals result, algebraically, in the elimination of the influence of q, resulting in recursively calculated outputs which are simple analytic functions of the original input signal.

This new approach to filtering eliminates the need for an elaborate design calculation to obtain positional weightings explicitly, while achieving the same end indirectly. As demonstrated in Section III, it is possible to calculate, for a given set of Prism parameters *fs, m* and *h*, an equivalent set of positional coefficients, which may be used to generate essentially identical outputs to the Prism using the conventional FIR convolution calculation method. As demonstrated in Section IV, this approach loses the recursive property, so that the resulting filter is just as costly to operate as any other conventional FIR filter of the same order, and therefore has little to recommend it. As Section IV further demonstrates, for long filters, the Prism calculation can be achieved with several orders of magnitude less computation resources than the conventional FIR equivalent.

The price to be paid for the Prism's low computational cost is design inflexibility: it is not currently possible to specify some desired frequency response (say) and design a corresponding Prism which offers a recursive calculation. However, given the computational advantage the Prism enjoys over conventional FIR filters, greater design flexibility can be introduced by the creation of Prism networks where the properties of each Prism are selected to create a desired filtering performance for the whole network. For example, [9], [10] show how a chain of six Prisms can be used to build a bandpass filter having a desired central frequency and bandwidth. Ultra-narrow band filters with orders of several hundred million can be constructed from a chain of Prisms: real-time filtering is achieved on a single FPGA, where the conventional non-recursive FIR calculation would require supercomputer resources to achieve the same throughput – assuming such a conventional filter could be successfully designed. The narrowband technique uses a combination of high sampling rate and low values of m (as illustrated in section VI), together with high values of h.

The Prism therefore offers an alternative approach to signal processing design, based on creating networks of low cost and relatively simple (and inflexible) processing nodes. As argued in [1], other network processing approaches have proved highly successful, most obviously in the case of neural networks. The Prism's properties may thus render it a useful tool for the development of next generation instrumentation, particularly in the context of the greater flexibility and autonomy required by the Internet of Things [1].

VII. Conclusions

This paper has presented the mathematics underlying the operation of the Prism as a fully recursive FIR filter. It has demonstrated that the continuous time analytic equations from which the Prism is derived can be matched to high precision (e.g. to within ± 3e-14) using discrete time numerical calculations through the application of Romberg Integration. A conventional, convolutional, non-recursive FIR filter equivalent to the Prism has been derived. Simulations have demonstrated the computational advantage of the recursive Prism calculation over both its convolutional equivalent and the widely used Least Squares FIR filter. For filter orders of 96 up to 96,000 the Prism computation was 3 to 3,000 times faster than the convolutional forms. Prism implementation issues have been discussed, with a particular focus on low-cost, low precision (e.g. single floating point) platforms. Using a single stage of Romberg Integration, 26 multiplications and 50 additions are needed to update both Prism outputs for each new input value, irrespective of filter length, while 18 multiplications and 33 additions are required to generate a single Prism output.

Further discussion on the Prism and its applications are given in [1] – [10]. Future publications will provide more detailed mathematical developments of specific topics, including:

- The application of Romberg Integration to time series data within the Prism
- Prism-based trackers which estimate frequency, amplitude and/or phase values of an input sinusoid.

- The design of low pass, band pass, and notch filters using Prism networks.
- Dynamic notch filtering: separating two or more sinusoids simultaneously in the same input signal using Prisms.
- Instrumentation applications.

REFERENCES

[1] Henry, MP, Leach, F, Davy, M, Bushuev, O, Tombs, MS, Zhou, FB and Karout, S. "The Prism: Efficient Signal Processing for the Internet of Things", IEEE Industrial Electronics Magazine, pp 2–10, December 2017.

[2] Henry, MP. "An Introduction to Prism Signal Processing applied to Sensor Validation", Measurement Techniques, pp 1233 – 1237, Mar 2018.

[3] Henry, MP, Bushuev, O, Ibryaeva, O. "Prism Signal Processing for Sensor Condition Monitoring", 26th IEEE International Symposium on Industrial Electronics, Edinburgh, UK, 2017.

[4] Henry, MP, Sinitsin, V. "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] Henry, MP, Sinitsin, V. "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] Chen, X, Henry, MP, Duncan, SR. "An Enhanced Algorithm for Frequency Estimation in Power Systems", 2018 UKACC 12th International Conference on Control, Sheffield, Sept 2018.

[7] Leach, F, Karout, S, Zhou, FB, Tombs, MS, Davy, M and Henry, MP. "Fast Coriolis mass flow metering for monitoring diesel fuel injection", Flow Measurement and Instrumentation, 58 (2017), pp 1–5.

[8] Leach F, Davy M, Henry M, Tombs M, Zhou F. "A new method for measuring fuel flow in an individual injection in real time", SAE International Journal of Engines, Volume 11, 2018.

[9] Henry, MP. "Ultra narrowband filtering with Prism signal processing: design and simulation", IEEE IECON, Washington DC, Oct 2018.

[10] Owen, J, Henry, MP. "384 TMAC/s FIR filtering on an Artix-7 FPGA using Prism signal processing", IEEE IECON, Washington DC, Oct 2018.

[11] R. Lyons and A. Bell, "The swiss army knife of digital networks," in IEEE Signal Processing Magazine, vol. 21, no. 3, pp. 90-100, May 2004.

[12] T. G. Campbell and Y. Neuvo, "Predictive FIR filters with low computational complexity," in IEEE Transactions on Circuits and Systems, vol. 38, no. 9, pp. 1067-1071, Sep 1991.

[13] T. Saramaki and A. T. Fam, "Properties and structures of linear-phase FIR filters based on switching and resetting of IIR filters," IEEE International Symposium on Circuits and Systems, New Orleans, LA, 1990, pp. 3271-3274 vol.4.

[14] E. Hogenauer, "An economical class of digital filters for decimation and interpolation," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 29, no. 2, pp. 155-162, Apr 1981.

[15] M. Mottaghi-Kashtiban, A. Jalali, "FIR filters involving shifts and only two additions, efficient for short word-length signal processing",Microelectronics Journal, Volume 49, 2016, Pages 57-63

[16] S. Chu and S. Burrus, "Efficient recursive realizations of FIR filters, Part I: The filter structures," Circuits Syst. Signal Process., vol. 3, no. 1, pp. 3–20, 1984.

[17] S. Chu and S. Burrus, "Efficient recursive realizations of FIR filters, Part II: Design and Applications," Circuits Syst. Signal Process., vol. 3, no. 1, pp. 21–57, 1984.

[18] R. Lehto, T. Tauren, and O. Vainio, "Recursive FIR filter structures on FPGA," Microprocess. Microsyst., vol. 35, no. 7, pp. 595–602, Oct. 2011.

[19] K. Mukumoto and T. Wada, "Realization of Root Raised Cosine Roll-Off Filters Using a Recursive FIR Filter Structure," in IEEE Transactions on Communications, vol. 62, no. 7, pp. 2456-2464, July 2014.

[20] H. Murakami, I. Reed and A. Arcese, "Recursive FIR digital filter design using a z-transform on a finite ring," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 31, no. 5, pp. 1155-1164, October 1983.

[21] G. Jullien, R. Krishnan and W. Miller, "Complex digital signal processing over finite rings," in IEEE Transactions on Circuits and Systems, vol. 34, no. 4, pp. 365-377, Apr 1987.

[22] H. Murakami, "Multiple FIR filters on a finite ring," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 36, no. 5, pp. 686-692, May 1988.

[23] H. Kikuchi, H. Watanabe, and T. Yanagisawa , "Linear Phase FIR Digital Filters with Cyclotomic Resonators", IEICE Transactions (1976-1990), Vol.E70, No.1, p.24-32.

[24] K. Berberidis and S. Theodoridis, "New Levinson, Schur, and lattice type algorithms for linear phase filtering," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 38, no. 11, pp. 1879-1892, Nov 1990.

[25] A. Bartolo, B. D. Clymer, R. C. Burgess and J. P. Turnbull, "An efficient method of FIR filtering based on impulse response rounding," in IEEE Transactions on Signal Processing, vol. 46, no. 8, pp. 2243-2248, 1998.

[26] O. Kwon, W. Kwon, K. Lee, "FIR filters and recursive forms for discrete-time state-space models", Automatica, Vol. 25, No. 5, 1989, p. 715-728.

[27] J. Han and D. Hatzinakos, "Joint blind deconvolution, fusion and classification of multi-frame imagery," Canadian Conference on Electrical and Computer Engineering 2004, pp. 2061-2064, Vol.4.

[28] V. Myasnikov V. Methods for designing Recursive FIR filters. In: Wojciechowski K., Smolka B., Palus H., Kozera R., Skarbek W., Noakes L. (eds) Computer Vision and Graphics. Computational Imaging and Vision, vol 32. Springer, Dordrecht, 2006.

[29] H. L. Kennedy, "Isotropic Estimators of Local Background Statistics for Target Detection in Imagery," IEEE Geoscience and Remote Sensing Letters, 2018.

[30] J. Dutka, "Richardson Extrapolation and Romberg Integration", Historia Mathematica, Vol. 11, pp 3-21, 1984.

[31] D. Rife and R. Boorstyn, "Single tone parameter estimation from discrete-time observations," IEEE Transactions on Information Theory, vol. 20, no. 5, pp. 591-598, Sep 1974.

[32] T. Parks and J. McClellan, "Chebyshev Approximation for Nonrecursive Digital Filters with Linear Phase," in IEEE Transactions on Circuit Theory, vol. 19, no. 2, pp. 189-194, March 1972.