Spectral Analysis Techniques using Prism Signal Processing

Henry, M.

DOI 10.1016/j.measurement.2020.108491
ISSN 0263-2241

Publisher: Elsevier

Spectral Analysis Techniques using Prism Signal Processing

(Abbreviated Title: "Prism Spectral Analysis")

Manus Henry[1,2,3]

[1]Fluid and Complex Systems Research Centre, Coventry University, UK.
[2]Department of Engineering Science, University of Oxford, UK.
[3]School of Electrical Engineering and Computer Science, South Ural State University, Chelyabinsk, Russia

manus.henry@eng.ox.ac.uk

Abstract. The Prism is a new signal processing module which implements fully recursive, linear phase FIR filtering, so its computational cost is fixed irrespective of filter length. The Prism also has negligible design cost. Recent work has demonstrated how, using simple design rules, a chain of six Prisms can create a narrowband filter with arbitrary central frequency and bandwidth. In this paper, the technique is applied to spectral analysis of data, whereby a sequence of filters is applied to a data set to provide narrow frequency analysis, yielding accurate estimates of the frequencies and amplitudes of spectral peaks in the data. Although this time domain technique remains computationally expensive compared to the FFT, it can identify and reject spectral leakage, offering an alternative analysis for low amplitude and/or adjacent spectral peaks, including hidden tones, where FFT discrimination may be limited.

Keywords: FFT, Prism signal processing, spectral analysis, spectral leakage, hidden tones

1. Introduction: Prism signal processing

The Prism (precise, repeat integral signal monitor), is a recently developed finite impulse response (FIR) signal processing module. It is based on a Fourier-style double integration applied recursively via a sliding windows calculation. Either one or two outputs are generated. If there are two, these form a sine/cosine pair from which the properties of an (assumed sinusoidal) input signal, such as amplitude, phase and frequency, can be calculated. Unlike conventional, convolution-based FIR filters, the Prism calculation is fully recursive, having a low computational cost per sample which is independent of window length; its benefits include a linear phase response, and a negligible design cost, as the filter 'coefficients' are simply linearly spaced sine and cosine values. Although the range of frequency responses available from a single Prism is limited, serial and/or parallel networks of Prisms can be created to provide a wider range of responses suited to a variety of tasks, including notch and bandpass filtering, or the tracking of multiple frequency components in a signal.

In [1], Prism Signal Processing (PSP) is introduced as having potential value for the Internet of Things/Industry 4.0. Current applications include sensor validation [2] generally, pressure sensors [3], the condition monitoring of rotating machinery [4, 5], and the first application of Coriolis mass flow metering to fuel injection monitoring in internal combustion engines [6], where both a fast dynamic response and good noise suppression are required.

The mathematics of the Prism is detailed in [7]. Consider a sinusoidal input signal (or one frequency component in a multi-component signal):

$$x(\tau) = A\sin(2\pi f\tau + \phi), \tag{1}$$

where t (or τ) is time, A is the amplitude, f is the frequency, and ϕ is the phase offset at t = 0. The basic mathematical operation used in the Prism is to calculate double integrals of the form:

$$I^h_{[S|C][S|C]} = m^2 \int_{-\frac{1}{m}}^{0} [\sin|\cos](2\pi hmt)\left(\int_{t-\frac{1}{m}}^{0} [\sin|\cos](2\pi hmt)\cdot x(\tau)d\tau\right)dt \quad (2)$$

where [s | c] indicates either s (sin) or c (cosine), h is the harmonic number, a positive integer, and m is the characteristic frequency of the Prism. Equation (2) describes a family of DFT–style double integrals whereby at each stage the input signal is multiplied by a modulating sine or cosine function and is integrated over the period of the characteristic frequency m. The harmonic number h (a positive integer) determines how many whole cycles of the modulation function occur over the period of integration $1/m$. In other words, the modulation function consists of h cycles of frequency hm. In any Prism, either one or two pairs of such integrals are updated every sample, using an efficient sliding window calculation. As explained in [7], summing the outputs of each integral pair produces a result that is independent of the current phase of the sliding window integrators. The resulting Prism output is one or both of:

$$G_s^h = I_{ss}^h + I_{cc}^h = A\operatorname{sinc}^2(r)\frac{r^2}{r^2-h^2}\sin(\phi-2\pi r) \quad (3)$$

$$G_c^h = I_{cs}^h - I_{sc}^h = A\operatorname{sinc}^2(r)\frac{hr}{r^2-h^2}\cos(\phi-2\pi r) \quad (4)$$

where $r = f/m$ is the dimensionless frequency of the input sinusoid $x(t)$. Hence, a Prism has distinct frequency responses for G_s^h and G_c^h for $h = 1, 2, \ldots$ [1]. The gains of the outputs, Γ_s and Γ_c, are readily derived as functions of r and h:

$$\Gamma_s(r,h) = sinc^2(r)\frac{r^2}{r^2-h^2} \quad (5)$$

$$\Gamma_c(r,h) = \frac{h}{r}\Gamma_s(r,h) \quad (6)$$

Given two orthogonal outputs from a Prism, further calculations can be applied to track the phase, frequency and/or amplitude of $x(t)$. Equations (3) and (4) assume constant values for r, A and ϕ, but only over the current data window. When applied to a time series on a sliding window basis, the parameter estimates, including phase, are updated to reflect the current data window so that parameter variation can be tracked sample-by-sample ([4 - 6]). Notches occur at all multiples of m, so tracking Prisms are designed to ensure that the frequency range of interest falls between notches. For example, with the default value $h = 1$, good tracking can be achieved for the approximate range $r \in [0.3, 0.7]$. A higher value of h may be appropriate in a tracker to reduce the influence of noise or to ensure sufficient window length when tracking high frequency components.

The Prism calculation employs an efficient time series implementation of Romberg Integration to generate high precision outputs [7]. The combined properties of efficient, recursive calculation, trivial design (for desired values of m and h), high precision, and linear phase, all support the use of the Prism as a convenient building block for creating a range of signal processing functions [1].

In [8, 9], simple design rules are developed for building narrowband FIR filters using a chain of six single output Prisms, grouped as three pairs and arranged in series. Figure 1 shows the frequency response of the generic Prism filter for an arbitrary central frequency f_c and bandwidth b. Given desired values of f_c and b the corresponding values of m and h for each of the Prisms can be calculated using simple expressions, as described later in this paper.

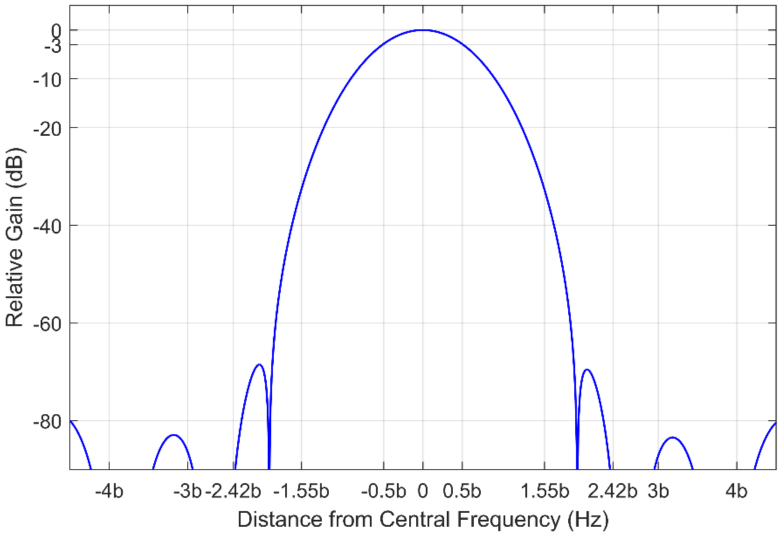

Figure 1. Frequency response of bandpass filter formed from three Prism pairs.

For real-time applications, Prism narrowband filtering offers a significant speed-up compared with conventional FIR filtering. Ref. [9] describes an FPGA implementation of real-time Prism narrowband filtering, in which a signal sampled at 2 MHz is filtered around a user selected central frequency with a bandwidth as narrow as 0.02 Hz. This requires a total Prism filter length of 193 million samples, but because the calculation is recursive, it can run in real time on a single FPGA, given sufficient memory storage. A conventional, convolution-based, non-recursive FIR filter would require 386 TMAC/s (i.e. 3.86 x 10^{14} multiply-and-accumulate operations per second - supercomputer performance) to achieve this throughput, assuming such a long filter could be designed. The combination of instant filter design and fast filter throughout provides the basis for developing Prism spectral analysis, whereby a series of narrowband filters and trackers is used to examine specific spectral regions of the dataset.

2. FFT spectral analysis – working example

The Fast Fourier Transform (FFT) is one of the most widely used techniques in signal processing [10], generating a set of Discrete Fourier Transform (DFT) results, linearly spaced in the frequency domain. Its key advantage is an *o(n log(n))* computational cost, for a time series of length *n* samples. However, the extraction of frequency, amplitude and/or phase information for the spectral peaks obtained from FFT/DFT calculations remains an active research topic ([11 – 16]). In short, unless a peak happens to coincide exactly with a DFT/FFT frequency, interpolation and/or other techniques are required to estimate the true peak frequency and amplitude. Additional challenges include spectral leakage, where DFT/FFT processing of a peak at one frequency may result in the apparent transfer of energy to adjacent frequencies which have negligible energy in the original signal. This in turn may result in 'hidden tones' [12] whereby a true peak of low magnitude, in the near vicinity of a higher peak, is obscured.

An example is used to illustrate the difficulties presented by spectral leakage and hidden tones, in order to demonstrate the potential benefits of Prism signal processing for spectral analysis. Consider the time series of a signal, sampled at 10 kHz over a period of 2 seconds, with two components:

1. Frequency = 200.123 Hz, amplitude 0.5 V
2. Frequency = 234.567 Hz, amplitude 1e-9 V

These frequencies are selected to avoid any favourable factorization of either the sampling rate or the time series duration. To demonstrate the high precision of the Prism technique, no noise is added. The phase of each component is initialized to zero radians at the start of the data set. The second component has very low magnitude and near proximity to the first, so that, using conventional FFT analysis, it becomes a 'hidden tone'. Figure 2 shows the FFT spectrum obtained after applying the conventional Hann/Hanning window [12] to the original signal. The true tone at ~ 234 Hz is more than two orders of magnitude smaller than the FFT spectrum leakage from the ~ 200 Hz tone. Even to detect the presence of the minor tone, based solely on the FFT output, is challenging; the requirement is to determine its frequency and amplitude.

Further issues associated with extracting peak frequency and amplitude data from an FFT spectrum are illustrated in Figure 3. This shows how, in the region of 200 Hz, the maximum magnitude (top graph) and corresponding frequency (middle graph) vary with the number of samples in the data set. These results has been calculated by cropping between 0 and 120 samples from the end of the original data set, recalculating the FFT for the cropped set, and finding the frequency and magnitude of the peak value around 200 Hz. Even though the full dataset contains 20,000 points, a small change in the data set length may result in a large variation in the observed peak magnitude. This occurs because the set of discrete frequencies for which the FFT is calculated varies with the data set length, and the observed peak magnitudes are a function of the difference between the true peak frequency and the closest FFT frequencies. The lowest graph shows, on the right axis, the number of periods of the 200.123 Hz tone in the data set, which decreases linearly as the number of cropped samples increases. On the left axis, the δ parameter ([11 - 16]) shows, for the different data set lengths, the distance between the true tone frequency and the nearest FFT frequency value, scaled by the FFT frequency step length (in this example 0.5 Hz). This illustrates the well-known result that an integral number of tone periods in the data set (right axis), corresponding to $\delta = 0$ (left axis), results in FFT peak frequency and magnitude values being close to the true values for the tone. Where only a single tone with known peak frequency is the subject of analysis, the FFT data set length may be adjusted to improve amplitude estimation.

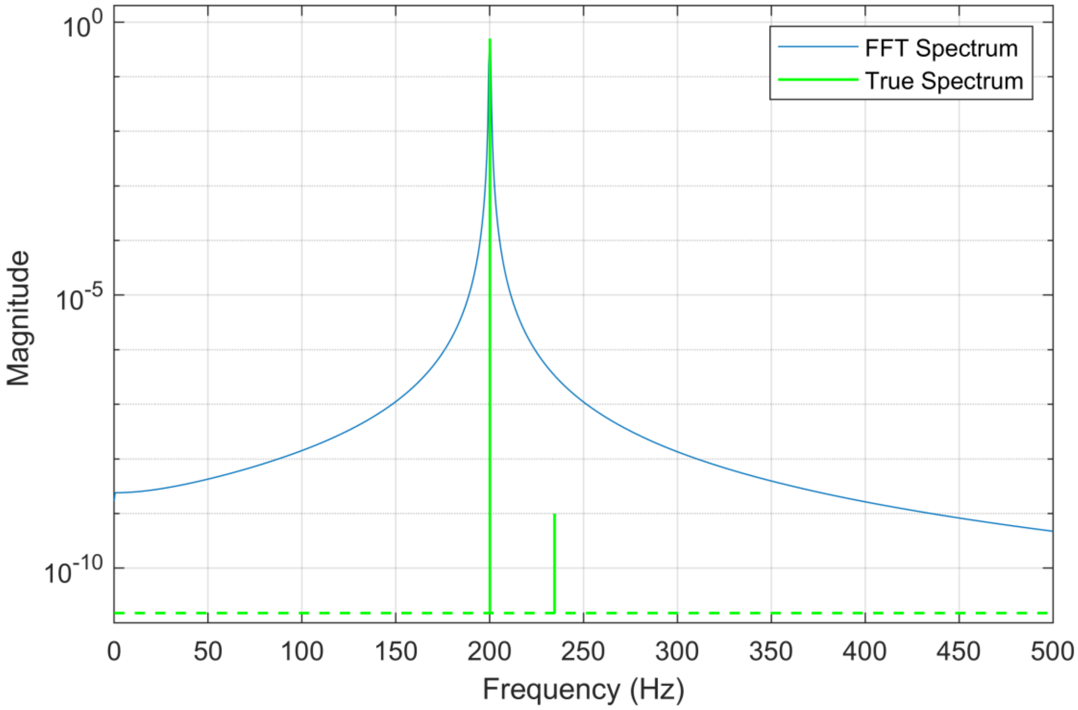

Figure 2. Simulation example: two tones with FFT output.

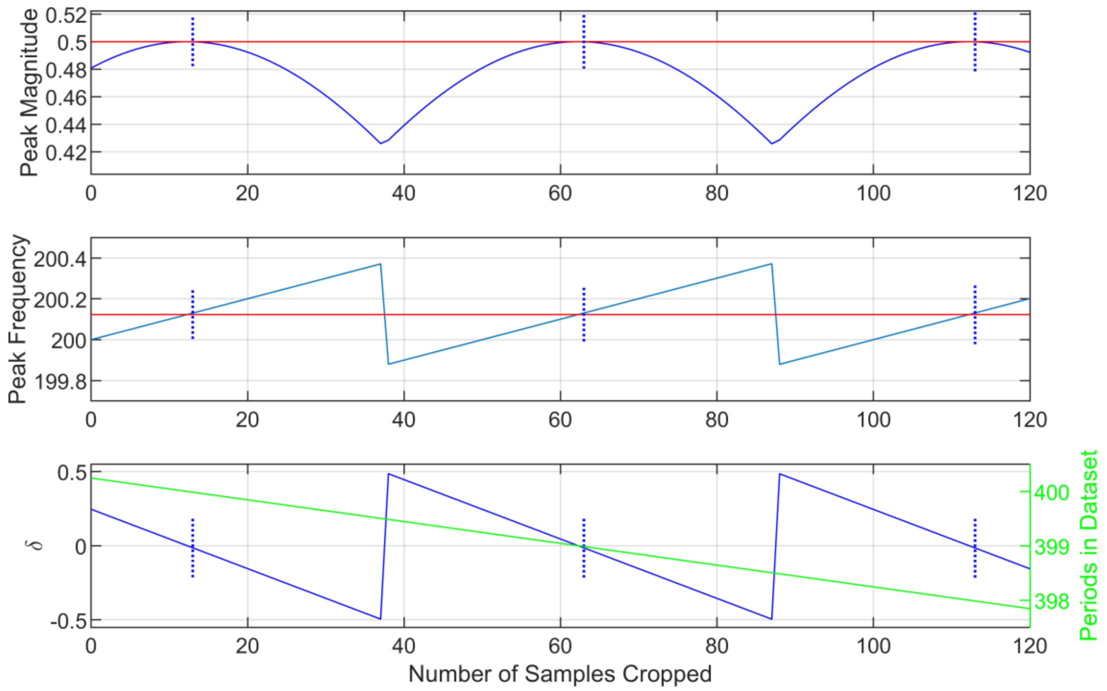

Figure 3. FFT peak frequency and amplitude around 200 Hz with varying dataset length.

FFT-based techniques for estimating tone frequency and amplitude ([11 – 16]) typically begin by identifying each peak, and then perform additional calculations, for example interpolating between one or more adjacent FFT/DFT results. Where peaks are sufficiently separated so that spectral leakage causes little or no interference between them, the techniques described in the literature deliver accurate frequency and amplitude estimates, close to the Cramer-Rao Lower Bound [11], with high computational efficiency. Where spectral leakage and/or tone proximity leads to significant tonal interference [12], or indeed hidden tones [13], then these techniques are less effective. For example, Figure 4 illustrates the behaviour of the FFT output nearest to the 234 Hz tone with varying record length: there is of course no true ‘peak’ observed as the local region of the spectrum is dominated by leakage from the adjacent, much higher peak at ~ 200 Hz (Figure 2). By definition, the reported FFT frequency matches the true tone when $\delta = 0$, but the corresponding magnitude does not, as it reaches up to 500 times greater than the true hidden peak amplitude.

Liguori and Paolillo [13] describe a technique to detect hidden tones, in which the spectral characteristics of non-hidden tones are compared to their expected behavior if no hidden tone is present. This technique is based on FFT and is computationally efficient, but is effective for tone amplitude ratios of no greater than 80:1. In the example under consideration here, the amplitude ratio is 500,000,000:1.

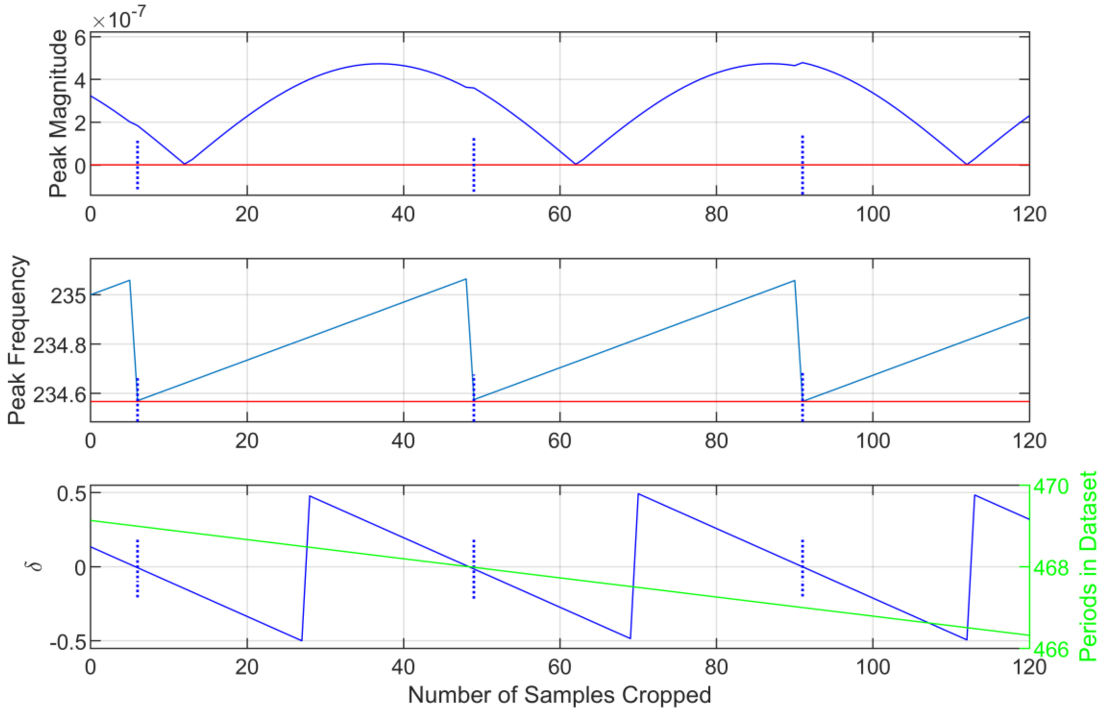

Figure 4. FFT peak frequency and amplitude around 234 Hz with varying dataset length. In the top plot, the true peak amplitude (red) is 1e-9 V.

3. Prism spectral analysis

This section demonstrates a range of Prism signal processing techniques by addressing the problem of finding low amplitude and/or hidden tones in the difficult example presented above. The intention is to be illustrative rather than striving for optimal computationally efficiency: optimality will depend upon the specific known features of any particular problem – for example whether the approximate location of a hidden peak is known, the relative amplitudes, frequency difference, number of peaks, data set length etc [13]. The efficiency issues are addressed further in the concluding section. Here we proceed to demonstrate how a series of narrow bandwidth Prism-based trackers can be created to search for peaks, including low amplitude and hidden peaks, in the test signal below (say) 1000 Hz, and to calculate the frequency and amplitude for each located peak.

3.1 *Bandpass and Tracker Design*

Figure 5 shows the architecture used in ([8], [9]) to provide real-time tracking of a signal, formed from a bandpass filter and tracker. The bandpass filter consists of six single output Prisms arranged as a series. More specifically, these are three pairs of Prisms, where each pair has a common value of h and similar but distinct values of m, as illustrated below. Collectively, the Prism chain matches the desired central frequency f_c and bandwidth b. This is followed by a dual output Prism which acts as a tracker. The phase, frequency and amplitude of the input signal are calculated from the tracker Prism outputs. Applied in real-time, all Prisms are updated every sample, while the estimates of phase, frequency and amplitude are updated at either the same or a lower rate as required. For this recursive FIR calculation, the computational cost is fixed irrespective of the length of the filters, so that the only constraints on the filter bandwidth b are available memory and the desired dynamic response of the tracker. In a typical instrumentation application (such as Coriolis mass flow metering [6]), the central frequency and bandwidth can be selected to match the expected frequency range of the input signal(s).

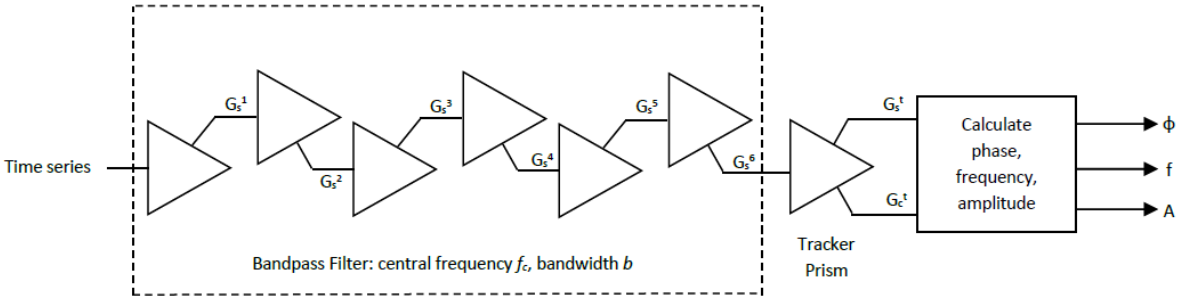
Time series
Gs1
Gs2
Gs3
Gs4
Gs5
Gs6
Bandpass Filter: central frequency fc, bandwidth b
Gst
Gct
Tracker Prism
Calculate phase, frequency, amplitude
ϕ
f
A

Figure 5. Prism signal processing scheme for time series

Figure 6 shows how this real-time architecture has been adapted to analyse a static, fixed length data set, at central frequency f_c with bandwidth b and sampling rate f_s. The bandpass filtering and tracking Prism designs are essentially the same as for Figure 5. Note however, that the length of a bandpass filter (for any chosen f_c) is approximately 1.93 f_s / b; b must therefore be chosen so that the filter is shorter than the data set. Here f_s = 10 kHz and the data record is 2s duration; selecting b = 1.5 Hz gives a filter length of approximately 13,000 samples. The filter design procedure from [8] is as follows. The h values for each Prism pair are given by:

$$h = \text{round}\,(0.0371 \times f_c/b \times [6, 4, 3]) \tag{7}$$

Then for each h value the corresponding values of m for the Prism pair are given by:

$$m_1 = f_c/(h + 0.371) \text{ and } m_2 = f_c/(h - 0.371) \tag{8}$$

In practice, the window length must be a whole number of samples, so the instantiated value of m is set as close as possible to the design value while ensuring an integral filter length. All subsequent calculations use the actual, rather than the desired, value of m.

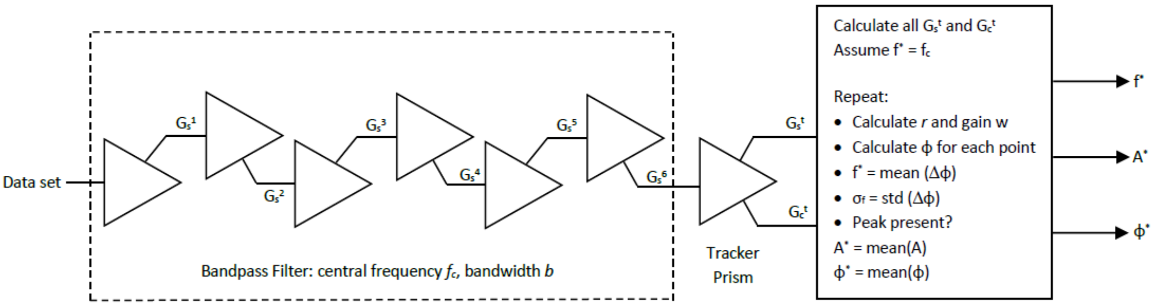

Figure 6. Prism signal processing scheme for spectral analysis

For example, Table 1 lists the Prism parameters to create a bandpass filter with f_c = 200 Hz, f_s = 10 kHz and b = 1.5 Hz. The total length of the filter is 12,992 samples.

Table 1. Bandpass filter parameters for f_c = 200 Hz, f_s = 10 kHz and b = 1.5 Hz

Prism Index, i	h^i	m^i (Hz)	length (samples)
1	30	6.793478	2944
2	30	6.648936	3008
3	20	10.080645	1984
4	20	9.765625	2048
5	15	13.586957	1472
6	15	13.020833	1536

The corresponding dual output tracker Prism uses h^t = 25 and m^t = 7.8125 Hz, resulting in a 2,560 sample filter. The data set is passed through the bandpass filter and tracker to generate reduced length time series outputs G_s^t (k) and $G_c^t(k)$ for k = 1 … N_k. For f_c = 200 Hz these time series have length 20,000 – 12,992 – 2,560, so N_k = 4,448 samples.

For this example, to search for low amplitude or hidden tones, a series of filter/trackers have been generated for the range $f_c \in$ [20, 1000] Hz in steps of b/4 i.e. 0.375 Hz. This small step size is not considered to be computationally optimal, but allows a clear illustration of the Prism technique.

3.2 Identifying true peaks

Given the set of G_s^t (k) and G_c^t (k) values generated by each filter/tracker, the mean frequency (f^*), amplitude (A^*) and phase (ϕ^*) of any spectral peak are to be determined. This task can be split into two stages: firstly, determining whether there is in fact a peak present over the frequency band of interest; and secondly, determining its properties. In this subsection, the first task is considered.

An obvious approach to determine whether a peak is present would be to apply some threshold to the magnitude/energy/amplitude of the tracker Prism outputs, and then to look for local maxima – this approach is typically used with FFT. However, despite the application of narrowband filtering, this approach remains susceptible to frequency leakage in the vicinity of a large peak. An alternative approach, which proves effective, is to first examine the calculated *frequency* of the output time series. The dual output tracking Prism provides a sin/cosine pair of values, from which a phase angle can readily be derived. The rate of change of this phase angle over consecutive samples provides an estimate of the frequency of the filtered signal. This is an advantage over DFT/FFT, which can only derive magnitude information at a set of discrete frequencies, fixed by the data set length (Figs 3 and 4).

For each pair G_s^t (k) and G_c^t (k), equations (3) and (4) constitute two equations in three unknowns: r, A, and ϕ. Assuming that r and A are constant (but initially unknown) over the data set, an iterative calculation can be applied to determine a best estimate of the mean frequency (f^*).

1. Initially, assume f^*=f_c, the central frequency.
2. Calculate the corresponding value of $r^* = f^* / m^t$
3. Calculate the relative gain between G_s^t and G_c^t at f^* (from equations (5) and (6):

$$w = r^* / h^t \tag{10}$$

4. For each of the N_k filtered data points, calculate the phase via the inverse tangent function:

$$\phi(k) = \text{atan2}(G_s^t(k), wG_c^t(k)) \tag{11}$$

5. From the time series $\phi(k)$ calculate a revised estimate of f^* as the average rate of change of phase, together with its standard deviation (σ_f), based on sample-by-sample step change estimates:

$$f^*_{est}\ (\mathrm{k}) = (\phi(k)) - \phi(\mathrm{k}\text{-}1)) \times fs\ /2\pi,\ k = 2, \ldots N_k \tag{12}$$

with appropriate handling of phase wraparound between ± π. The mean estimate of frequency is:

$$f^* = \sum_{k=2}^{N_k} f_{est}^*(\mathrm{k})/(N_k - 1) \tag{13}$$

The standard deviation is also calculated, and scaled as a percentage of the mean:

$$\sigma_f = \frac{100}{f^*}\sqrt{\sum_{k=2}^{N_k} (f_{est}^*(\mathrm{k}) - f^*)^2/(N_k - 1)} \tag{14}$$

6. The presence or absence of a peak in this region is determined based upon the values of f^* and σ_f, as discussed below. If no peak is present, the calculation terminates. If a peak is present, the calculation is repeated from step 2 using the revised value of f^* as the starting point to obtain a final best estimate of f^*.

Note that for this frequency calculation, the gain applied by the bandpass filter is common to both $G_s^t(k)$ and $G_c^t(k)$ and so does not affect the result. Similarly, the (constant) linear phase delays introduced by both the bandpass and tracking Prisms do not influence the calculated rate of change of phase.

Figure 7 shows the peak frequency obtained for each central frequency f_c after a first pass through the above calculation. The most striking feature of the data is that, for a wide range of centre frequencies (50 – 300 Hz) the observed peak frequency is often close to 200 Hz. In the absence of any genuine peak within the filter passband, the 200 Hz peak is being detected. This can be thought of as another form of frequency leakage, shown for the FFT in Figure 1. However, the peak frequency calculation serves as a means to distinguish between frequency leakage and a genuine tone. Other than the extended influence of the 200 Hz tone, the calculated peak frequencies show a considerable degree of scatter, with a higher density of points falling around the line $y = x$; this is likely to arise from bandpass filtering of the low levels of numerical noise within the calculation.

Figure 7 suggests a first test to identify the presence of a true tone (and therefore to reject frequency leakage and random scatter): for a true tone, the calculated peak frequency should fall within the passband of the associated filter/tracker, i.e. $f^* \in [f_c - b, f_c + b]$. Given that a systematic frequency sweep is being applied, there is no need to estimate (say) the 200 Hz component through the vantage point of a bandpass filter centred around (say) 160 Hz or 350 Hz – a better estimate will be produced by a filter centred at or near 200 Hz. Besides, a remote peak frequency may simply arise from low level random noise in the absence of a local peak. Requiring peak location to fall within the passband also promotes numerical stability. Beyond the passband, the filter gain becomes very small; applying gain compensation to any amplitude calculation may result in numerical instability.

The proximity test is illustrated in Figure 8, which shows the results from Figure 7 over the reduced region 0 – 400 Hz and with the line $y = x$ superimposed. Only a few points fall on or close to this line, including several in the low frequency region, and the two true peaks. Here the 234 Hz tone first emerges out of the spectral shadow of its far larger neighbour, as a potential true peak.

Figure 9 shows all the peaks that remain after the first test is applied, together with their corresponding amplitudes (as calculated in the next section). Already, the two true tones have the highest calculated amplitudes, but a number of false tones, attributable to random chance, remain. The frequency standard deviation calculated in equation (13) provides a second test: if a true tone is present, the estimated frequency value should be relatively steady on a sample-by-sample basis, and so the standard deviation should be small. Conversely, if a random signal has by chance generated an average frequency that falls within the filter bandwidth, it is likely that the corresponding frequency time series will show higher variation.

Figure 10 shows the frequency standard deviation calculated for all trackers up to 1000 Hz. As before, the 200 Hz region shows a clear structure arising from spectral leakage, with the smallest standard deviation at the true tone frequency. The peak at 234 Hz is also clearly indicated as a true tone. By contrast, the other potential peaks have relatively high standard deviations. Applying a simple threshold (say 10^{-2}), in addition to the first test, leaves only the two true tones identified (Figure 11).

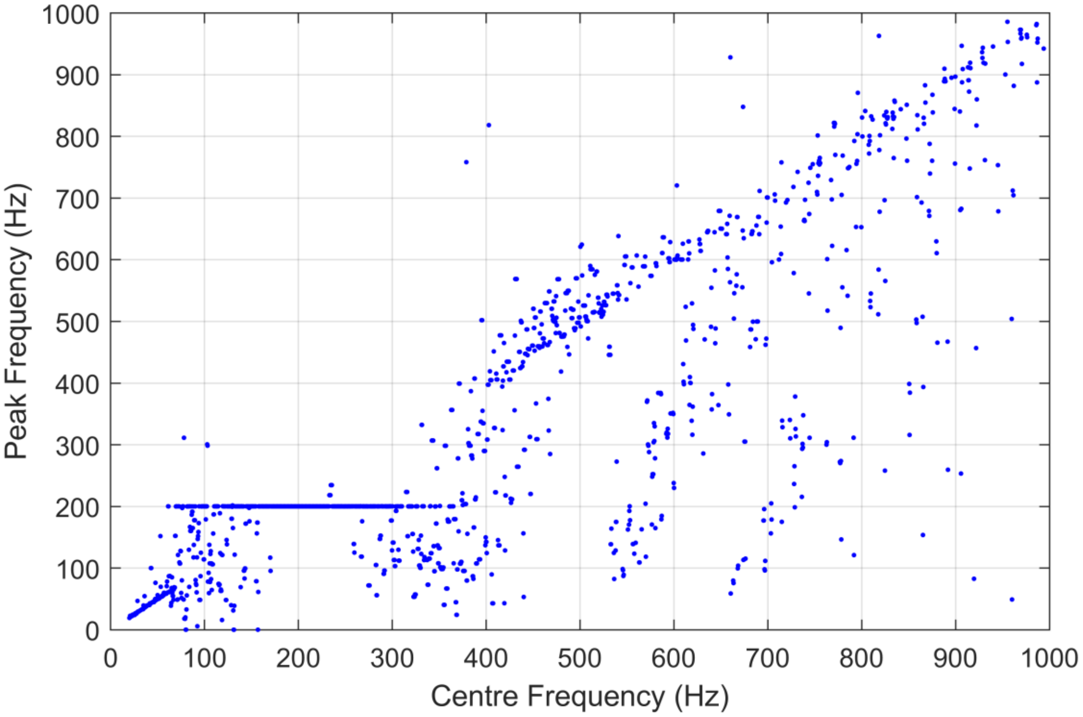

Figure 7. Peak frequency values from Prism spectrum analysis (up to 1000 Hz)

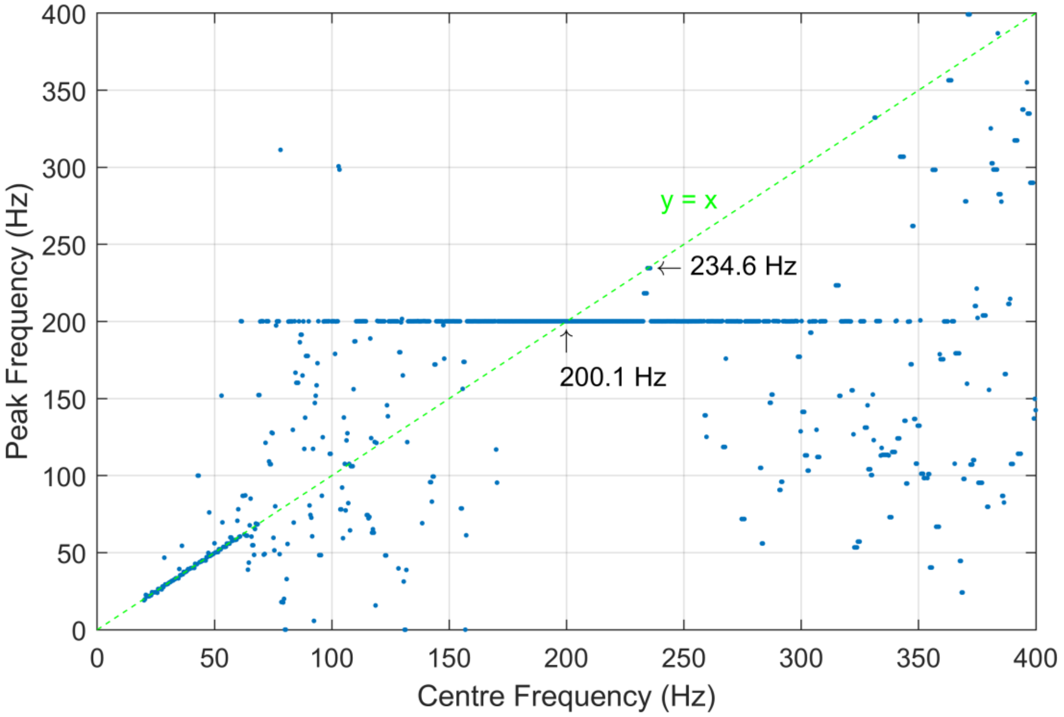

Figure 8. Peak frequency values from Prism spectral analysis (detail)

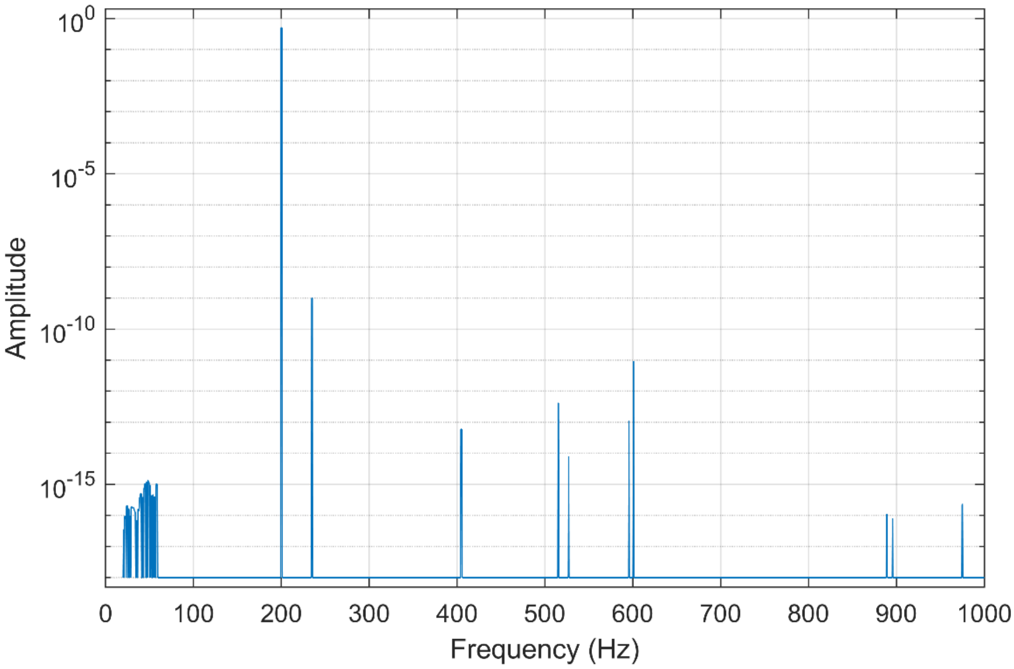

Figure 9. Peak frequency values falling within bandpass limits

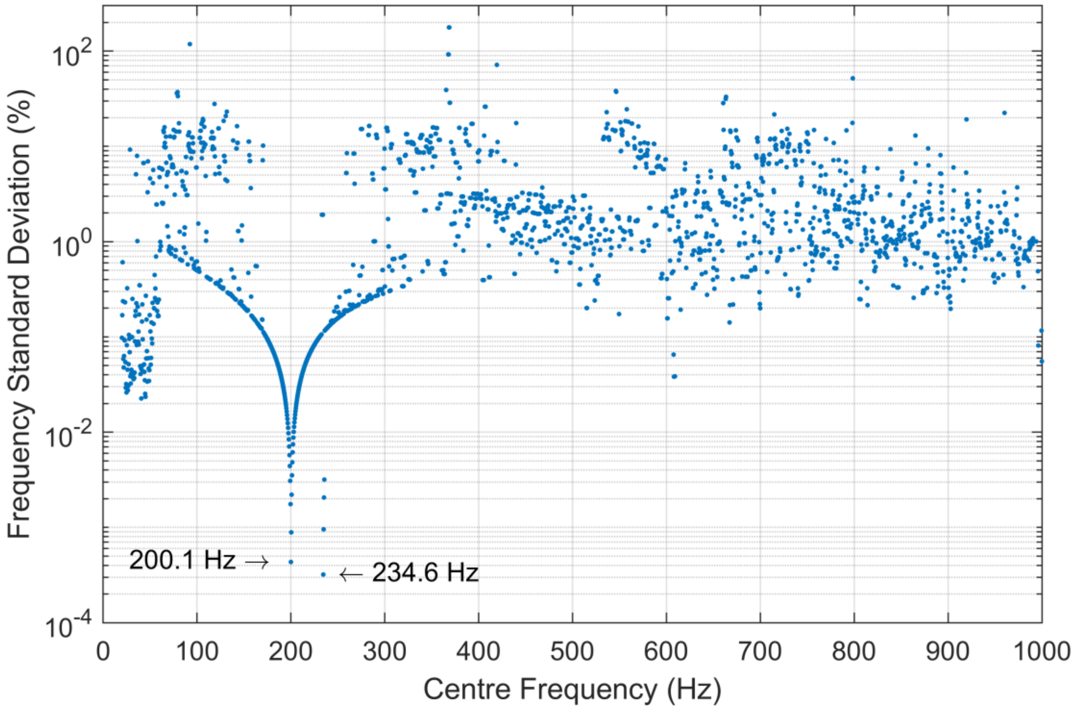

Figure 10. Peak frequency standard deviation

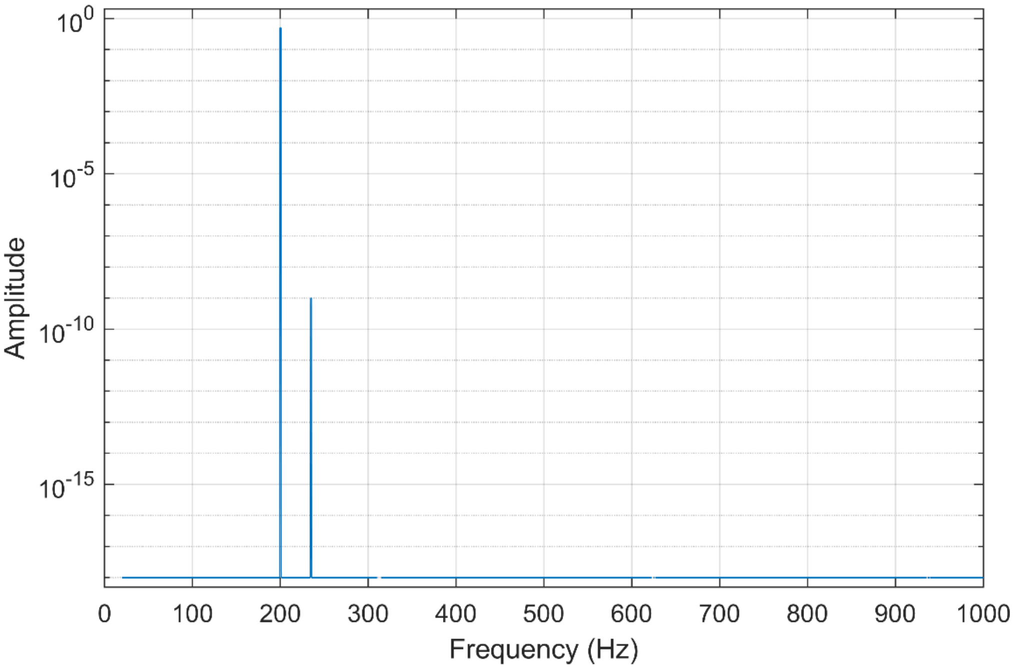

Figure 11. Identified true tones

3.2. Calculation of amplitude

Having identified each true tone frequency, the corresponding amplitude is calculated, through the following additional steps.

7. Calculate the bandpass filter gain Γ_{bp} at f^* as the product of the individual gains of each Prism:

$$\Gamma_{bp}(f^*) = \prod_{i=1}^{6} \Gamma_c(r^i, h^i), \text{ where } r^i = f^*/m^i \tag{15}$$

8. Calculate the tracker output gains at f^*

$$\Gamma_s(f^*) = \Gamma_s(r^t, h^t) \text{ and } \Gamma_c(f^*) = \Gamma_c(r^t, h^t), \text{ where } r^t = f^*/m^t \tag{16}$$

9. Calculate the sample-by-sample amplitude estimates based on the gain compensated $G_s^t\,(k)$ and $G_c^t(k)$ values, and their mean:

$$\mathrm{A(k)} = \sqrt{\left(\frac{G_s^t(k)}{\Gamma_s(f^*)}\right)^2 + \left(\frac{G_c^t(k)}{\Gamma_c(f^*)}\right)^2} \tag{17}$$

$$A^* = \frac{\sum_{k=1}^{N_k} \mathrm{A(k)}/(N_k)}{\Gamma_{bp}(f^*)} \tag{18}$$

Figures 12 and 13 show the resulting estimates of peak amplitude and frequency, including how these vary with data record length (compare Figures 3 and 4 for the FFT results). In the lowest plot, the right hand axis shows the number of periods of the peak frequency in the post-bandpass filter data set, and a corresponding value of δ is plotted on the left hand axis. There is significantly less variation with data set length for the Prism-based amplitude and frequency values, compared with those of the raw FFT values. However, while for the 200 Hz tone $\delta = 0$ results in the most accurate estimates of amplitude and frequency, the 234 Hz tone manifests some random drift; this may be attributed to the residual influence of the 200 Hz tone.

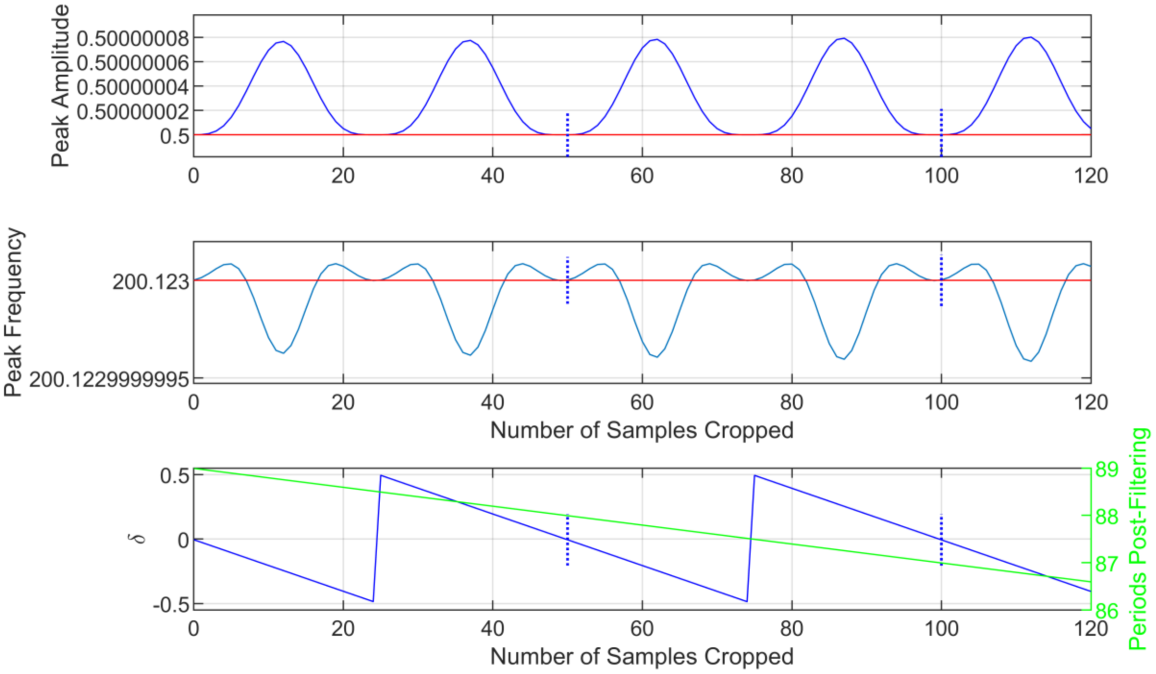

Figure 12. Peak frequency and amplitude around 200.123 Hz with varying dataset length.

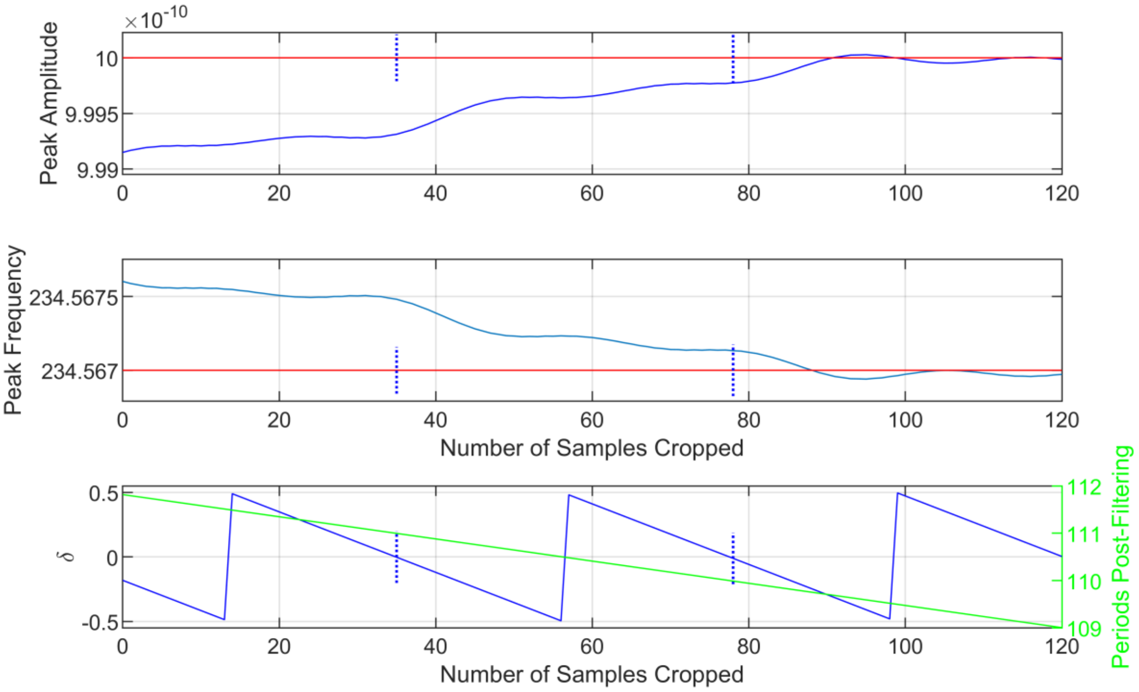

Figure 13. Peak frequency and amplitude around 234.567 Hz with varying dataset length.

3.3 Additional calculations

Two further improvements can be introduced into the calculations of frequency and amplitude, the first minor, and the second more substantial.

As suggested by figure 12, at least for strong and/or sufficiently separated tones, the amplitude and frequency estimates are improved when $\delta \approx 0$. Adjusting the data set length entails a complete recalculation for an FFT, but for the Prism calculation this simply involves truncating the data set used for calculating averages (i.e. the value of N_k) based on the value of f^*.

More substantially, if the parameters (frequency, amplitude and initial phase) of a strong tone can be determine with high accuracy, then a synthesized version of the tone can be removed from the original data set to assist in the tracking of weaker tones. This requires an accurate calculation of the initial phase of the tone in the data set, which in practice is straightforward. As all Prism calculations are linear phase, the average of the N_k phase values, generated in equation (11), yields the phase at the midpoint of the data set. Given the data set length and the tone frequency f^*, it is trivial to calculate the phase at the start of the data set from the phase at its midpoint.

Figure 14 shows a flow chart summary of the full calculation, including these additional two enhancements, while Figures 15 and 16 show the corresponding results. The parameters of both tones show reasonable independence to data length, and the errors are small. For the 200 Hz tone, the frequency error is approximately 1.6e-12 Hz and the amplitude error is approximately -4.4e-11. For the 234 Hz tone, the frequency error is approximately 1.8e-5 Hz and the amplitude error is approximately -2.9e-13. The lowest plot in each figure shows the value of δ generated as the data set length is varied. This is kept close to zero, resulting in low and stable error values.

4. Parameter Study

The numerical example in the previous section provides a step-by-step introduction to the Prism-based spectral analysis techniques proposed in this paper. Any claim for general applicability requires an exploration of a variety of parameters. For example, the FFT-based hidden tone detection technique described in [13] considers the following parameters: the distance between the frequency peaks (separated into the discrete number of FFT steps, and the value of δ), the initial phase of each peak, and the amplitude ratio between peaks. The number of peaks and of samples are kept constant.

A reduced parameter study is presented here. The number of peaks is kept at 2. The high amplitude ratio is retained. The distance between the two peaks is varied as follows. The major tone at 200.123 Hz is kept stationary, while the minor tone is moved from 180.623 Hz (19.5 Hz below the major tone) to 220.623 Hz (19.5 Hz above the major tone), in steps of 1 Hz, so that the closest gap between the two tones is 0.5 Hz. For each minor tone frequency step, 1000 simulations are performed where the initial phases of the two tones are selected at random. The calculated values for the amplitude, frequency and initial phase of the two tones are recorded and the root means square error (RMSE) is calculated over the 1000 trials. The results for the two tones are shown in Figures 17 and 18, where the x-axis gives the distance between the two tones scaled by the filter bandwidth (1.5 Hz).

The RMSE is expressed as a percentage of the true value for amplitude, in radians for the phase, and as a fraction of the filter bandwidth for frequency. For the first peak (Figure 17), the RMSE errors for all parameters are steady as long as the distance between the peaks exceeds approximately ± 2.5 bandwidths. Note this limit corresponds to the start of the transition region for the passband filter (Figure 1). Outside of this region, the errors are steady, with frequency at ~ 5e-12 bandwidths, phase at ~ 5e-11 radians, and amplitude ~ 1e-8 %.

Where the peak frequencies converge, the interference is greater, and the errors for the dominant peak increase by up to two orders of magnitude for phase and frequency, and one order of magnitude for amplitude. The errors for the second peak (Figure 18) are significantly higher and show greater variation. Outside the ± 2.5 bandwidth region, RMSE ranges are approximately as follows: frequency $\in$[5e-5, 5e-4] bandwidths, phase $\in$[5e-4, 5e-3] radians, amplitude $\in$[1e-2, 1e-1] %. Within the ± 2.5 bandwidth region, errors rise by between 3 and 4 order of magnitude. Further refinements of the technique are required to deliver improved estimates for this combination of extreme amplitude imbalance and very close peak proximity.

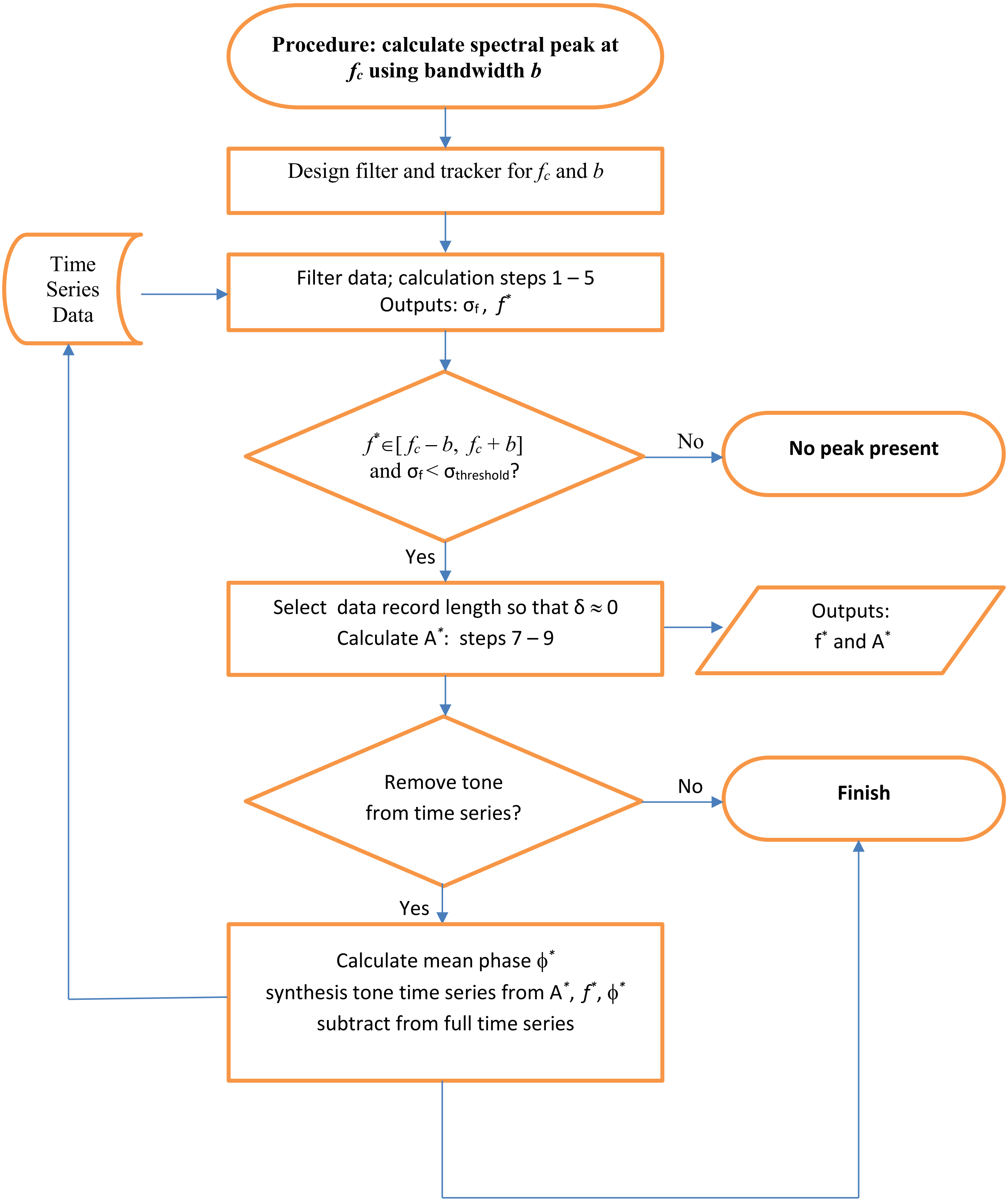
Procedure: calculate spectral peak at f_c using bandwidth b
Design filter and tracker for f_c and b
Time Series Data
Filter data; calculation steps 1 – 5
Outputs: σ_f, f^*
$f^* \in [f_c - b,\ f_c + b]$
and $\sigma_f < \sigma_{threshold}$?
No
No peak present
Yes
Select data record length so that $\delta \approx 0$
Calculate A^*: steps 7 – 9
Outputs:
f^* and A^*
Remove tone
from time series?
No
Finish
Yes
Calculate mean phase ϕ^*
synthesis tone time series from A^*, f^*, ϕ^*
subtract from full time series

Figure 14. Flow chart of complete Prism spectral analysis algorithm.

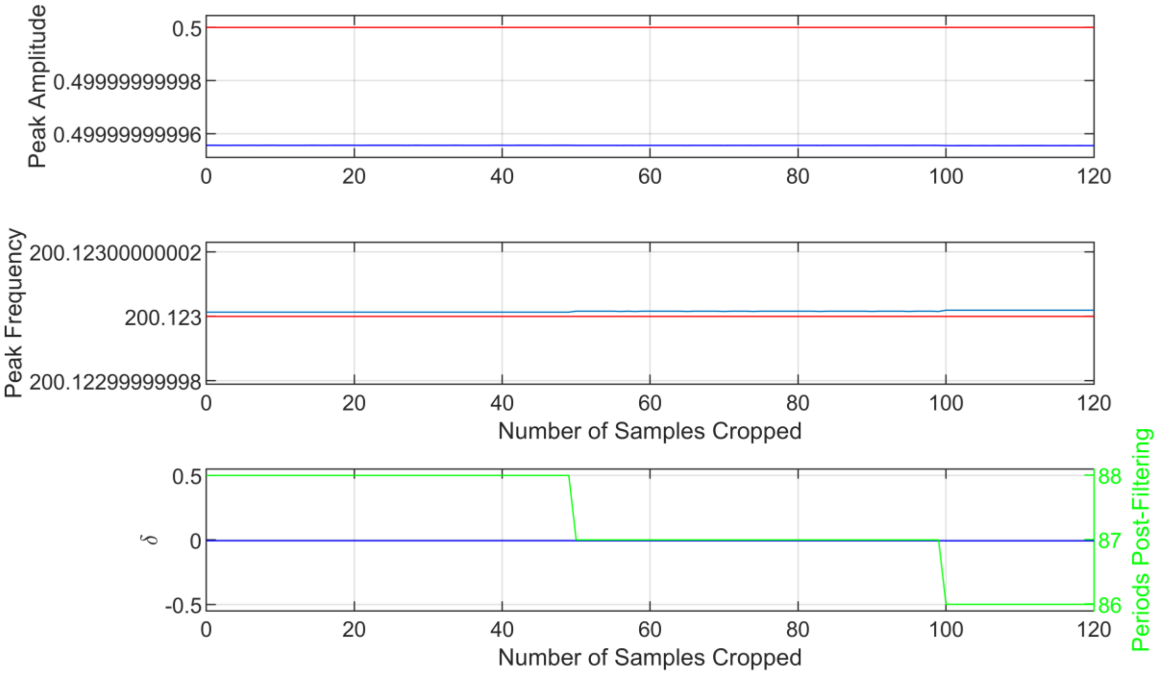

Figure 15. Prism peak frequency and amplitude around 200.123 Hz with varying dataset length.

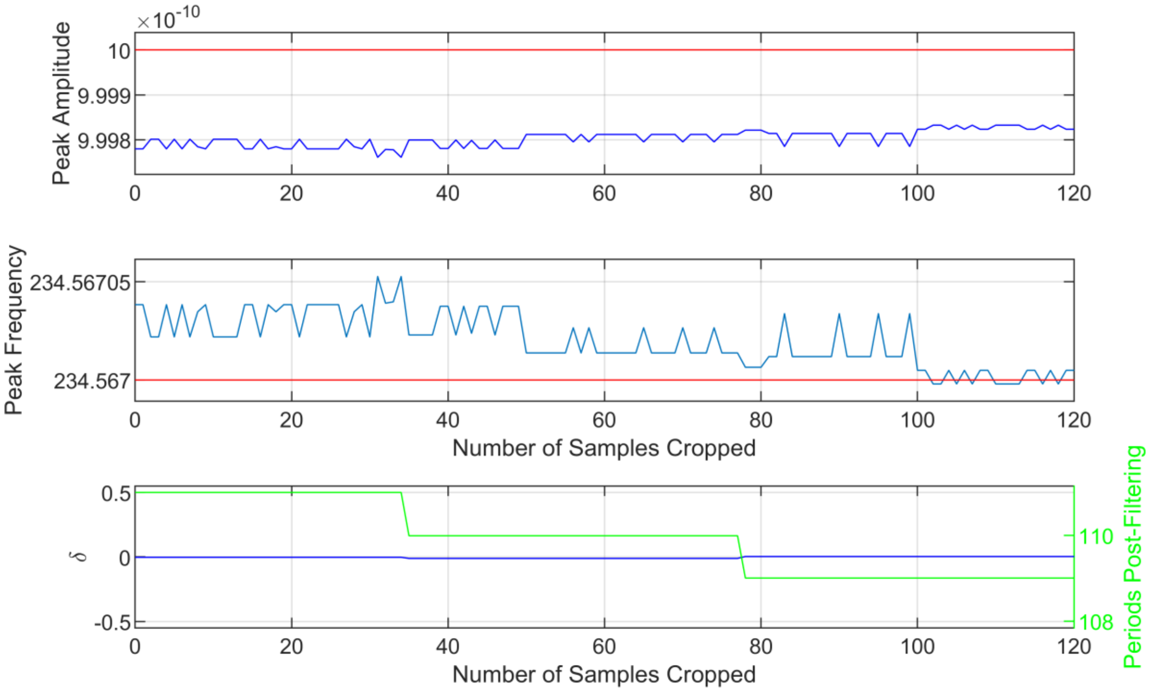

Figure 16. Prism peak frequency and amplitude around 234.567 Hz with varying dataset length.

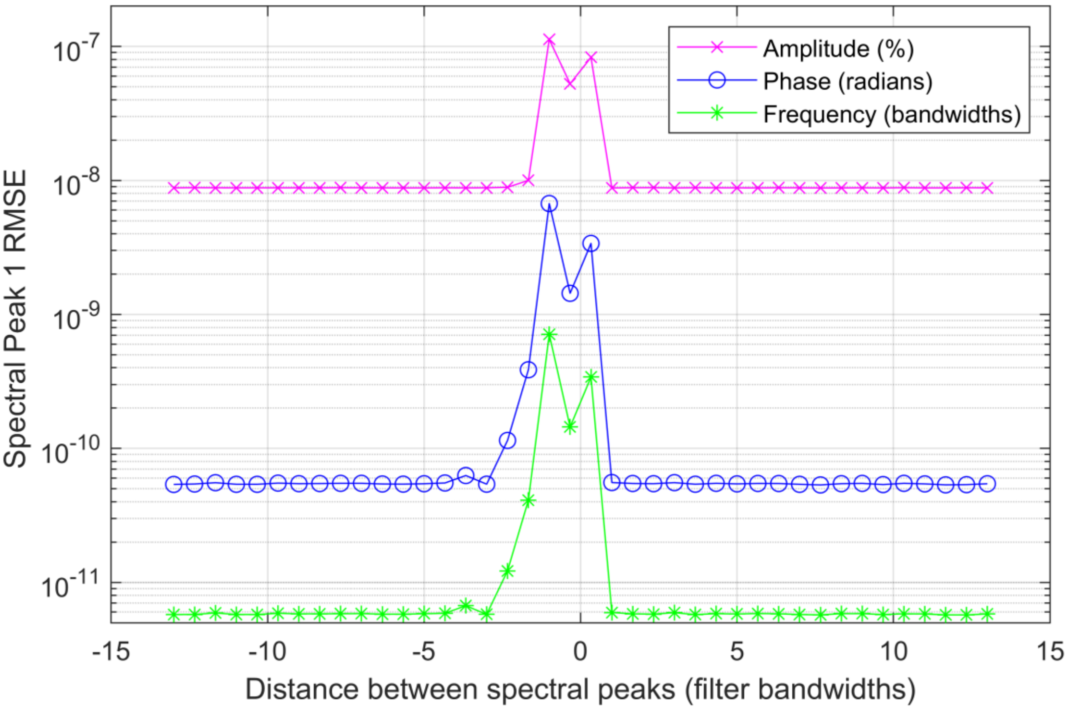

Figure 17. RMSE of major tone parameters with random phase and varying frequency gap

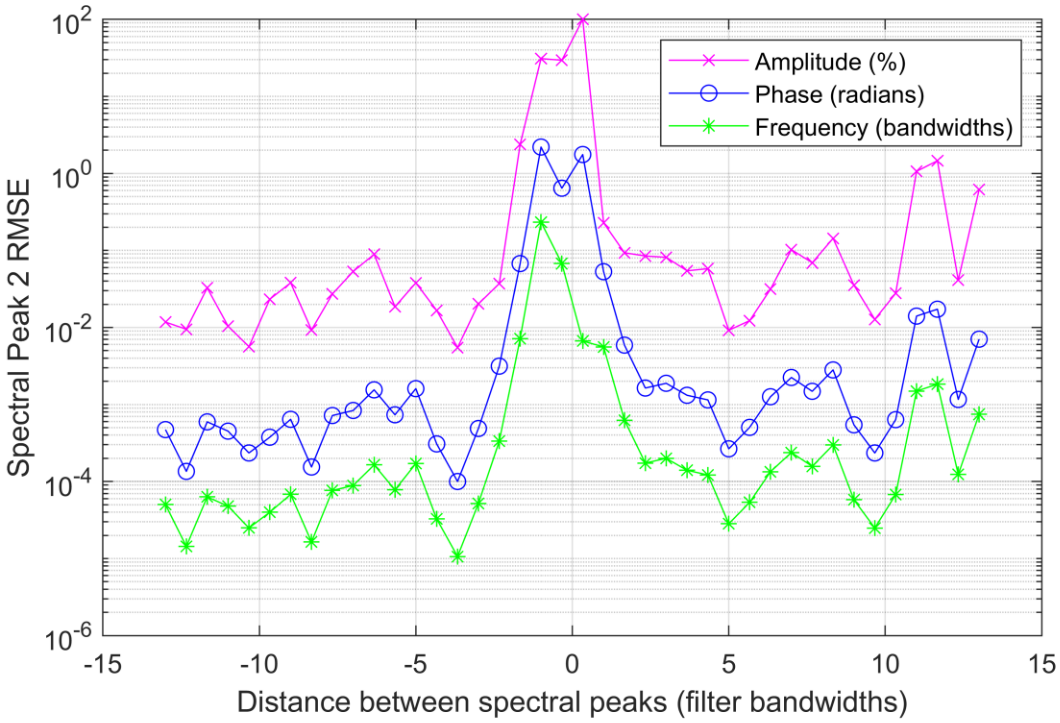

Figure 18. RMSE of minor tone parameters with random phase and varying frequency gap.

5. Discussion and Conclusions

Prism signal processing is a recursive FIR technique offering low design and computational cost compared with conventional convolutional forms of FIR filtering. This paper has demonstrated its potential as a tool for spectral analysis. The FFT calculates magnitude at fixed frequencies; the reported magnitude of any spectral peak depends upon its distance (δ) from adjacent FFT frequencies. By contrast, the Prism technique estimates frequency, (mean) phase and amplitude of the dominant tone within a passband. Frequency leakage is reduced by the use of narrowband filtering, but may also be identified and rejected based on the detected frequency for the current passband.

Prism-based tracking is inherently a real-time, sliding window algorithm, with a fixed computational cost, which, for a narrowband filter, this is approximately 300 floating point operations per sample. In instrumentation or other real-time applications, where the task is to track a specific peak within well-defined frequency limits (for example from a resonant transducer), a single tracker, of the form shown in Figure 4, can be an efficient FIR solution. Specifically, the computational cost remains fixed however narrow the bandpass filter ([8], [9]). When deployed to perform spectral analysis over a fixed data set as presented here, computational efficiency certainly cannot be claimed. For example, the one-time cost of calculating linearly-spaced sine and cosine values during Prism initialisation can be ignored for a continuous real-time operation, but is a significant cost in the analysis here. Accordingly, the technique is not presented as a blanket alternative to conventional FFT analysis. Rather, it may be used as a conceptually simple, supplementary, and relatively expensive tool to search for low amplitude and/or hidden tones that may not be observable using conventional FFT techniques.

The Prism technique may be of value applied directly to spectrally based measurements, for example in spectroscopy. However, it may also prove beneficial for a broader range of functions, such as condition monitoring [4, 5] and sensor validation [17]. Here, the provision of a precise and flexible spectral analysis of the full transducer bandwidth may give early indications of malfunction in the sensor or monitored process, and may further support fault diagnostics and measurement correction once a malfunction is detected.

The examples provided here use noise-free data, and double precision calculations are applied. These choices were made in order to demonstrate the technique’s high accuracy (especially through the application of Romberg Integration, which overcomes the errors arising from discrete time sampling), and its suppression of frequency leakage, as advantages over conventional FFT methods. The application of Romberg Integration to time series, particularly for Prism signal processing, will be described in detail in a future paper. With regard to noise, the basic Prism, and the bandpass filter derived from it, are robust to the introduction of noisy data and of reduced precision arithmetic. For example, [7] reports results for Prism operation based on single precision arithmetic, while [8] demonstrates good tracking of a sinusoid in white noise despite a signal-to-noise ratio of -60 dB.

A recently developed family of ‘low pass’ Prism filters [18] has the potential, when applied to spectral analysis, to provide a useful tradeoff between suppressing white (or other) noise vs spectral leakage. The filters have between 1 and 6 Prism ‘layers’, yielding stop band attenuation from -20 dB to -140 dB, where the filter length is proportional to the number of layers. For a given length of a spectral data set, using a smaller number of Prism layers will allow a narrower bandwidth *b* and therefore better (white) noise suppression; using a higher number of Prism layers will require a wider *b* but will deliver deeper stopband attenuation and therefore better suppression of adjacent peaks i.e. spectral leakage. The application of low pass Prism filtering to spectral analysis will be described in a future paper.

Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] Henry M, Leach F, Davy M, et al. 2017 *IEEE Ind Electron Mag.* **11**. URL 10.1109/MIE.2017.2760108

[2] Henry, MP 2018 *Measurement Techniques* URL 10.1007/s11018-018-1345-1

[3] Henry, MP, Bushuev, O, Ibryaeva, O. 2017 26th IEEE International Symposium on Industrial Electronics. URL 10.1109/ISIE.2017.8001451

[4] Henry M, Sinitsin V V. Proc - 2018 IEEE Ind Cyber-Physical Syst ICPS 2018. 2018:452-457. URL10.1109/ICPHYS.2018.8390747

[5] Henry M, Sinitsin V V. Proc - 2018 IEEE Ind Cyber-Physical Syst ICPS 2018. 2018:458-463. URL 10.1109/ICPHYS.2018.8390748

[6] Leach F, Karout S, Zhou F, Tombs M, Davy M, Henry M. 2017 *Flow Meas Instrum*. URL 10.1016/j.flowmeasinst.2017.09.009

[7] Henry, MP. IEEE Trans. Instr. Meas. URL 10.1109/TIM.2019.2916943

[8] Henry MP. 2018 2018 IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society.

[9] Owen J, Henry M. 2018 IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society. URL 10.1109/iecon.2018.8591619

[10] Duhamel P, Vetterli M. "Fast Fourier Transforms: A tutorial review and a state of the art. Signal Processing 19 (1990), 259-299.

[11] Aboutanios E, Mulgrew, B. "Iterative Frequency Estimation by Interpolation on Fourier Coefficients", IEEE Transactions on Signal Pocessing, 53, (2005), 1237 – 1242.

[12] Liguori C, Paolillo A. "Estimation of Signal Parameters in the Frequency Domain in the Presence of Harmonic Interference: A Comparative Analysis", IEEE Transactions on Instrumentation and Measurement, 55, (2006) 562-569.

[13] Liguori C, Paolillo A, Piignotti A. "IFFTC-Based Procedure for Hidden Tone Detection", IEEE Transactions on Instrumentation and Measurement, 56, (2007) 133-139.

[14] Belega D, Petri D, Dallet D. "Accurate amplitude and phase estimation of noisy sine-waves via two-point interpolated DTFT algorithms", Measurement 127 (2018) 89-97.

[15] Belega D, Petri D. "Influence of the noise on DFT-based sine-wave frequency and amplitude Estimators", Measurement, 137 (2019) 527-534.

[16] Fan L, Qi G, Xing J, Jin Jm Liu J, Wang Z, "Accurate Frequency Estimator of Sinusoid Based on Interpolation of FFT and DTFT", IEEE Access (2020) 44373 – 44380.

[17] Taymanov R, Sapozhnikova K, "Metrological self-check and evolution of metrology", Measurement, Volume 43, Issue 7, 2010, Pages 869-877.

[18] Henry, MP. "Low Pass, Low Cost Prism Filters", 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, Rome, June 2020.