Measurement 220 (2023) 113372

Contents lists available at ScienceDirect

Measurement

journal homepage: www.elsevier.com/locate/measurement

An ultra-precise Fast Fourier Transform

Manus Henry *

Fluid and Complex Systems Research Centre, Coventry University, UK
Department of Engineering Science, University of Oxford, UK

ARTICLE INFO

Keywords:
FFT
Prism signal processing
Spectral analysis
Spectral leakage
Hidden tones
Romberg integration

ABSTRACT

The Fast Fourier Transform (FFT) is a cornerstone of digital signal processing, generating a computationally efficient estimate of the frequency content of a time series. Its limitations include: (1) information is only provided at discrete frequency steps, so further calculation, for example interpolation, may be required to obtain improved estimates of peak frequencies, amplitudes and phases; (2) 'energy' from spectral peaks may 'leak' into adjacent frequencies, potentially causing lower amplitude peaks to be distorted or hidden; (3) the FFT is a discrete time approximation of continuous time mathematics. A new FFT calculation addresses each of these issues through the use of two windowing functions, derived from Prism Signal Processing. Separate FFT results are obtained by applying each windowing function to the data set. Calculations based on the two FFT results yields high precision estimates of spectral peak location (frequency), amplitude and phase while suppressing spectral leakage.

1. Introduction

The Fast Fourier Transform (FFT) has been described as "the most important numerical algorithm of our lifetime", used billions of times a day [1]. Improving FFT performance is a long-standing research topic [2], for example by seeking to improve the precision of frequency, amplitude and phase (FAP) estimates of spectral components, and/or by locating low amplitude 'hidden tones'. A recent paper [3] makes a first attempt at applying Prism Signal Processing (PSP) [4] to the FFT problem and provides a background discussion to the current work.

The Prism is a signal processing node which implements one or two linear phase FIR filters via a recursive, low-cost calculation, incorporating Romberg Integration (RI) [5] to deliver high precision outputs. Prism networks can be used to instantiate bandpass [6] and lowpass filters [7], as well as 'trackers' [7] which calculate the FAP content of a sinusoidal input. The Prism was conceived as an alternative to the (almost universally used) convolutional form of FIR filter, intended for real-time applications, providing sample-by-sample outputs for a time series input. By contrast, the FFT is usually viewed as a single calculation applied to a static data set, generating a vector of (complex) amplitude results for a linearly spaced set of frequencies. In [3], PSP is applied to spectral analysis by treating the data set as a time series. Narrow bandpass filters and trackers are created to locate spectral peaks and calculate the corresponding amplitude and phase values. While this approach can successfully determine spectral peak parameters to high precision, and detect 'hidden tones' (for example, locating a low amplitude peak 10^{-9} times smaller than an adjacent peak), the computational cost associated with the design and operation of many filters is much higher than the *nlog(n)* efficiency of the FFT.

This paper presents a more efficient synthesis of PSP and FFT [8], in which two Prism networks are converted into convolutional form and used as windowing functions for the data set. Applying the standard FFT calculation to both windowed data sets generates a pair of spectra from which peak locations can be resolved to high precision. This technique simultaneously addresses three FFT error sources: spectral leakage, the frequency discretization of the FFT outputs, and the fundamental issue of time discretization associated with any sampling system. Spectral leakage is suppressed via Prism windowing functions. Frequency discretization is addressed by using two windowing functions simultaneously, from which spectral peaks are readily located. Errors introduced by time discretization are addressed by incorporating RI into the Prism windowing function design.

In current practice the FFT calculation is employed over a wide range of applications, on data sets with different numerical properties. The Prism technique was developed in the context of instrumentation applications [9] where the spectra of interest typically exhibit low noise, high dynamic range (i.e. the spectral peaks have widely varying amplitudes) and moderate peak separation along the frequency axis.

* Address: Fluid and Complex Systems Research Centre, Coventry University, UK.
E-mail address: manus.henry@coventry.ac.uk.

https://doi.org/10.1016/j.measurement.2023.113372
Received 17 February 2023; Received in revised form 18 July 2023; Accepted 21 July 2023
Available online 23 July 2023

Accordingly, the simulation example used throughout this paper has these characteristics. It demonstrates that the Prism FFT can provide FAP precision up to several orders of magnitude higher than FFT methods using conventional windowing functions. However, it is acknowledged at the outset that in many practical applications, particularly where the FFT implementation has been optimized to match the numerical characteristics of the data, the Prism technique may be of more limited value. It is not possible, in a single paper, to provide performance comparisons against the full range of FFT-related methods. The intention here is therefore to introduce the basic Prism FFT technique, describing its advantages and limitations, so that it may be considered as a further option in the spectral analysis toolset. Accordingly, the paper has been written in a tutorial style aimed at a broad instrumentation readership, introducing the various issues and demonstrating the Prism-based solution step-by-step, with appropriate levels of mathematical detail at each stage. MATLAB code for an implementation of the Prism technique is provided in the supplementary material.

The paper is organized as follows. Section 2 introduces the simulation, illustrating the limitations of the conventional 'plain' FFT calculation, and the performance advantage offered by the Prism technique. Section 3 addresses spectral leakage and introduces windowing functions used in both conventional and Prism FFT techniques; conventional windowing reduces spectral leakage without matching Prism FFT performance for the simulation. Section 4 addresses the FFT frequency discretization issue and explains how the Prism FFT technique calculates peak frequency (and other parameters) to high precision without using interpolation. Section 5 describes how the Prism windowing functions are designed, addressing the sampling/time discretization issue through the use of Romberg Integration to provide high precision results. Section 6 demonstrates how the Prism FFT algorithm performs on the simulation example with varying levels of noise. Section 7 provides a final discussion and identifies areas for future development.

2. Limitations of the basic FFT calculation

In this section, basic FFT concepts and limitations are introduced and illustrated using a set of simulations based on a time series model. Fig. 1 shows an example of the time series and its spectral components. The time series consists of 48,000 values sampled at 48 kHz, with a duration of 1 s. It contains ten spectral components (hereafter referred to as tones or peaks) which have amplitudes at exact powers of 10 ranging from 1 Volt down to 1 nV. Components are spaced approximately 60 Hz apart, ranging between 9030 Hz and 9570 Hz, with low amplitude tones adjacent to high amplitude tones. The time series can be described as follows:

$$d(i) = n(i) + \sum_{k=0}^{9} A_k \sin(2\pi f_k t(i) + \varphi_k), i = 0 \cdots 47,999 \quad (1)$$

where:

$n(i)$ is additive white noise with a specified standard deviation;

A_k = [1e-4, 1e-7, 1e-2, 1e-9, 1e0, 1e-8, 1e-1, 1e-6, 1e-3, 1e-5] are the tone amplitudes (V);

$f_k = 9030 + 60k + \delta_k$ are the tone frequencies (Hz), with.$\delta_k \in [-0.5, 0.5]$

$t(i) = i/f_s$, is time, with sample rate f_s = 48 kHz.

φ_k are the initial phases (radians) for the tones, with $\varphi_k \in [-\pi, \pi]$.

Throughout the paper, these tones are indexed either in frequency order from lowest to highest with the index $k = 0, 9$; or, more usually, in amplitude order from highest to lowest with index $j = 1,10$. While the amplitude A_k of each tone is constant for all simulations, the initial phase of each tone φ_k is selected at random, and white noise with a specified standard deviation is added to each sample. Furthermore, the frequency of each tone may be offset by δ_k, as discussed further below.

The characteristics of this data have been selected to match the instrumentation interests of the author, while highlighting some of the limitations of conventional FFT calculations. The near proximity of spectral tones with high dynamic range (1:10^9, matching that demonstrated in [3]) combined with low noise, offers a good demonstration of the issues addressed by the Prism FFT technique.

Applied to this 1 s data set, the FFT algorithm generates complex amplitude results for frequencies from 0 Hz to 23,999 Hz at 1 Hz intervals. Throughout the paper, only results in the range 9000—9600 Hz are presented, but there are no spectral range limitations for the techniques described. The accuracy of any FFT calculation is sensitive to the distance between each true tone and the nearest FFT reporting frequency. The parameter δ [3] may be used to describe this distance, scaled by the FFT reporting interval, taking a value in the range [-0.5, 0.5]. When $\delta = 0$ the peak frequency coincides exactly with an FFT reporting frequency; with $\delta = \pm 0.5$, the peak falls exactly halfway between two adjacent FFT reporting frequencies. For the simulation example used here, the FFT reporting interval is 1 Hz, so that the δ_k term in eqn (1) gives the numerical value of both the offset from the nearest integer frequency value in Hertz, and well as the proportional offset from the nearest FFT reporting frequency. For example, if the true tone frequency of the first component ($k = 0$) is 9030.123 Hz, then $\delta_{k=0} = 0.123$.

The term 'bin' is often used to describe the regularly spaced FFT results (see for example [13]), based on the analogy that the spectrum

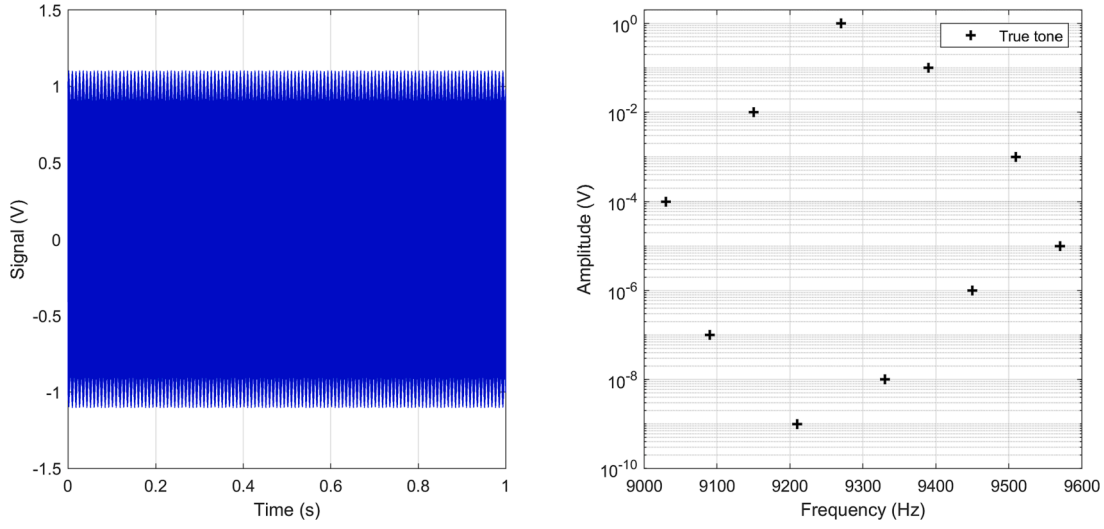

Fig. 1. Simulated data for FFT analysis: (left) time series; (right) spectral components.

has been partitioned into a discrete series of 'bins'. For brevity, the terms 'bin' and 'bin frequency' will be used here to refer to the discrete set of results provided by the FFT.

Fig. 2 shows a typical FFT spectrum obtained (using the built-in MATLAB function *fft)* when $\delta = 0$ for all tones. This condition is achieved (eqn. (1) by setting the true tone frequencies to be exactly 9030 + 60 k Hz, for $k = 0 \ldots 9$, and so can be referred to as $\delta_{\forall k} = 0$ or equivalently $\delta_{\forall j} = 0$. The initial phase of each true tone φ_k is randomly selected from the range [-π, π] radians, while white noise $n(i)$ is added to the time series with standard deviation 1e-10 V. The results obtained are labelled 'Plain' FFT as no additional conditioning (such as the use of a windowing function) has been applied before invoking the standard FFT algorithm.

In Fig. 2, the noise floor between the peaks is low, at approximately 1e-12 V, all peaks are clearly visible in the spectrum, and the location of each true tone appears well matched by a corresponding FFT peak. This is confirmed in Table 1, which summarizes results obtained from 10,000 random simulations with $\delta_{\forall j} = 0$. For each true tone, the corresponding root mean square error is given for the FFT-based estimates of frequency, amplitude and (mid-dataset) phase. These results are obtained by the simple procedure of selecting the highest local peak in the FFT spectrum, using its bin frequency as the estimated peak location, and using the standard root sum square and arctan calculations (via the MATLAB functions *abs* and *angle*) to obtain corresponding values of amplitude and phase.

The RMS errors are converted to percentage values (providing an indication of relative accuracy) on the following basis. The amplitude RMS error is expressed as a percentage of the true tone amplitude. The frequency error is expressed as a percentage of the FFT reporting interval i.e. 1 Hz, while the phase error is expressed as a percentage of 2π radians.

For this ideal case, the errors are low. The frequency errors are, by definition, zero, as with $\delta_{\forall j} = 0$, each bin frequency coincides exactly with the corresponding true tone frequency. The amplitude and phase errors are both proportional to the signal-to-noise ratio i.e. the ratio of the peak amplitude to the white noise standard deviation. The amplitude error rises from 6.4e-11 % to 6.4e-2 %, while the phase error rises from 1.0e-11 % to 1.0e-2 %, as the peak amplitude drops from 1 V to 1 nV.

In practice, the condition $\delta_{\forall j} = 0$ is almost never achievable, particularly for multitone signals: it requires that all tone frequencies are known in advance and that the data record length can be selected to contain an exact whole number of waveforms for every tone. A more realistic evaluation of FFT performance is obtained under conditions of 'non-coherent sampling' [17], where the value of each δ_j may vary within the full range [-0.5, 0.5]. The simulation is readily extended to provide this condition by randomly selecting a frequency offset for each tone from the range [-0.5, 0.5] Hz; this simulation case is referred to as $\delta_{\forall j} \neq 0$. Fig. 3 shows a typical FFT spectrum calculated from such data

Table 1
Plain FFT root mean square error results from 10,000 simulations with $\delta_{\forall j} = 0$.

Tone Index (j)	True Amp (V)	True Freq (Hz)	RMS Error		
			Freq (%)	Amp (%)	Phase (%)
1	1.0e-00	9270.0	0.000e + 00	6.407e-11	1.026e-11
2	1.0e-01	9390.0	0.000e + 00	6.444e-10	1.021e-10
3	1.0e-02	9150.0	0.000e + 00	6.512e-09	1.030e-09
4	1.0e-03	9510.0	0.000e + 00	6.474e-08	1.029e-08
5	1.0e-04	9030.0	0.000e + 00	6.433e-07	1.025e-07
6	1.0e-05	9570.0	0.000e + 00	6.519e-06	1.033e-06
7	1.0e-06	9450.0	0.000e + 00	6.443e-05	1.009e-05
8	1.0e-07	9090.0	0.000e + 00	6.465e-04	1.024e-04
9	1.0e-08	9330.0	0.000e + 00	6.376e-03	1.036e-03
10	1.0e-09	9210.0	0.000e + 00	6.436e-02	1.023e-02

while Table 2 provides the corresponding error analysis for 10,000 simulations.

In Fig. 3 the noise floor is significantly higher than in Fig. 2, rarely dropping below 1e-3 V. As a consequence, all tones with amplitudes below 1e-3 V are 'hidden' i.e. not directly observable as peaks within the FFT spectrum. Where peaks are observable, the FFT spectrum amplitudes do not appear to accurately match the corresponding true tone amplitudes, even on the logarithmic scale of Fig. 3. For example, the FFT spectrum peak value is clearly lower than the true tone amplitude at 1 V and 1e-2 V.

Table 2 confirms the relatively poor accuracy of the Plain FFT when $\delta_{\forall j} \neq 0$. To simplify the analysis, results are based on the bin value closest to the true tone frequency, including for hidden tones. Accordingly, the frequency RMS error is approximately 29 % for all tones, the amplitude RMS error is at least 17%, and all errors are several orders of magnitude higher than for the ideal, $\delta_{\forall j} = 0$, case presented in Table 1.

A variety of techniques have been developed to improve the accuracy of FFT peak estimation [10 –20], including locating hidden peaks [16]. Windowing functions are commonly used to reduce spectral leakage, as discussed in the next section. Other techniques include zero padding (e.g. in [10;13]) to reduce the FFT frequency reporting interval, and/or interpolation techniques around a detected peak [10,14,16–20]. Particular adaptations of the basic FFT method may be adopted based on

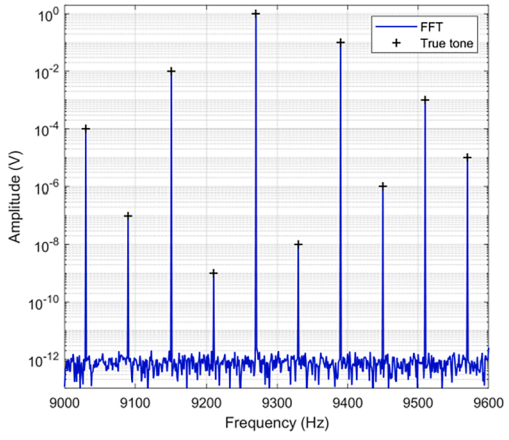

Fig. 2. Plain FFT spectrum with $\delta_{\forall j} = 0$.

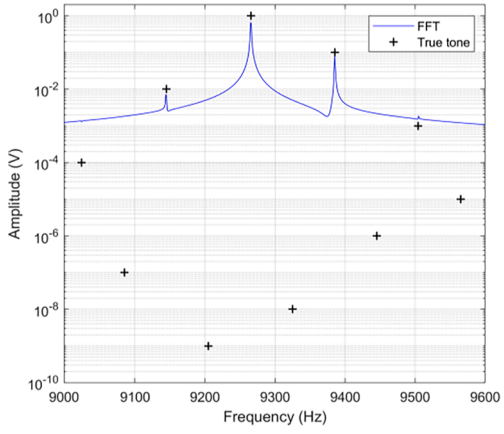

Fig. 3. Plain FFT results with each δ_j randomly selected from the range [-0.5, 0.5].

Table 2
Plain FFT root mean square errors for 10,000 simulations with each δ_j randomly selected from the range [-0.5, 0.5].

Tone	True Amp (V)	True Freq (Hz)	RMS Error		
Index (j)			Freq (%)	Amp (%)	Phase (%)
1	1.0e-00	9270.0	2.891e + 01	1.694e + 01	2.531e-03
2	1.0e-01	9390.0	2.902e + 01	1.698e + 01	2.483e-01
3	1.0e-02	9150.0	2.890e + 01	2.062e + 01	2.510e + 00
4	1.0e-03	9510.0	2.896e + 01	5.761e + 01	1.595e + 01
5	1.0e-04	9030.0	2.910e + 01	8.542e + 02	2.711e + 01
6	1.0e-05	9570.0	2.880e + 01	7.512e + 03	2.868e + 01
7	1.0e-06	9450.0	2.881e + 01	1.304e + 05	2.888e + 01
8	1.0e-07	9090.0	2.904e + 01	1.252e + 06	2.878e + 01
9	1.0e-08	9330.0	2.852e + 01	3.767e + 07	2.888e + 01
10	1.0e-09	9210.0	2.904e + 01	3.751e + 08	2.895e + 01

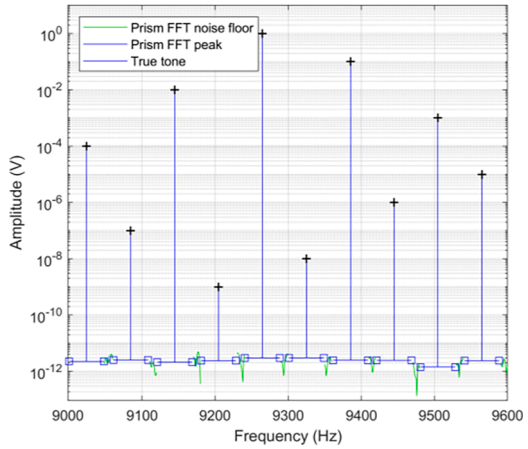

Fig. 4. Prism FFT results with each δ_j randomly selected from the range [-0.5, 0.5].

the specific characteristics of the dataset, windowing function and/or accuracy requirements for a given application.

Fig. 4 shows the Prism FFT obtained for the same data set as Fig. 3, i.e. for $\delta_{\forall j} \neq 0$. These results appear closer to those of Fig. 2 than of Fig. 3, featuring a low noise floor and good matching of all true tones. A distinctive feature of the Prism FFT is the 'dead zone' around each identified peak – shown in blue and terminated by squares – superimposed on the conventional noise floor which is plotted in green. Table 3 shows the corresponding RMS error analysis, which confirms the high precision of the Prism FFT. Both the amplitude and phase errors are only ~ 2.5 times larger than those in Table 1, and many orders of magnitude smaller than those in Table 2. In Table 1 the frequency errors are by definition zero, while in Table 3 the Prism FFT technique generates high precision results, ranging from 2e-10 % for the highest amplitude peak down to 2e-1 %. Another similarity between Tables 1 and 3 is that the amplitude, phase (and frequency) errors are proportional to the signal/noise ratio for each peak.

Accordingly, based on this example, the Prism FFT technique can be characterised as providing FAP results with a precision similar to the idealised (and rarely practical) case $\delta_{\forall j} = 0$, for datasets where the δ_j can take any value. As explained below, this is achieved using a simple algorithm with a computational cost approximately twice that of a conventional FFT using a windowing function.

An important caveat to this comparison is to note that the Prism method is more constrained with regard to the distance between spectral tones: as suggested by the dead-zone spacing in Fig. 4, the tones need to be sufficiently separated (depending on the Prism filter design: here the required gap is approximately 48 Hz), whereas for the Plain FFT results in Fig. 2, if $\delta_{\forall j} = 0$, the tones need only be spaced 1 Hz apart for the same precision to be achieved. This and other limitations are explained and discussed later in the paper.

3. Windowing functions: Addressing spectral leakage

The phenomenon of spectral leakage, and the use of windowing functions to address it [11], is illustrated in Fig. 5. The left-hand plot shows a typical time series of the simulation example for $\delta_{\forall j} \neq 0$, where the start and end points, marked with red circles, have distinct values. Note that if $\delta_{\forall j} = 0$, then the start and end values will, excluding additive noise, be identical. In mapping from the time domain to the frequency domain, the FFT calculation assumes an infinitely repeating times series. A discontinuity between the start and end of the time series creates an impulse, generating broadband noise in the corresponding FFT output, as illustrated in the difference between the noise floors in Figs. 2 and 3. Windowing functions improve FFT results by suppressing frequency leakage. The right-hand plot of Fig. 5 shows the effect of applying the Hann windowing function to the data set on the left, which results in the start and end values becoming identically zero, thereby suppressing any discontinuity in the original data set.

A variety of windowing functions have been developed, typically constructed as sums of weighted cosines; these include the widely used Hann, Hamming and Blackman windows. Harris [11] and Nuttall [12]

Table 3
Prism FFT root mean square errors from 10,000 simulations with each δ_j randomly selected from [-0.5, 0.5].

Tone	True Amp (V)	True Freq (Hz)	RMS Error		
Index (j)			Freq (%)	Amp (%)	Phase (%)
1	1.0e-00	9270.0	2.479e-10	1.729e-10	5.016e-11
2	1.0e-01	9390.0	2.234e-09	1.632e-09	2.629e-10
3	1.0e-02	9150.0	2.229e-08	1.642e-08	2.571e-09
4	1.0e-03	9510.0	2.247e-07	1.642e-07	2.573e-08
5	1.0e-04	9030.0	2.237e-06	1.636e-06	2.624e-07
6	1.0e-05	9570.0	2.244e-05	1.620e-05	2.580e-06
7	1.0e-06	9450.0	2.224e-04	1.637e-04	2.579e-05
8	1.0e-07	9090.0	2.207e-03	1.648e-03	2.603e-04
9	1.0e-08	9330.0	2.213e-02	1.602e-02	2.626e-03
10	1.0e-09	9210.0	2.227e-01	1.620e-01	2.609e-02

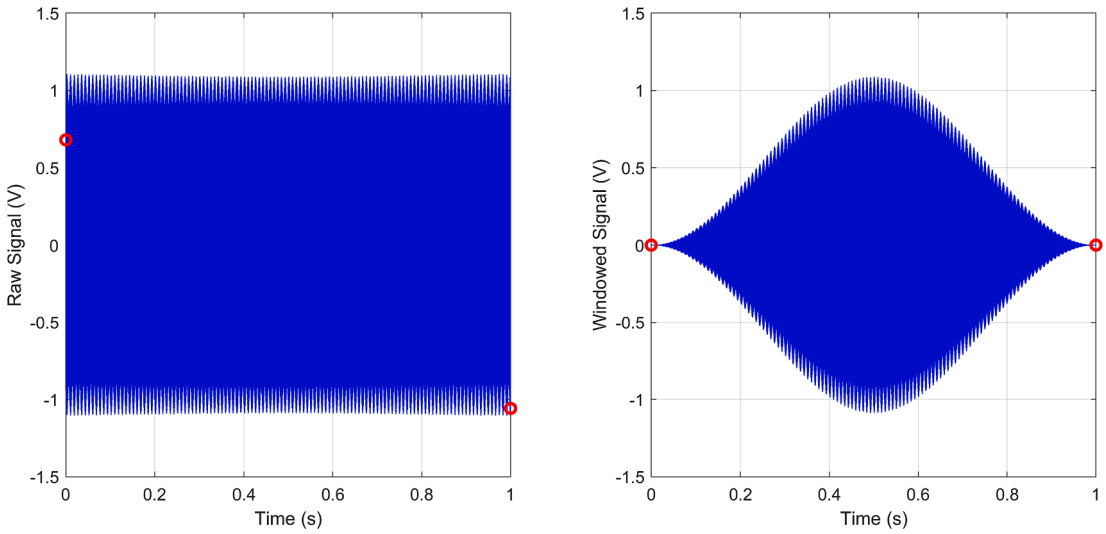

Fig. 5. (left) Time series (with $\delta_{\forall j} \neq 0$) with start and end samples marked with red circles; (right) the same time series after the application of the Hann windowing function.

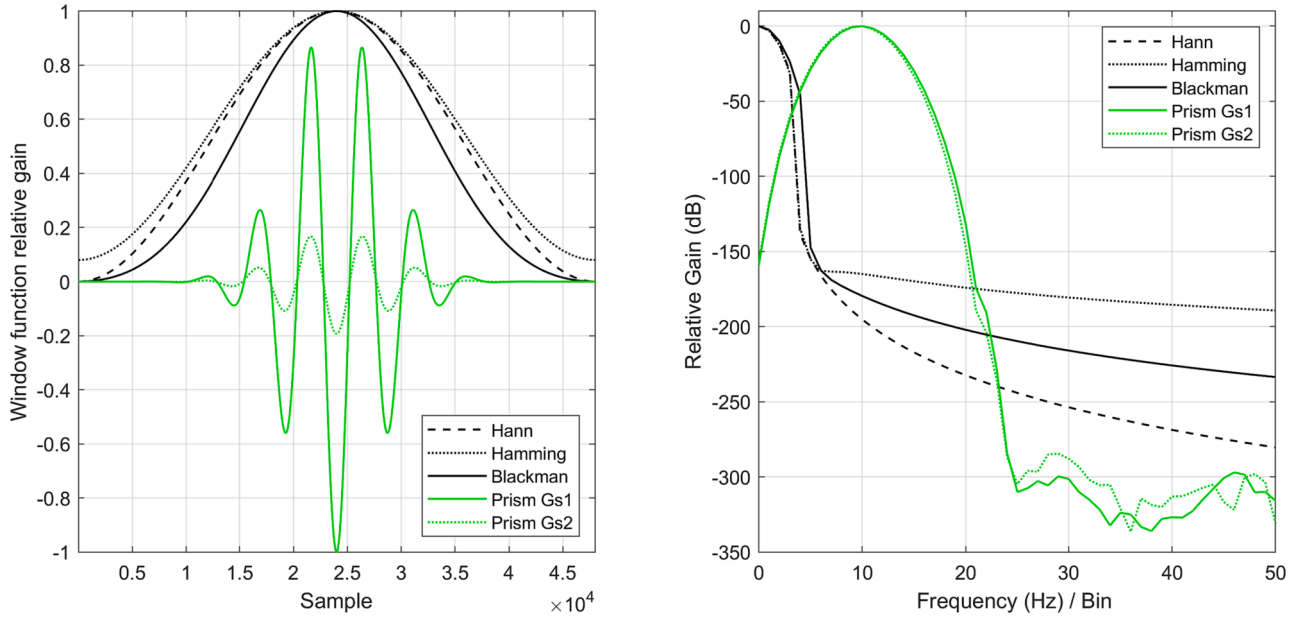

Fig. 6. Windowing functions: (left) as a function of window sample number; (right) frequency spectra.

analyze the properties of these and similar windowing functions. Harris notes the unavoidable "processing loss" associated with using windowing functions (as demonstrated below). Belega and Dallet [19] apply cosine windowing functions, "even though windowing increases the effect of wideband noise on the estimated parameters", while Ofelli and Petri [25] note that "the penalty required to reduce the spectral interference by means of windowing is an increase in the variance of the estimated parameters".

Harris further claims that calculating the "maximum dynamic range of multitone detection requires the [Fourier] transform of the window to exhibit a highly concentrated central lobe with very low sidelobe structure", and this approach, via the use of weighted cosine windows (WCWs), has become established practice.

Fig. 6 shows the Hann, Hamming and Blackman windowing functions implemented as 48 k sample windows, along with the Prism windowing functions *Gs1* and *Gs2*, described later. The weighted cosine windows (WCWs) have broadly similar properties, reflecting the recommendations of Harris, each consisting of a single lobe in the time domain with a non-zero mean value. In the frequency domain the WCWs gains drop rapidly from a peak at zero Hz. The Prism windowing functions have distinct characteristics, diverging from the Harris recommendation (and most conventional windowing functions). The Prism windows each consist of a zero mean multilobe function in the time domain, resulting in a relatively wide bandpass filter with peak frequency at approximately 9.5 Hz, and deep stopband attenuation. A windowing function with a peak gain that is offset from zero Hz creates unusual properties for the corresponding FFT, as demonstrated below.

Figs. 7 and 8 show typical results obtained by applying the WCWs to the simulation example for the cases $\delta_{\forall j} = 0$ and $\delta_{\forall j} \neq 0$ respectively, comparable to Figs. 2 and 3 for the Plain FFT. For the realistic $\delta_{\forall j} \neq 0$ case, spectral leakage is significantly reduced (compare Fig. 8 with Fig. 3), with only the lowest two peaks hidden in all three spectra. However, a comparison of Figs. 7 and 2 suggests that the ideal results obtained for $\delta_{\forall j} = 0$ are not preserved using WCWs – the noise floor is

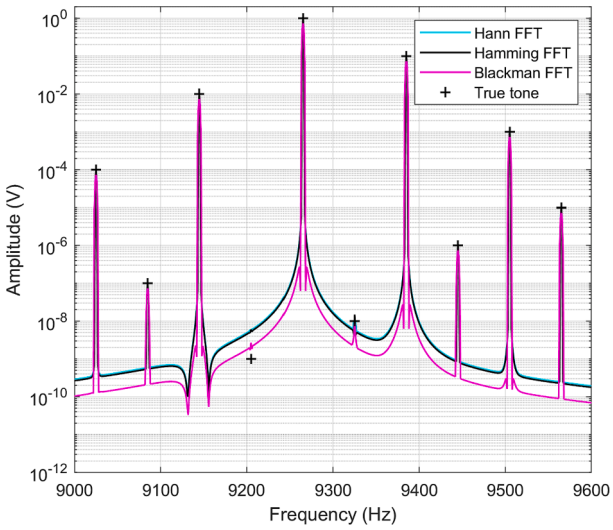

Fig. 7. Typical WCW FFT results with $\delta_{vj} = 0$.

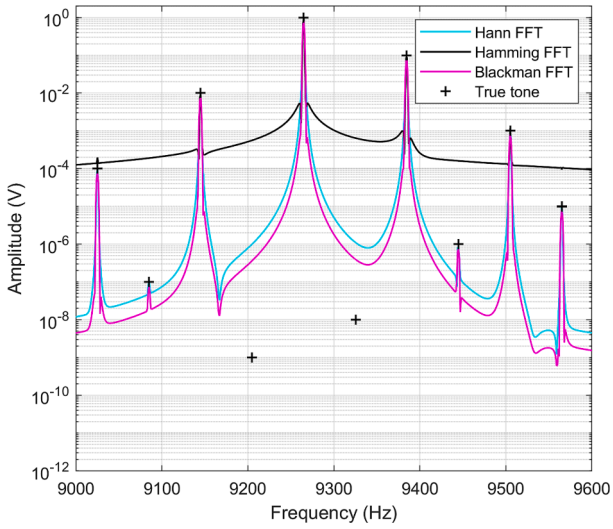

Fig. 8. Typical WCW FFT results with $\delta_{vj} \neq 0$.

higher in Fig. 7 while the lowest peak remains hidden. These observations are confirmed in Tables 4 and 5 which show the RMS errors for amplitude and phase obtained from 10,000 random simulations. Note that results for frequency are not shown – these are identical to those for the Plain FFT in Tables 1 and 2 respectively, as the same simple procedure of selecting the bin frequency with the highest local amplitude has been used.

Considering the $\delta_{vj} \neq 0$ results (Table 5) first, for those tones not hidden in the Plain FFT results (Table 2) there are modest reductions in error, while there are significant improvements for the previously hidden tones. At the highest amplitude, the RMS errors show reductions by a factor of 2 – 3 for amplitude and 5 – 10 for phase. The WCW methods deliver approximately constant amplitude error over several orders of magnitude (Hann and Blackman down to Tone 6, Hamming down to Tone 3), reflecting the reduced noise floors in Fig. 8, hence significantly reducing error compared with the Plain FFT. For phase, both the Hann and Black windows deliver an almost constant error over several orders of magnitude of peak amplitude. Overall, the WCW methods improve upon the $\delta_{vj} \neq 0$ results of the Plain FFT, particularly by reducing the noise floor so that more peaks are revealed.

Overall, the Blackman window performs best for this simulation example while the Hamming window performs least well and is not considered further.

For the $\delta_{vj} = 0$ results shown in Table 4, the introduction of the windowing functions leads to an increase in amplitude and phase errors of at least two orders of magnitude compared to the idealized Plain FFT results of Table 1. This illustrates Harris's "processing loss" [11,19,25]: the reduction of spectrum-wide leakage (beneficial for the almost universal $\delta_{vj} \neq 0$ case) is obtained at the cost of introducing localized distortion around each peak (observable when $\delta_{vj} = 0$).

It is important to further note that the WCW $\delta_{vj} = 0$ errors in Table 4 are, in all cases, larger than the corresponding Prism FFT errors for $\delta_{vj} \neq 0$ (Table 3). In other words, the errors introduced by conventional windowing functions in the ideal case (where all peaks coincide exactly with FFT reporting frequencies) are larger than those of the Prism technique where peaks may occur at any frequency. For the highest amplitude, the Prism errors are approximately 10 times smaller than those for Blackman; for the midrange amplitudes the error ratio is approximately 100; while the Prism algorithm successfully tracks the two lowest peaks that are hidden to all the WCW methods.

Further insight into the relationship between δ and FFT error, and the effects of windowing, are provided in Figs. 9 - 12. These show the variation in amplitude and phase errors for the highest amplitude tone, labelled Tone 1, as its δ value reduces exponentially from 0.5 towards zero. Note that only positive values of δ are considered here. As shown below, the Prism FFT errors are essentially independent of the value of δ, including its sign. Errors for the conventional methods are influenced by the sign of δ, due to minor variations in frequency leakage caused by the relative positioning of adjacent peaks, but this effect does not affect the primary findings and is not considered further. Frequency errors are not shown – for the Plain and WCW results these are equivalent to the value of δ_1 on the x axis. In Fig. 9, $\delta = 0$ for all other peaks i.e. $\delta_{j>1} = 0$; this minimizes spectral leakage so that the results highlight the influence of δ_1 and windowing function "processing loss". In Fig. 10, the other peaks have random δ values i.e. $\delta_{j>1} \neq 0$, so the influence of typical spectral leakage is included.

Table 4
FFT RMS error results for Hann, Hamming and Blackman windowing functions with $\delta_{vj} = 0$.

Tone Index (j)	True Amp (V)	Mean Freq (Hz)	RMS Error					
			Amplitude (%)			Phase (%)		
			Hann	Hamming	Blackman	Hann	Hamming	Blackman
1	1.0e-00	9270.0	1.033e-08	3.086e-04	4.424e-09	1.629e-09	1.387e-09	6.976e-10
2	1.0e-01	9390.0	1.027e-06	3.086e-04	4.401e-07	1.621e-07	1.381e-07	6.947e-08
3	1.0e-02	9150.0	1.021e-05	3.088e-04	4.373e-06	1.633e-06	1.391e-06	6.996e-07
4	1.0e-03	9510.0	2.743e-05	3.095e-04	1.175e-05	4.391e-06	3.741e-06	1.882e-06
5	1.0e-04	9030.0	2.564e-04	3.770e-04	1.098e-04	4.075e-05	3.472e-05	1.747e-05
6	1.0e-05	9570.0	1.699e-03	1.479e-03	7.280e-04	2.700e-04	2.300e-04	1.157e-04
7	1.0e-06	9450.0	6.123e-02	5.218e-02	2.623e-02	9.689e-03	8.255e-03	4.150e-03
8	1.0e-07	9090.0	4.584e-01	3.905e-01	1.964e-01	7.249e-02	6.176e-02	3.107e-02
9	1.0e-08	9330.0	4.065e + 01	3.476e + 01	1.757e + 01	6.868e + 00	5.761e + 00	2.822e + 00
10	1.0e-09	9210.0	4.882e + 02	4.047e + 02	1.728e + 02	2.741e + 01	2.704e + 01	2.503e + 01

Table 5
FFT RMS error results for Hann, Hamming and Blackman windowing functions with $\delta_{\forall j} \neq 0$.

Tone Index (j)	True Amp (V)	Mean Freq (Hz)	RMS Error					
			Amplitude (%)			Phase (%)		
			Hann	Hamming	Blackman	Hann	Hamming	Blackman
1	1.0e-00	9270.0	6.926e + 00	8.408e + 00	5.429e + 00	3.011e-04	4.554e-04	3.011e-04
2	1.0e-01	9390.0	6.921e + 00	8.406e + 00	5.425e + 00	3.025e-04	3.349e-02	3.023e-04
3	1.0e-02	9150.0	6.876e + 00	8.545e + 00	5.389e + 00	3.371e-04	3.354e-01	3.074e-04
4	1.0e-03	9510.0	6.918e + 00	1.277e + 01	5.422e + 00	3.899e-04	1.712e + 00	3.191e-04
5	1.0e-04	9030.0	6.956e + 00	8.741e + 01	5.452e + 00	1.975e-03	1.909e + 01	8.819e-04
6	1.0e-05	9570.0	6.866e + 00	1.041e + 03	5.381e + 00	1.100e-02	2.737e + 01	4.658e-03
7	1.0e-06	9450.0	1.030e + 01	1.924e + 04	6.314e + 00	1.333e + 00	2.881e + 01	5.618e-01
8	1.0e-07	9090.0	2.812e + 01	1.854e + 05	1.303e + 01	4.912e + 00	2.878e + 01	2.009e + 00
9	1.0e-08	9330.0	1.038e + 04	5.571e + 06	4.391e + 03	2.869e + 01	2.888e + 01	2.843e + 01
10	1.0e-09	9210.0	1.041e + 05	5.549e + 07	4.452e + 04	2.882e + 01	2.895e + 01	2.879e + 01

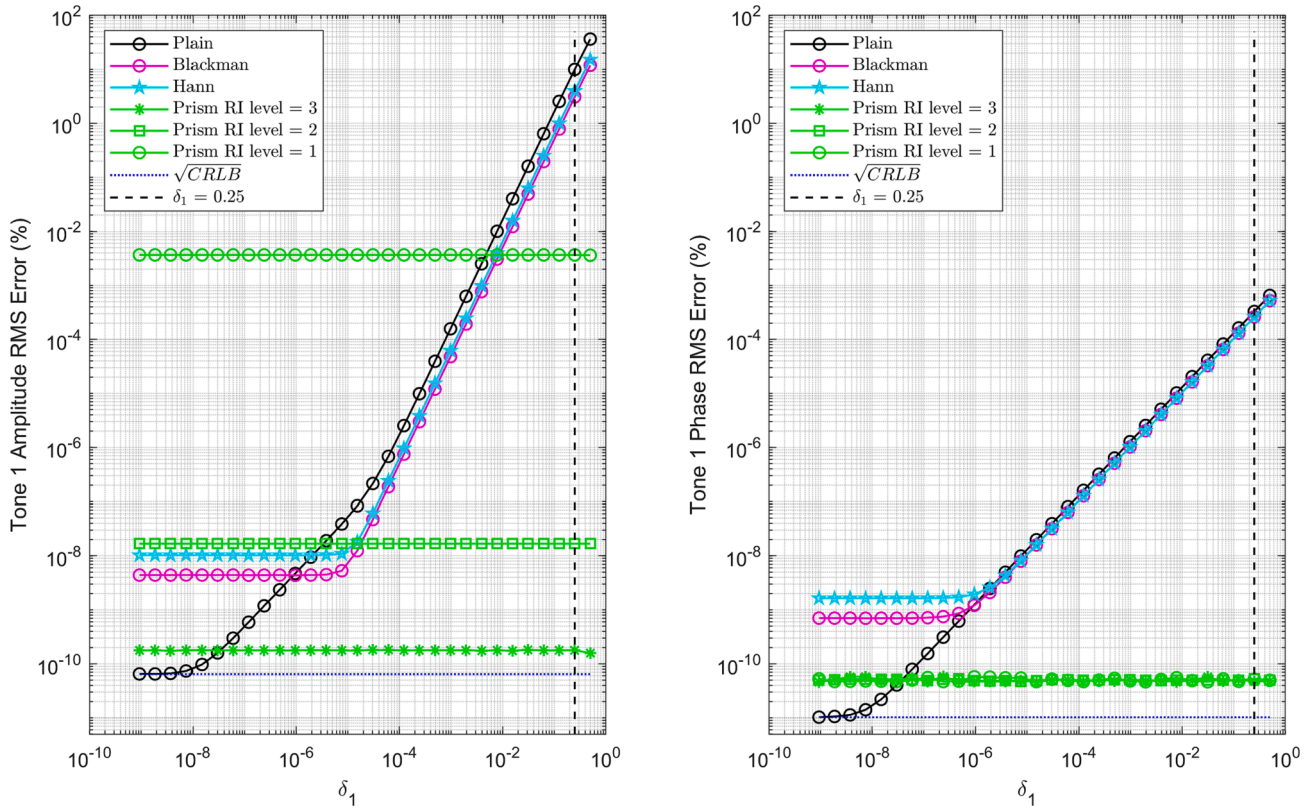

Fig. 9. FFT errors for Tone 1 against $\delta_{j=1}$ for $\delta_{j>1} = 0$. (left) amplitude; (right) phase; 1e-10 V s.d. white noise added; each result is based on 10,000 simulations.

For reference, the square root of the Cramér-Rao Lower Bound (CRLB), the theoretical error limit, is also plotted for each parameter. The signal-to-noise ratio SNR $= A^2/2\sigma^2$ where A is the true tone amplitude and σ^2 is the additive white noise variance. The CRLBs are [14,20]:

$$CRLB_f \cong \frac{12f_s^2}{(2\pi)^2.SNR.N(N^2-1)} \quad (2)$$

$$CRLB_A \cong \frac{2\sigma^2}{N} \quad (3)$$

$$CRLB_\phi \cong \frac{1}{N.SNR} \quad (4)$$

where N is the number of samples (here 48,000). In order to compare these limits against algorithmic performance the values are expressed in the equivalent error percentage scale for each parameter.

In Fig. 9, where spectral leakage is minimal, the Plain FFT amplitude and phase errors both drop steadily with δ_1, moving below all Prism errors for $\delta_1 <$ 3e-8, and approach the $\sqrt{CRLB}$ for $\delta_1 <$ 1e-8. The Black and Hann FFT errors also drop steadily with δ_1 but settle at minimum values of order 1e-8 % for $\delta_1 <$ 1e-5 (amplitude) and at 1e-9 % for $\delta_1 <$ 1e-6 (phase), demonstrating "processing loss", in both cases about two orders of magnitude higher than the corresponding Prism error for any δ_1. Note this error saturation only occurs when the noise level is very low, as demonstrated later in the paper.

Fig. 10 shows the impact of spectral leakage induced by random values of δ for the other tones. The Plain FFT is particularly sensitive, with amplitude and phase errors settling at ~ 1e-2 % and 1e-3 % respectively, and no further error reduction for $\delta_1 <$ 1e-2. The WCW windowing functions reduce spectral leakage resulting in lower errors, settling at ~ 1e-6 % (amplitude) and ~ 1e-7 % (phase) for $\delta_1 <$ 1e-4. However, there remains a residual increase in error over the $\delta_{j>1} =$ 0 results by a factor of approximately 100 for both parameters: in this

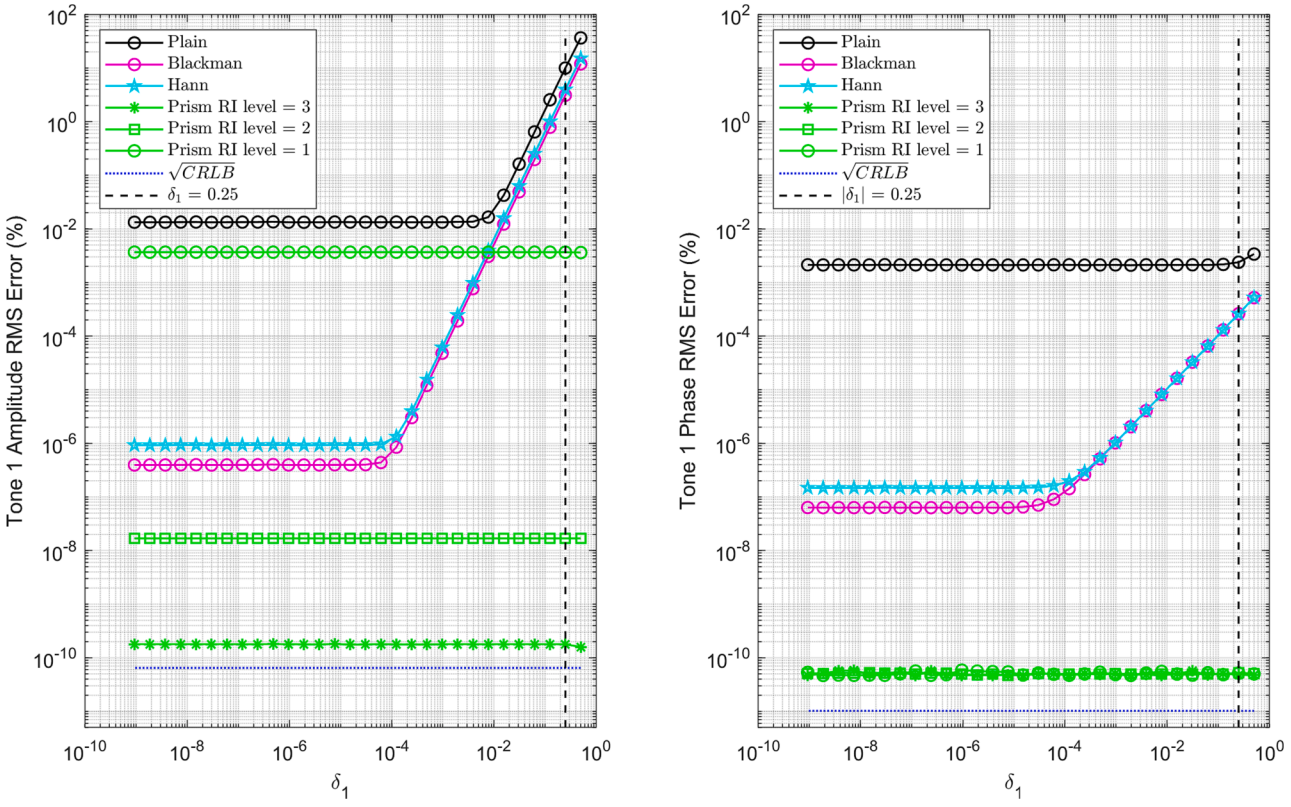

Fig. 10. FFT errors for Tone 1 against $\delta_{j=1}$ for $\delta_{j>1} \neq 0$. (left) amplitude; (right) phase; 1e-10 V s.d. white noise added; each result is based on 10,000 simulations.

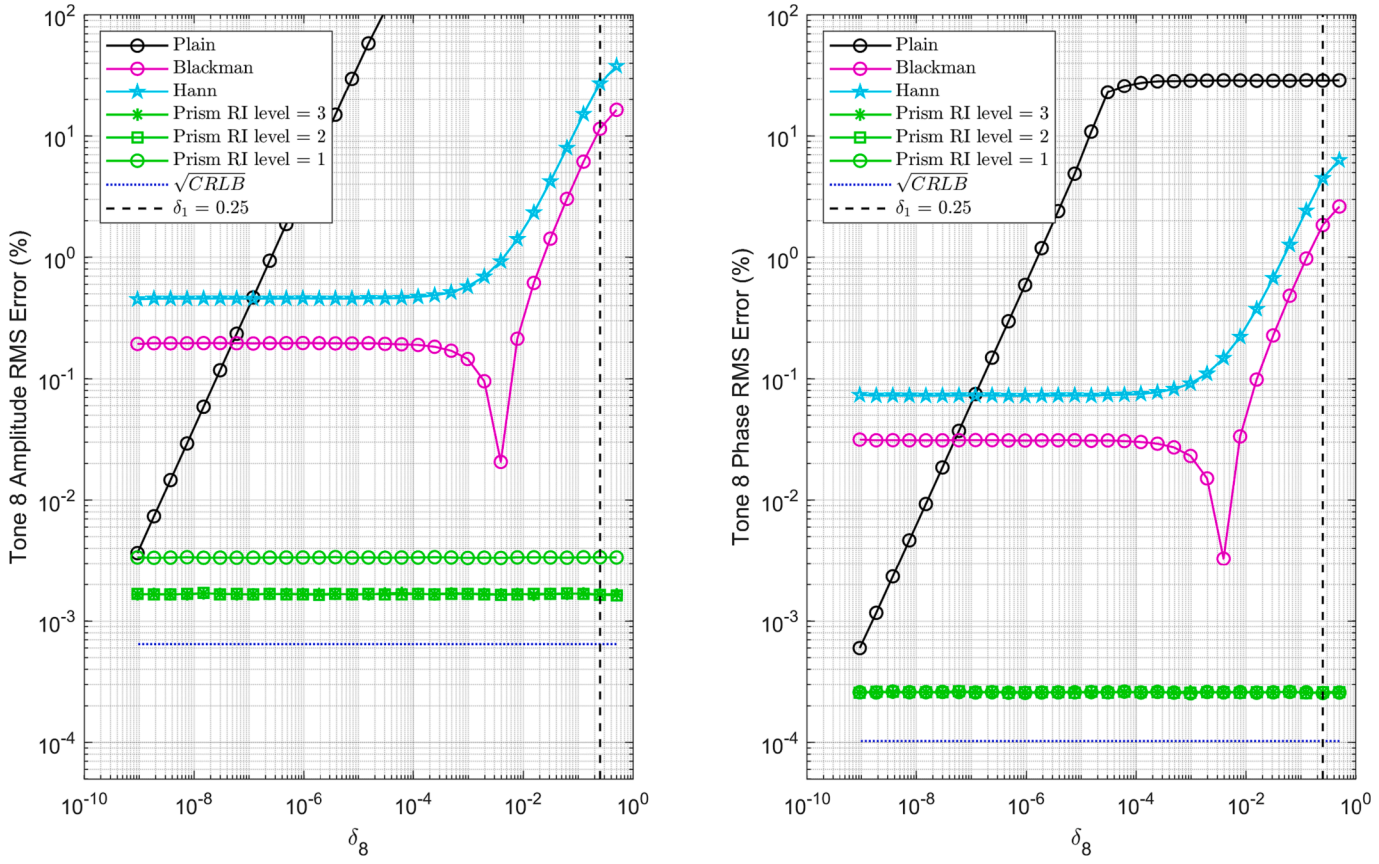

Fig. 11. FFT errors for Tone 8 against $\delta_{j=8}$ for $\delta_{j\neq 8} = 0$. (left) amplitude; (right) phase; 1e-10 V s.d. white noise added; each result is based on 10,000 simulations.

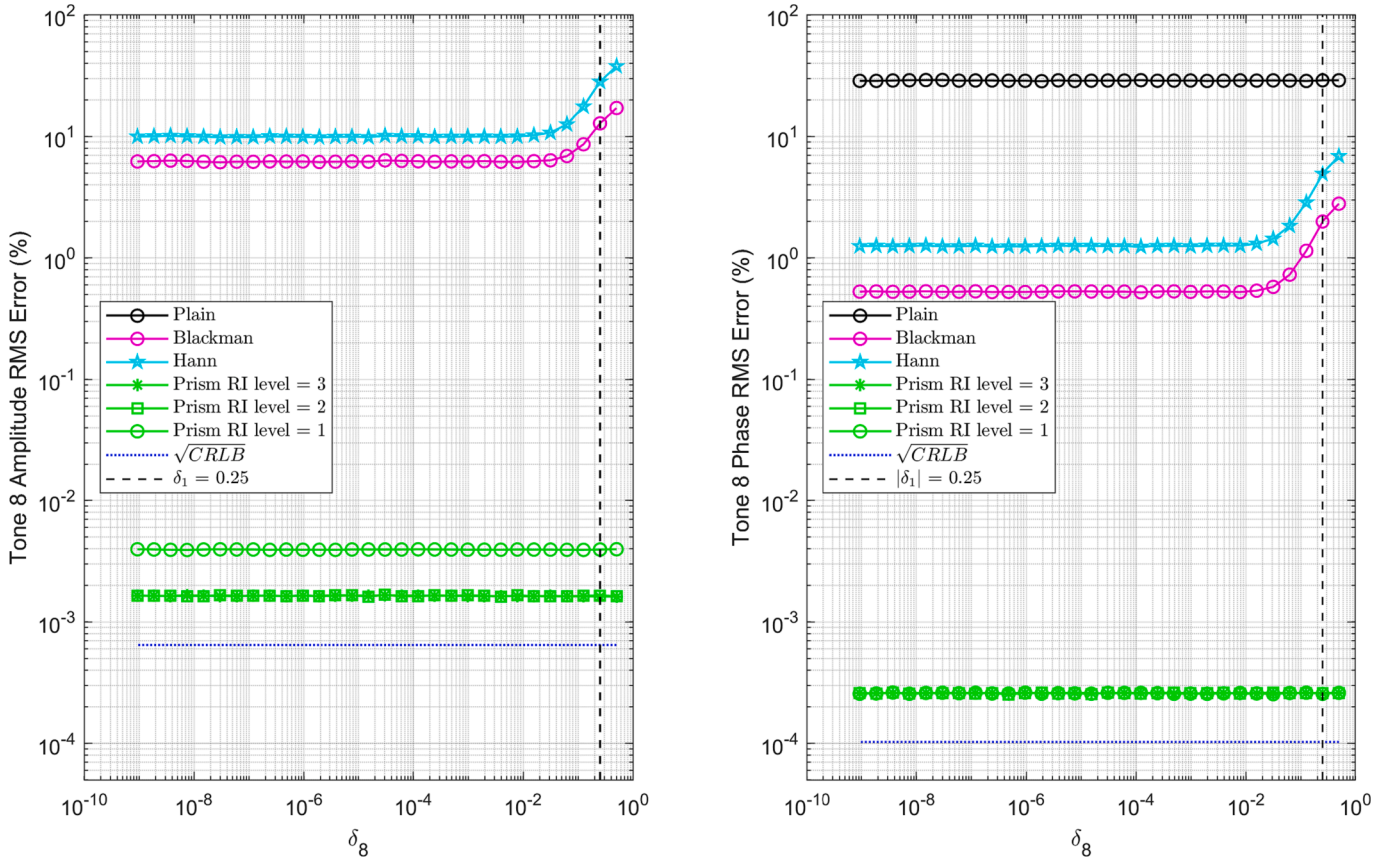

Fig. 12. FFT errors for Tone 8 against $\delta_{j=8}$ for $\delta_{j\neq 8} \neq 0$. (left) amplitude; (right) phase; 1e-10 V s.d. white noise added; each result is based on 10,000 simulations.

simulation these WCWs are not able to entirely remove the effect of spectral leakage even on the largest tone.

Figs. 9 and 10 also show the Prism RSMEs for amplitude and phase using different levels of Romberg Integration (RI). For a given RI level (RIL), the Prism errors are approximately constant for all values of δ_1 and indeed $\delta_{\forall j}$ (comparing Fig. 9 and Fig. 10). The deep attenuation of the Prism window (Fig. 6) results in the complete removal of spectral leakage influence on the largest tone.

The RI level has little impact on the Prism phase error, which remains approximately constant at ~ 6e-11 %. However, for amplitude (and, as will be seen, frequency) higher levels of RI deliver substantial reductions in RMSE error, dropping from 3.7e to 3 % (RIL 1) to 1.7e-8 % (RIL 2) to 1.8e-10 (RIL 3). Note that Prism results reported earlier (Table 3, Fig. 4) all use RIL 3. Overall, for amplitude and phase, the Prism RIL 3 errors are approximately constant at $\sim 2.5 \times$ and $\sim 5 \times \sqrt{CRLB}$ respectively for all values of δ_1. This demonstrates that the incorporation of Romberg Integration into window design (as described in further detail below) provides a means of substantially reducing the "processing loss" associated with windowing functions.

Figs. 9 and 10 show the results obtained for the largest Tone 1, which, due to its relative magnitude, is the least affected by spectral leakage. Accordingly, it is useful to consider the FFT performance for lower amplitude peaks which, by design in this example, all have significantly larger neighbours. While the Prism FFT successfully identifies all peaks, the smallest peak seen by the WCW methods is Tone 8, with amplitude 1e-7 V. This is centred at 9090 Hz (Eq. (1), and has neighbouring peaks with amplitudes 1e-4 V and 1e-2 V, i.e. 1,000 and 100,000 times larger than its own. Note further that the signal to noise ratio for Tone 8 is also significantly reduced.

Figs. 11 and 12 show the corresponding results for the cases $\delta_{j\neq 8} = 0$ and $\delta_{j\neq 8} \neq 0$ respectively, which broadly reproduce the behaviour manifest in Figs. 9 and 10. The Prism errors are independent of the value of δ_8 and retain (for RI = 3) approximately the same ratios to the corresponding CRLB (which is higher than in Figs. 9 and 10 due to the reduced signal-to-noise ratio for Tone 8). The Plain FFT results are highly sensitive to all sources of spectral leakage, dropping rapidly only when $\delta_{j\neq 8} = 0$ (Fig. 11) and when δ_8 approaches zero. The two WCW methods saturate as δ_8 reduces; in Fig. 11, where leakage from the adjacent peaks is suppressed by setting $\delta_{j\neq 8} = 0$, the errors remain approximately 100 times those of the Prism (compare Fig. 9); however, in Fig. 12 the WCW errors are approximately 10,000 (amplitude) and 1,000 (phase) times greater than the Prism errors. These higher errors are attributed to the limited capability of the WCWs to suppress spectral leakage from the large neighbouring peaks which, however, the Prism windows are able to achieve. One unexplained feature of these results is the pre-saturation drop and subsequent rise in the Blackman errors for $\delta_8 \sim$ 4e-3 in Fig. 11, but this does not undermine the basic findings of these results as discussed below.

The simulations in this section have demonstrated that, while windowing functions have an essential role in reducing spectral leakage, for WCW functions the resulting FFT calculations remain sensitive to the value of δ for each tone, "processing loss", and spectral leakage from large adjacent neighbouring peaks. By default, in most practical applications, |δ| has a random uniform distribution with an average value of 0.25 (marked with a horizontal dashed line on Figs. 9 - 12). The results in Tables 1 and 4 reflect the idealized conditions in Figs. 9 and 11 as δ tends to zero. The results in Tables 2 and 5 reflect the real-world default case shown in Figs. 10 and 12 at the $\delta = 0.25$ line. While there is scope for the conventional techniques to provide reduced errors if δ can be accurately estimated, the Prism technique is able to provide results close to the CRLB which are independent of δ and $\delta_{\forall j}$. The FFT frequency discretization issue is addressed in the next section.

4. Frequency interpolation: Addressing frequency discretization

For some applications, a basic spectral analysis of a data set, using a WCW followed by the selection of FAP results based on peak bin values, may provide acceptable precision. However, a number of methods have been developed to further reduce FFT error.

One simple and widely used technique is zero packing [13], in which the dataset (after windowing) is appended with zero values to extend its nominal length. This has the effect of reducing the FFT reporting frequency interval, which may in turn reduce the corresponding peak bin FAP errors. For the current simulation, zero packing provides some improvement. For example, if the $\delta_{\forall j} \neq 0$ WCW data sets are doubled in length by appending 48 k zero values, the FFT results show a reduction of the frequency error by a factor of 2 (as would be expected), with corresponding reductions by factors of up to 4 and 2 for amplitude and phase respectively (results not shown). These modest error reductions come at the increased computational cost of the longer FFT and, as discussed in [13], there are usually limits to the precision improvements that can be obtainable via any degree of zero packing.

Other techniques aim to locate the frequency of a peak to a higher precision than that provided by bin discretisation, in other words to estimate δ. This in turn can lead to significantly improved estimates of FAP. The calculation of δ has been an area of active research over several decades (e.g. [10,14–20]), typically entailing the use of a specified windowing function (e.g. Hann and/or Hamming [15 –19]) followed by interpolation of the FFT results (IpFFT) around each peak value; interpolation has also been used to locate hidden peaks [16]. It is not possible in this paper to review the mathematical details of the variety of techniques which, with increasing sophistication, and computational cost, exploit particular properties of the windowing function and/or data set. As an example, in [20] FAP estimates are calculated, with errors approaching the CRLB limits, using a combination of IpFFT and modelling of the effects of spectral leakage, where the input signal is assumed to contain a series of harmonics and additive noise. In [24], the signal is assumed to consist of a single component with exponential decay, where the decay rate is also estimated.

For the current simulation example, improvements over the coarse grain search results of Table 5 to be derived from the application of IpFFT (depending on the particular implementation) can be illustrated in three steps, with increasing level of sophistication:

- The error in δ for each individual peak is reduced towards zero. The corresponding change in amplitude and phase estimates are illustrated by the reduction in errors for the WCW functions as δ moves towards zero in Fig. 10 (for Peak 1), and Fig. 11 (for Peak 8). Here the spectral leakage from adjacent peaks is still a significant source of error.
- The effects of spectral leakage from adjacent peaks may be reduced (for example by modelling and/or using an improved WCW function [17]); potential improvements are illustrated by the differences (for near-zero δ) between Figs. 10 and 9, and Figs. 12 and 11, respectively, where in the latter figures the effects of spectral leakage from adjacent peaks has been removed. Table 4 provides results where the effects of spectral leakage are completely removed.
- Finally, compensation for windowing effects may be included (e.g. [20] uses a linear fit procedure). This can be thought of a shift downwards, for near-zero δ, from the saturated WCW error curves towards the Prism results and the CRLB limits in Figs. 9 and 11.

Much of the current literature emphasizes computational efficiency, seeking to deliver results over short data sets (e.g. 512 [19,20] or 1024 [14,17] samples), with a moderate signal-to-noise ratio taking into consideration factors such as ADC bit precision [15,16], with a view towards fast, real-time implementation [10,20]. There has been less emphasis on dealing with large dynamic range between peaks, such as the 10^9:1 ratio used in the current example. As previous discussed, the

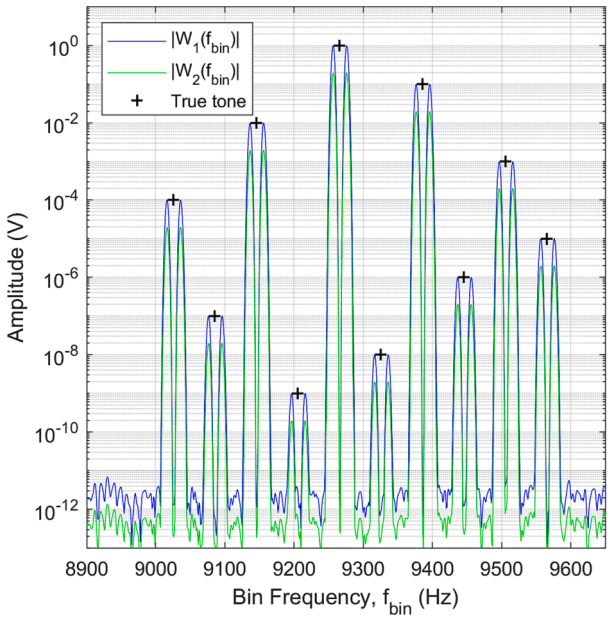

Fig. 13. Amplitudes from FFT results $W_1(f_{bin})$ and $W_2(f_{bin})$ generated by Prism windows *Gs1* and *Gs2*.

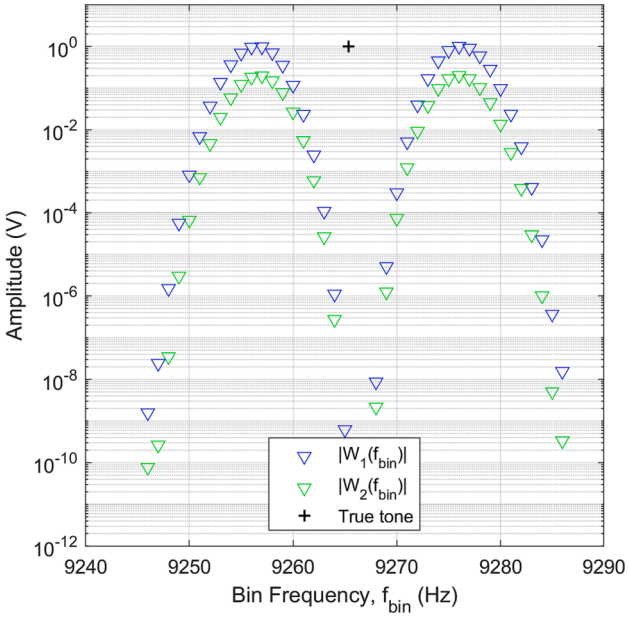

Fig. 14. Prism FFT bin amplitudes around Tone 1.

results in Table 3 (Prism with $\delta_{\forall I} \neq 0$,) compare favourably against Table 4 (WCW with $\delta_{\forall I} = 0$) where there is little scope for further improvement from an IpFFT technique (i.e. steps 1 and 2 above are carried out perfectly).

The Prism technique for locating peak frequency is now described; this makes no assumptions other than a sufficient separation of the tones. While IpFFT techniques interpolate between bin values to calculate δ, the primary division of interest is the Prism filter lobe, shown in Fig. 6. As explained in Section 5, the Prism filter design has a characteristic frequency, m [4], corresponding to the lobe width in Fig. 6. In general, the dimensionless frequency $r = f/m$ for arbitrary f, describes the relative position of f within the lobe, where the gains of the windowing functions *Gs1* and *Gs2* depend on r. Prism FFT peak detection can be viewed as an inverse problem: given the relative amplitudes of

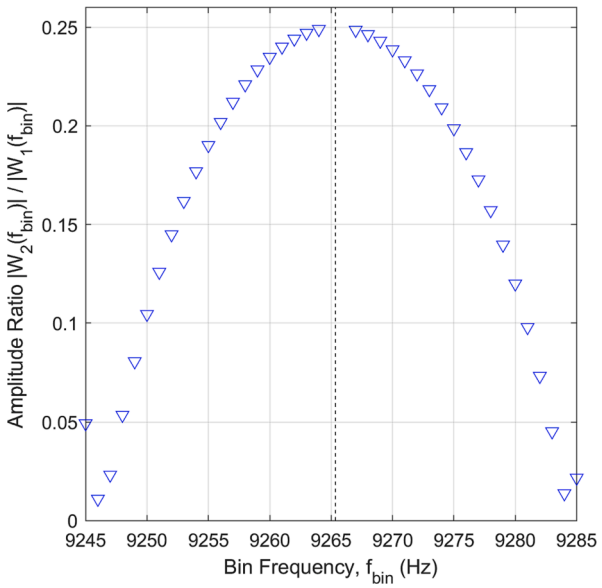

Fig. 15. Amplitude ratio $x(f_{bin}) = |W_2(f_{bin})| / |W_1(f_{bin})|$;vertical line shows true peak frequency on x axis.

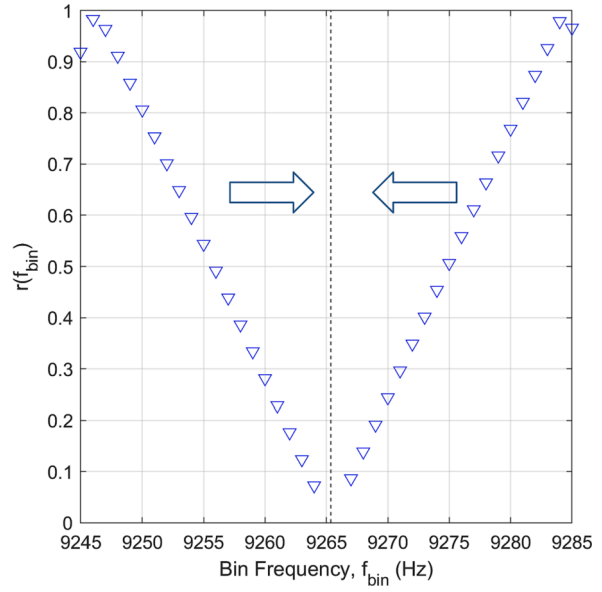

Fig. 16. Dimensionless frequency $r(f_{bin})$; vertical line shows true peak frequency on x axis.

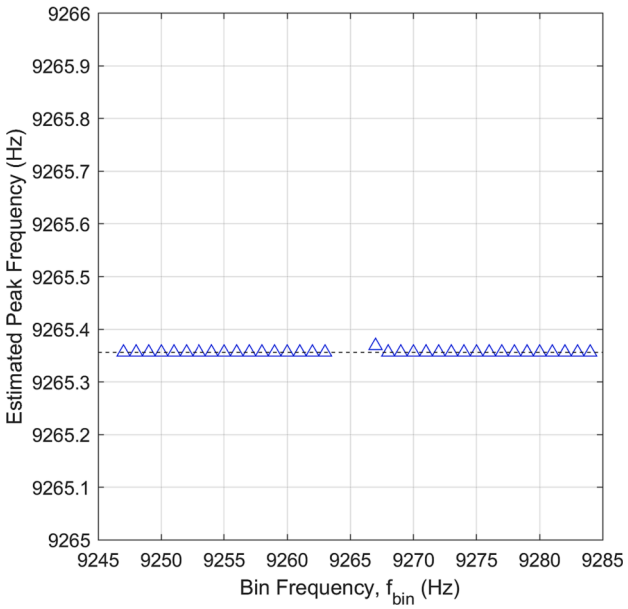

Fig. 17. Estimates of local peak frequency; horizontal line shows true peak frequency on y axis.

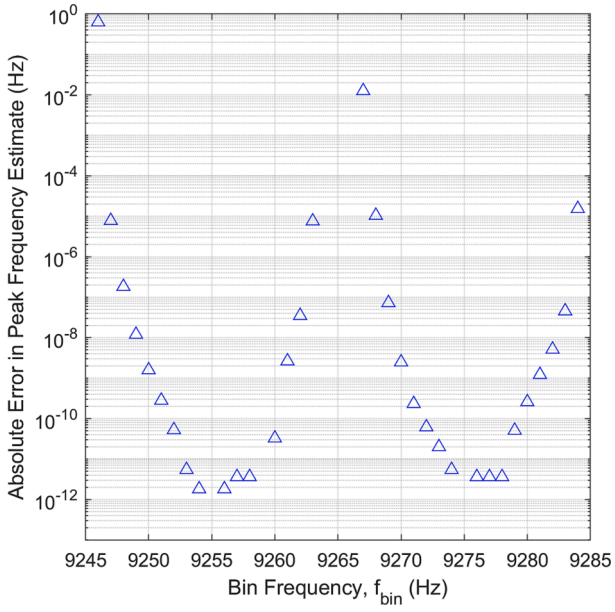

Fig. 18. Error in peak frequency estimate.

Gs1 and *Gs2* - as observed at each bin location - calculate the value of *r,* and hence the distance to the local peak (which occurs at $r = 0.5$). Here sufficient separation is assumed so that there is at most one peak in any lobe. The process of peak location is illustrated in Figs. 13-18. Given the $N = 48{,}000$ data values $d(i)$, and values $Gs1(i)$ and $Gs2(i)$, windowed data sets $w_1(i)$ and $w_2(i)$ are calculated as follows:

$$w_1(i) = Gs1(i).d(i), w_2(i) = Gs2(i).d(i) \tag{5}$$

The standard FFT calculation applied to $w_1(i)$ and $w_2(i)$ generates corresponding spectral values $W_1(f_{bin})$ and $W_2(f_{bin})$ for each bin frequency f_{bin}. Fig. 13 shows typical amplitude (root sum square) results $|W_1(f_{bin})|$ and $|W_2(f_{bin})|$. Given the high stopband attenuation of the Prism windowing functions (Fig. 6) the noise floors are low, so all the true peaks are revealed. However, two lobes are generated symmetrically around each true peak. These occur because, as shown in Fig. 6, the Prism filters are offset from zero Hz (explained in more detail in section 5). In the final results (e.g. in Fig. 4), each double lobe artifact is replaced by the estimate of the true peak and a dead-zone.

Fig. 14 shows the FFT results in detail around the highest peak, at approximately 9270 Hz. Amplitude values $|W_1(f_{bin})|$ and $|W_2(f_{bin})|$ are shown for each FFT bin frequency f_{bin}. Fig. 13 shows the corresponding amplitude ratio $x(f_{bin}) = |W_2(f_{bin})|/|W_1(f_{bin})|$. As explained in Section 5 and Appendix 1, the corresponding value of r for each bin location f_{bin} is:

$$r(f_{bin}) = \sqrt{\frac{4x(f_{bin}) - 1}{x(f_{bin}) - 1}} \tag{6}$$

Fig. 16 shows the resulting values of $r(f)$, which vary linearly with the distance from the true local peak frequency. The direction of the peak is implied by the positive or negative slope of adjacent r values. Fig. 17 shows the estimated values of the peak calculated from each bin

result, using:

$$\widehat{f}_{peak} = f_{bin} \pm 0.8mr(f_{bin}), \tag{7}$$

where an addition or subtraction is applied depending on the slope of r. Most of these values appear close to the true peak: note that the y-scale covers the distance between adjacent bin frequencies, where the randomly selected value in this case is $\delta = 0.356122$. Fig. 18 shows the corresponding frequency errors. Good peak location is achieved for several of the bins, with the absolute error dropping as low as 2e-12 Hz. The lowest errors are associated with high W_1 amplitudes (Fig. 14) and r values close to 0.5 (Fig. 16). A first estimate of the peak location is selected, for example based on the bin with the highest local W_1 amplitude, or with the value of r closest to 0.5. Where there is a lower signal-to-noise ratio, and/or where additional application-specific knowledge is available, a more sophisticated and/or robust selection may be made.

Having estimated the peak frequency $\widehat{f}_{peak}$ from r, the corresponding values of peak amplitude and phase are derived. The *Gs1* windowing function is scaled so that the W_1 amplitude is accurate if $\widehat{f}_{peak}$ exactly matches the bin frequency, i.e. $\delta = 0$. There are a variety of means to compensate for the δ offset, such as applying a gain correction, or applying an interpolation technique. Here a conceptually simple and accurate technique is used, which may be described as 'interrogating' the windowed data set w_1 at the peak frequency $\widehat{f}_{peak}$, thereby ensuring the corresponding value of δ is close to zero. This is implemented as follows. As the *Gs1* filter peak is offset 0.4 m from zero Hz, an offset frequency f_o is calculated:

$$f_o = \widehat{f}_{peak} - 0.4m \quad (\text{or} f_o = \widehat{f}_{peak} + 0.4m) \tag{8}$$

New Fourier coefficients are calculated using:

$$s_o = \sum_{i=1}^{N} w_1(i).\sin(2\pi f_o.i.\Delta t) \tag{9}$$

$$c_o = \sum_{i=1}^{N} w_1(i).\cos(2\pi f_o.i.\Delta t) \tag{10}$$

where Δt the sampling period = $1/fs$, the sampling rate. Estimates of the amplitude and phase at the peak are calculated from the root sum square and arctan of s_o and c_o (see Section 5 and Appendix 1). If desired, revised values of the *Gs2* Fourier coefficients can also be obtained using equations (9) and (10) with the $w2$ values, so that an improved estimate of the peak frequency can be obtained via recalculations of equations (6) and (7). All results reported here used the interrogation technique to improve FAP values for each peak.

The most appropriate implementation of the Prism peak location method may require application-specific knowledge. For example, a rule is needed to identify the likely presence of a peak against the noise floor. This may use a fixed threshold, a relative threshold against the local noise floor, or a combination of the two. Localised searching should be guided by the lobe width m and the presence of two spectral lobes either side of each true peak. The selection of the most suitable bin amplitude and the direction of the true peak frequency, whether positive or negative (Fig. 14), must be robust to the presence of noise. Further steps may be required to analyse spectra where peaks are closer than 2 m Hz, as discussed in the concluding section.

5. Prism filter design: Addressing time discretization

The derivation of the Prism FFT calculation is explained in this section by describing the development of the Prism filter design. A brief overview of the Prism is provided in the Introduction, and a full mathematical presentation is given in [4]. Fig. 19 shows its structure, which consists of a cascade of Fourier-style integral blocks. The Prism has two parameters, the characteristic frequency m and harmonic number h. Each of the Prism integrals cover a time interval of $1/m$, so that the total Prism filter duration is $2/m$ seconds. Within each integral the input signal is multiplied by a sine or cosine modulation function of frequency hm. h is usually a small positive integer –in this paper the highest value is $h = 3$, but values in excess of 1,000,000 have been used (see [6]). The Prism generates one or two outputs, G_s^h and G_c^h, which form an orthogonal sine/cosine pair. The output gains are functions of h and the dimensionless frequency $r = f/m$:

$$\Gamma_s(r,h) = sinc^2(r)\frac{r^2}{r^2 - h^2} \tag{11}$$

$$\Gamma_c(r,h) = sinc^2(r)\frac{hr}{r^2 - h^2} \tag{12}$$

This variation in gain with both r and h supports the direct calculation of peak frequencies in the Prism FFT. The double Fourier integral structure in the Prism results in the outputs G_s^h and G_c^h both having zero gain (i.e. a notch) at all multiples of m, including at zero Hz. The Prism behaves as an FIR filter but performs its calculation recursively; for long filters this can results in computational reductions of several orders of magnitude compared with the conventional, convolutional form of FIR filter [6].

5.1. Romberg Integration

Romberg Integration (RI) is a powerful technique used in numerical analysis. Dutka [5] provides an historical review of its background and

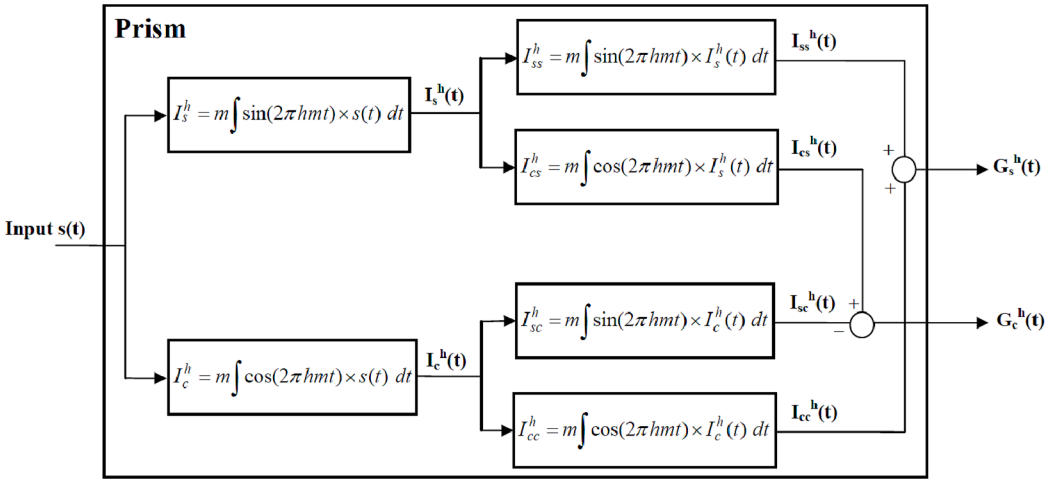

Fig. 19. Prism structure (from [4]).

development. In summary, RI generates a sequence of trapezoidal integrations over the same interval of interest, using different numbers of data points, and then combines the results of these integrations to reduce numerical error, often by several orders of magnitude. At the lowest level of integration, all data points are used. At the next level, only half the data points are used, selecting every other value. Each subsequent level reduces the number of data points by a factor of two. The trapezoidal integral results at each level are combined to give an improved estimate with reduced error. The error reduction is limited by various factors, including machine precision and the noise/error/uncertainty in the function or data being integrated. The number of RI stages is usually low; here no more than 3 stages are considered.

In most applications, and as described in standard textbooks on integration (e.g. [21]), RI is applied to a function which can be evaluated at any point over the desired interval of integration, and where such evaluations may occur in any order. For example, if some physical phenomenon is described by a formula which includes several variables, then RI may be used to numerically integrate the formula against one or more of the variables over a desired range of values.

In the Prism, RI is applied to a sliding window of time series data with a fixed order and fixed sampling interval; it is used to compute the output of each integration block in Fig. 19. A detailed description of the implementation is beyond the scope of the current paper and will be the subject of a future publication. In summary, RI is implemented efficiently via a set of totalizers for each RI level and each integral block. One totalizer is needed for the first level; two for the second level; four for the third level, and so on. The utilization of RI constrains the Prism window length in samples to be a multiple of some small power of two, which is readily accommodated.

For the purposes of the current paper, i.e. the design of Prism windowing functions for FFT analysis, the inclusion of various levels of RI results in small refinements to the values of the window coefficients, but these provide significant improvements in FFT precision, as shown above in Figs. 9 and 10. The adjustments to window function coefficients are described later in this section.

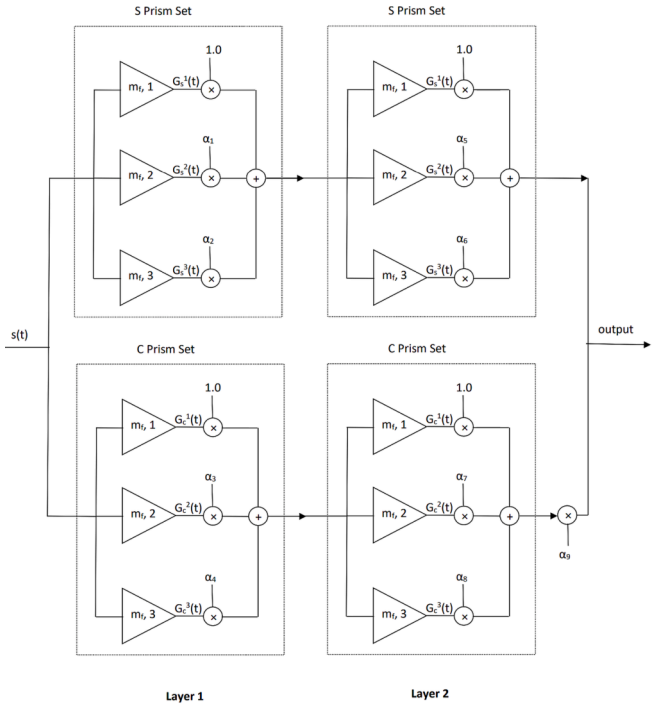

Fig. 20. Structure of two layer 'low pass' Prism filter network. Each triangle represents a single output Prism.

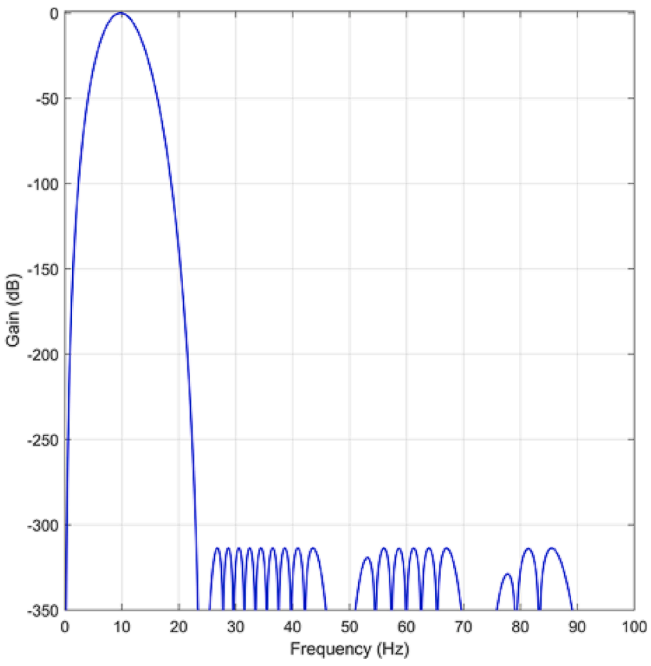

Fig. 21. Frequency response of 12 layer 'low pass' Prism filter, with m = 23.81 Hz.

5.2. Prism 'low pass' filter designs

Fig. 20 shows the structure of a Prism network implementing a low pass filter, comprising of two layers, each with six single output Prisms, all sharing a common value of m. Each layer has three Prisms with sine outputs for $h = 1, 2, 3$ and similarly three cosine output Prisms. Each set of three outputs are combined using optimized weights α_i. After two layers the sine and cosine paths can be combined (applying a further weight α_9) as they are 180 degrees out of phase. This is a generalization of the design described in [7], which only uses cosine output Prisms. The weights α_i are selected to provide some desired filter output characteristic, which in this case is to maximize the stopband attenuation above m Hz. Further layers, with independent weightings, may be added to provide additional filtering, where each pair of layers provides approximately 50 dB of stopband attenuation. From a given set of weightings, a lowpass filter with a desired value of m, delivering a specific filter bandwidth, can be instantiated. Fig. 21 shows the frequency response of the 12 layer 'low pass' filter, incorporated into the Prism windowing functions used in this simulation example, where m is 23.81 Hz.

5.3. Prism tracking

Fig. 22 shows a typical architecture used to perform real-time tracking of frequency, phase and/or amplitude of a signal component using a low pass filter and a tracking stage. While the filter design of Figs. 20 and 21 has powerful attenuation, it is inflexible: the passband is fixed to frequencies around $0.4m_f$ where m_f is the selected value of m for the lowpass filter. Heterodyning is a widely used technique for shifting a target frequency into a filter passband. This is achieved by multiplying the original signal by a pure sinusoid with a frequency equal to the difference (or sum) of the target and the peak frequency. The original target frequency is readily calculated after the heterodyned signal has been filtered and tracked. After lowpass filtering, a tracking stage follows, which uses two dual-output Prisms with values $h = 1$ and 2. For notational convenience here the outputs of the $h = 1$ Prism are labelled simply G_s and G_c, while those for the $h = 2$ Prism are labelled G_z and G_k. Optimally, the tracking Prisms use a smaller value of m than the lowpass

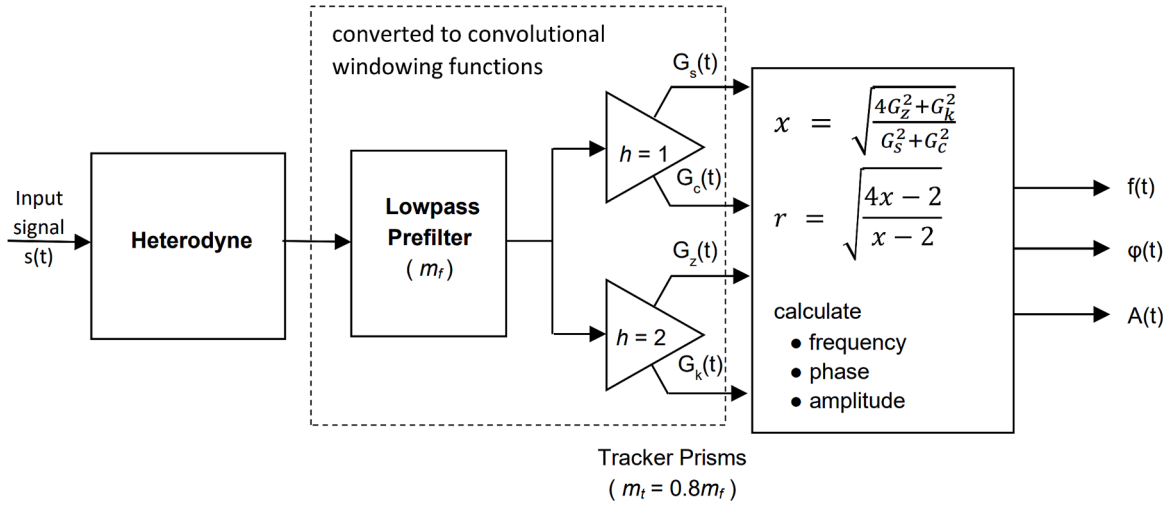

Fig. 22. Stages used in a real-time system to calculate FAP values via Prism low pass filtering and tracking. The dashed box shows the subsection of the network that is converted to convolutional form to create windowing functions *Gs1* and *Gs2*.

filter, $m_t = 0.8m_f$ so that for the trackers, $r = 0.5$ coincides with the peak filter gain at $0.4m_f$. Assuming the filter output contains only a single sinusoidal component, an estimate of its dimensionless frequency r can be obtained sample-by-sample from the tracker Prism outputs using the following equations (See Appendix 1 for a full derivation):

$$x = \sqrt{\frac{4G_z^2 + G_k^2}{G_s^2 + G_c^2}} \tag{13}$$

$$\hat{r} = \sqrt{\frac{4x-2}{x-2}}. \tag{14}$$

From the estimated value of r, corresponding values of frequency, amplitude and mid-window phase of the tracked component are derived from the $h = 1$, G_s and G_c results as follows:

$$\hat{f} = \hat{r}m \tag{15}$$

$$\hat{A} = \left[\frac{\pi}{sin(\pi\hat{r})}\right]^2 \left[1-\hat{r}^2\right]\sqrt{G_s^2 + \hat{r}^2 G_c^2} \tag{16}$$

$$\hat{\varphi} = \text{atan2}(-G_s, \hat{r}G_c) \tag{17}$$

Note that to calculate the corresponding parameter values for the original signal *s(t)*, compensation must be applied for the prefiltering and heterodyning stages; these depend upon the implementation details (for example, whether the heterodyning frequency is fixed or dynamic) which are not required for the purposes of the current paper.

5.4. Convolutional implementation and windowing functions

In the first application of Prism signal processing to FFT analysis [3], a signal processing path similar to Fig. 22 is used (see Fig. 5 in [3]). A single bandpass filter stage is used instead of heterodyning/lowpass filter steps in Fig. 22, while a single tracker Prism, using a different tracking calculation, replaces the dual Prism tracker used here. The data set was treated as a time series, pushed through the signal processing path to obtain a corresponding time sequence of FAP estimates which were averaged to provide final FFT results. However, this method entails the use of many bandpass filters, each covering a narrow range of the spectrum, to be designed, instantiated and applied to the dataset, which is computationally expensive.

The processing path of Fig. 22 leads to the development of the FFT Prism windowing functions by the following steps:

1. To carry out a systematic analysis of the full spectrum of the input signal *s(t)* using the scheme of Fig. 22, only the heterodyning frequency need vary: all other components remain unchanged.
2. The basic, 'plain' FFT calculation can be thought of as being equivalent to a regularly spaced set of heterodynes, where:
   a. Pairs of orthogonal (sine/cosine) heterodynes are generated;
   b. Each FFT output is a summation of the heterodyned data. The signal component at each bin frequency is 'shifted' to zero Hz and its average value taken over the data window.
3. All signal processing steps in Fig. 22, except the final FAP calculations, are linear, so their order can be rearranged without changing the resulting value. Accordingly, the heterodyning stage can be implemented as the final step; implemented using the 'plain' FFT calculation, a systematic set of heterodyned results can be generated with $o(nlog_n)$ efficiency.
4. The lowpass prefilter combined with any single output from a tracker Prism can be converted into convolutional form via the well-known method of calculating the impulse response ([4]).
5. As the 'plain' FFT calculation provides systematic heterodyning and final summation, then each such convolution can be used as a windowing function applied over the entire data set; the FFT calculation provides a single, averaged (complex) result for each bin frequency instead of a time series. However, as the FFT 'bin' frequency is the heterodyne frequency, Prism windowing functions will generate offsets in the observed peak frequency (as seen e.g. in Figs. 11 and 12).
6. In Fig. 22 a single heterodyne is generated from the input signal, and therefore potentially four convolutional windowing functions are needed to reproduce the four inputs to the tracking calculation. However, the FFT calculation inherently and efficiently generates results for sine/cosine pairs of heterodynes, so that in practice only two Prism windowing functions are needed. These are labelled *Gs1* and *Gs2*, and correspond to the low pass filter combined with the G_s ($h = 1$) and G_z ($h = 2$) tracker outputs of Fig. 22 (see Fig. 23). As explained in Appendix 1, this requires minor modifications to the calculation of r: the calculation of the *Gs1/Gs2* amplitude ratio followed by equation (6) is equivalent to the calculation of r via equations (13) and (14).

The windowing functions are created as follows. Given a desired FFT data window length (here 48 k samples), and a desired filter/tracker structure (here 12 layer filter and tracker), the first step is to determine the number of samples to be used in each Prism integral, where each integral has a duration $1/m$. Each Prism has two integrals, so for a 12

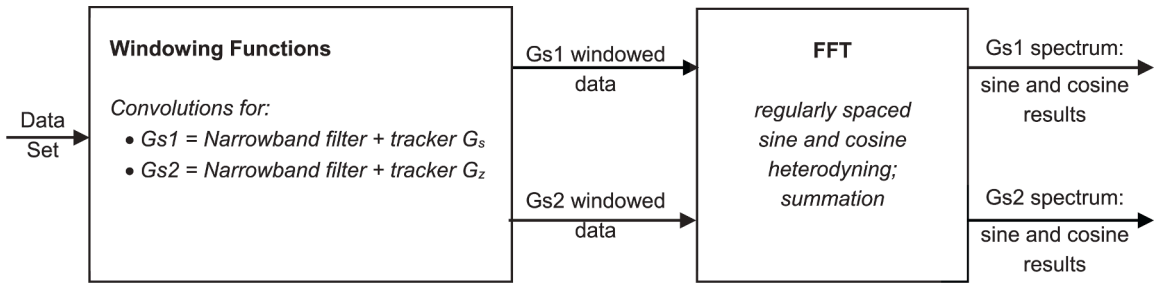

Fig. 23. Stages used in tracking FAP values via Prism low pass filter and tracking.

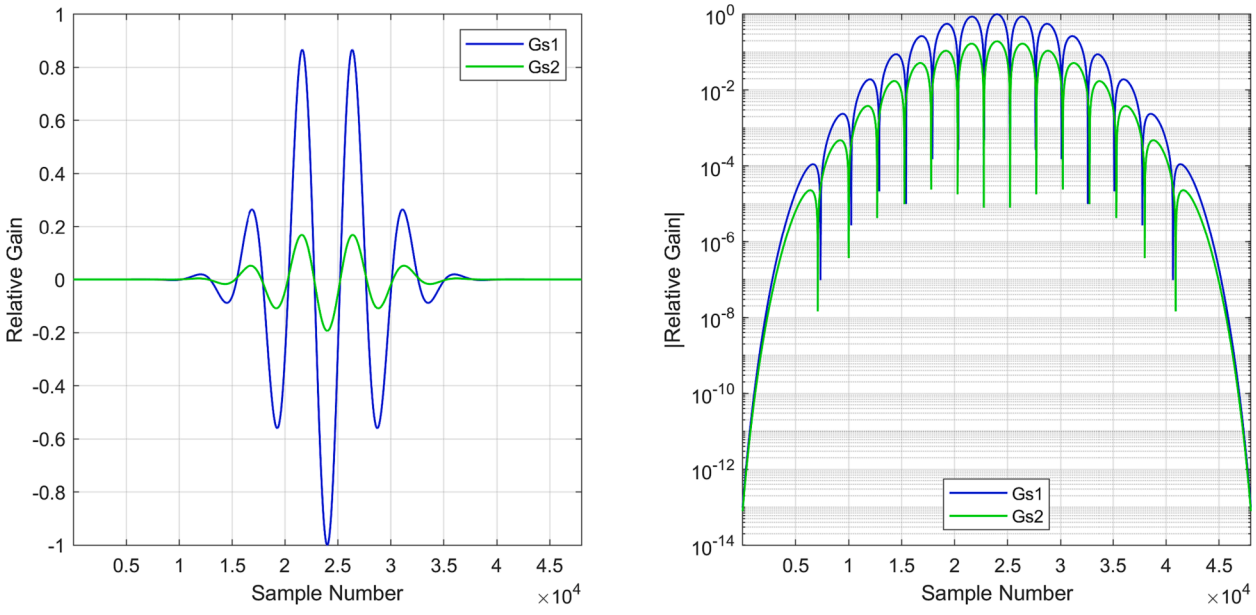

Fig. 24. *Gs1* and *Gs2* windowing functions (left) linear scale; (right) logarithmic scale.

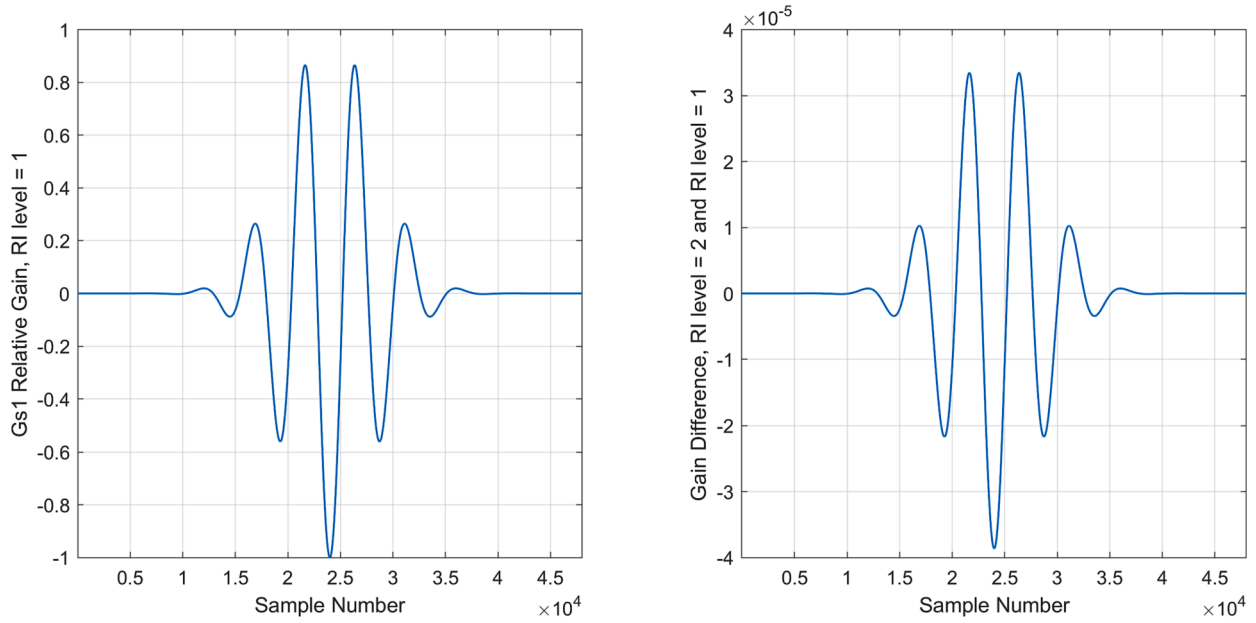

Fig. 25. Application of Romberg Integration to window function design: (left) GS1 windowing function using RIL 1; (right) difference in *Gs1* windowing function value for RIL 1 and RIL 2.

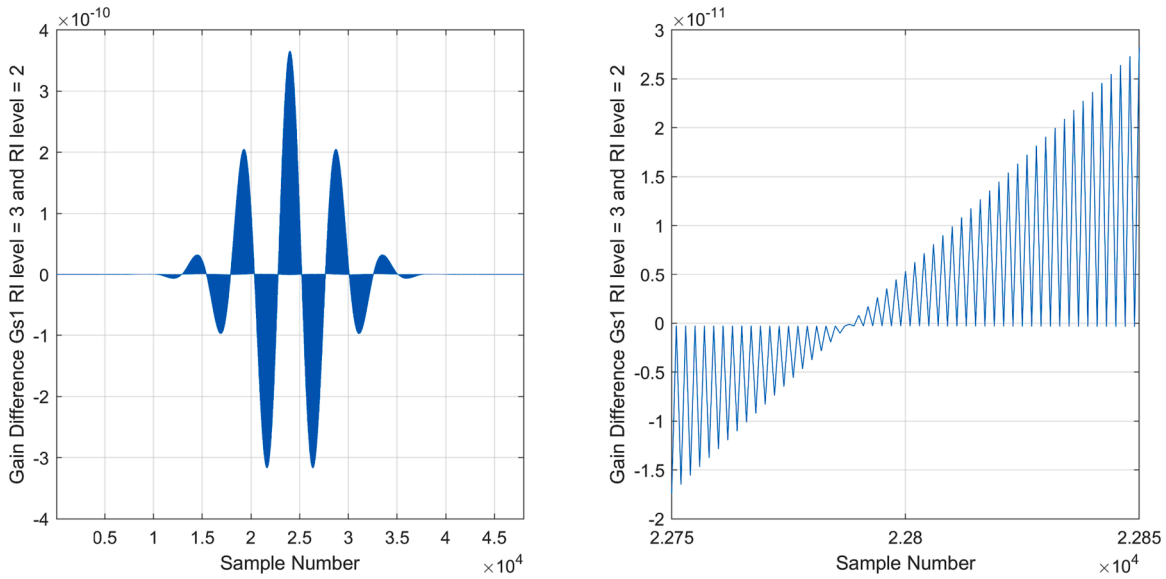

Fig. 26. Difference in *Gs1* windowing function value for RIL 2 and RIL 3: (left) complete windowing function; (right) in detail, centred around sample 22,800.

layer low pass filter concatenated to the tracking layer using 0.8 *m*, the total number of Prism integral lengths is approximately 26.5. For a desired total length of 48 k, this suggests approximately 1800 samples per *m* (n.b. irrespective of the sample rate to be used). As will be seen, the coefficients at start and end of the convolution are sufficiently small so that some truncation can be applied with no loss of precision, so that a higher value of *samples_per_m* can be used. A further consideration is that this number has to be some multiple of a low power of 2, so that Romberg Integration can be applied within the Prism integrals. In this design *samples_per_m* is selected as 2016, which has 32 as a factor. A Prism network consisting of the low pass filter and tracking Prisms of Fig. 22 is instantiated, where the integral window lengths are 2016 samples in the low pass filter, and 2520 samples in the tracking Prisms (8 is a factor of 2520). To create convolutional equivalents of the Prism network, a time series consisting of a single sample with a value of unity followed by a sequence of zeros is passed through this network. The resulting time series outputs from G_s and G_z are collected, and, after truncating to the desired length, these form the convolutional filters equivalent to the Prism network, and hence provide the desired windowing functions *Gs1* and *Gs2*. Of course, FFT windowing functions may be applied to any data set of appropriate length, where the sampling rate f_s is arbitrary. When applying Prism FFT to a specific data set, the corresponding value of *m* is f_s / *samples_per_m*. For the simulation example used in this paper, with f_s = 48 kHz and *samples_per_m* = 2016, *m* = 23.81 Hz (Fig. 21).

Fig. 24 shows the *Gs1* and *Gs2* windowing functions on linear and logarithmic scales. The wide dynamic range of the coefficients, with small values at beginning and end, enables truncation of the window to the desired length. Removing an equal number of samples from the start and end of the impulse response preserves the correct phase delay of the filter.

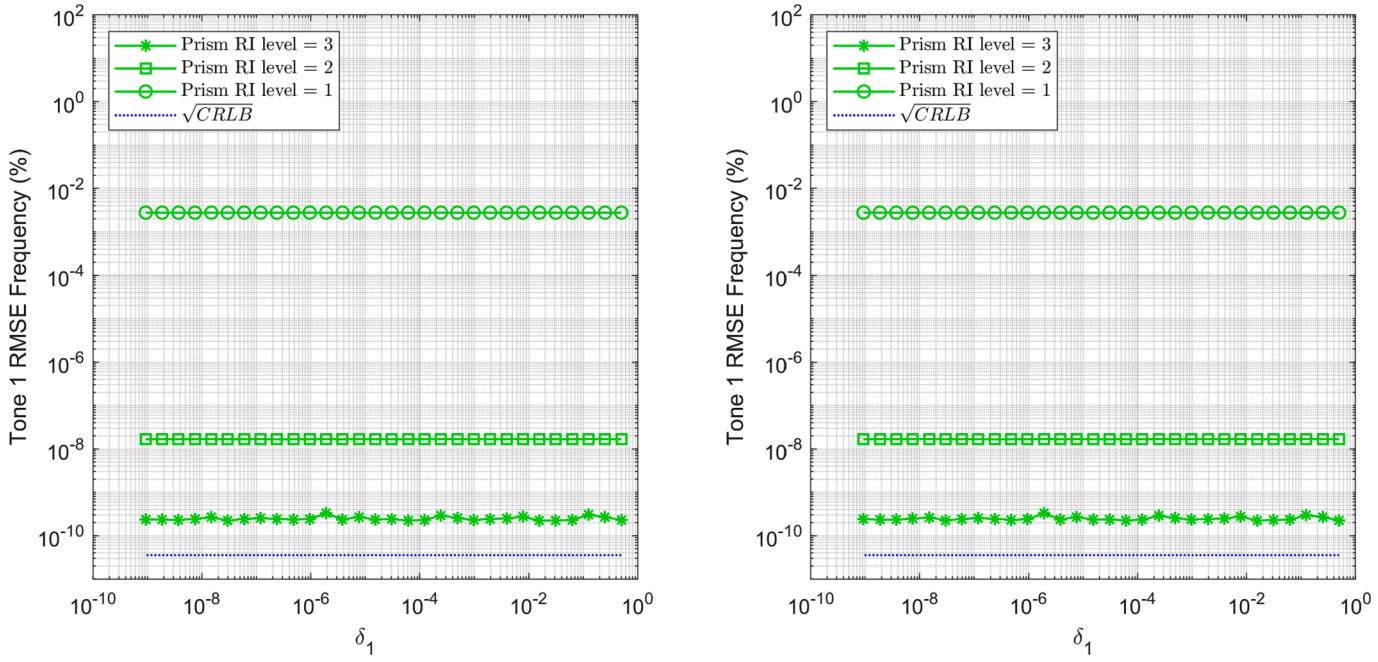

Fig. 27. FFT frequency errors for Tone 1 against $|\delta_1|$ for (left) $\delta_{j>1} = 0$; (right) $\delta_{j>1} \neq 0$; 1e-10 V s.d. white noise added; each result is based on 10,000 simulations. Compare with Fig. 9 (left) and 10 (right).

Figs. 25 and 26 demonstrate the results of applying different levels of Romberg Integration during the impulse response calculation of the windowing functions. In Fig. 25, the left-hand plot shows the windowing function generated using RIL 1, which superficially looks identical to the *Gs1* plot in Fig. 24, which uses RIL 3. However, the right-hand plot of Fig. 25 shows the difference in *Gs1* values generated using RIL 1 and RIL 2, indicating a difference that is roughly proportional to the *Gs1* value, with a scaling factor of approximately 4e-5. Fig. 26 shows the difference in *Gs1* value for RIL 2 and RIL 3. There is a negative correlation between the *Gs1* value of Fig. 24 and the difference shown in Fig. 26, on a smaller scale of approximately 4e-10. The appearance of shading in the left-hand plot of Fig. 26 is explained in the right-hand plot, which shows the same windowing function difference on a more detailed x-axis scale. The difference is not smooth, but has a complex, alternating pattern, jumping in consecutive samples between near zero and higher values. This intricate coefficient behaviour is generated by the interaction of trapezoidal integrals at multiple levels of granularity which are inherent to the RI technique.

The apparently small differences in windowing function value shown in Figs. 25 and 26 are wholly responsible for the substantial improvements in amplitude precision with increasing levels of RI shown in Figs. 9 - 12. There is no difference in the calculations used to generate the Prism FFT other than the values of the *Gs1* and *Gs2* windowing functions applied (this can be verified using the MATLAB example code provided with this paper, which includes windowing functions using RI = 1, 2 and 3). Thus, the change in windowing function from RIL 1 to RIL 2 delivers an amplitude error reduction by a factor of approximately 100,000, while the change from RIL 2 to RIL 3 provides a further improvement by a factor of 100. Similar improvements occur for the Prism calculation of frequency for Tone 1, as shown in Fig. 27, where the left and right plots show equivalent results to those of Figs. 9 and 10 respectively; here RIL 3 delivers $\sim 6.3\sqrt{CRLB}$ errors. The errors for phase (Figs. 9 and 10 right-hand plots) do not improve with RI level, as even for RIL 1 they are close to the CRLB. It is speculated that the arctan calculation (eqn. (17)), based on the sine and cosine heterodynes over the common windowing function *Gs1*, provides a large error cancelling effect on any precision errors introduced by *Gs1*. Fig. 28 shows the equivalent frequency results for Tone 8, where the signal-to-noise ratio is significantly lower. Here the RIL2 and RIL3 results coincide, while maintaining a value of $\sim 6.3\sqrt{CRLB}$.

Conventional FFT windows are constructed from analytical functions, typically weighted cosines, designed primarily to minimize spectral leakage by providing a rapid drop in gain (Fig. 6), but at the expense of optimal precision, as demonstrated by the error saturations shown in Figs. 9-12. The Prism FFT method uses non-analytic, indeed non-smooth windowing functions (Fig. 26), generated via RI, which minimize precision loss.

6. Simulation results using varying levels of noise

In this paper so far, simulation results have been presented with a low level of additive white noise, so that the various structural limitations of conventional FFT techniques can be highlighted. In this section, the same simulation example is used with increasing levels of noise. This provides a broader comparison of the Prism and conventional FFT methods over more practical noise levels, and in particular demonstrates the robustness of the Prism technique.

A series of simulations were carried out using the realistic condition $\delta_{\forall j} \neq 0$, i.e. the value of δ for each peak was randomly selected from the range [-0.5, 0.5]. Additive white noise with standard deviations ranging from 1e-10 V up to 1e-1 V, increasing logarithmically by a factor of $\sqrt{10}$ at each step, was included in the simulation. The RMS errors of the Prism FFT technique, together with those of the Blackman FFT (overall the best performing conventional windowing function for this example), were collated for each tone and for each noise level, over 10,000 random simulations.

As a reference, the CRLB of each parameter estimate (eqns. (2) – (4), for each tone at each noise level, was also calculated. The results of the simulations for frequency, amplitude and phase are shown in Figs. 29 – 31 respectively. In each case, the RMS errors for the Blackman and Prism FFT results are shown as percentages and as a proportion of $\sqrt{CRLB}$. Different plotting symbols are used to indicate the different tones to assist visualisation of data trends. As the level of noise increases, the FFT noise floor rises, and lower amplitude tones become hidden, including for the Prism technique. For clarity, therefore, results are not plotted where the true tone amplitude is less than 10 × noise standard deviation, for both Blackman and Prism results.

Fig. 29 shows the results for frequency estimates. With $\delta_{\forall j} \neq 0$, the most significant source of error for the Blackman frequency estimate is δ itself. The RMS errors (top left) are constant at ~ 29 %, for all tones and noise levels, reproducing the results of Table 2 over a wider parameter

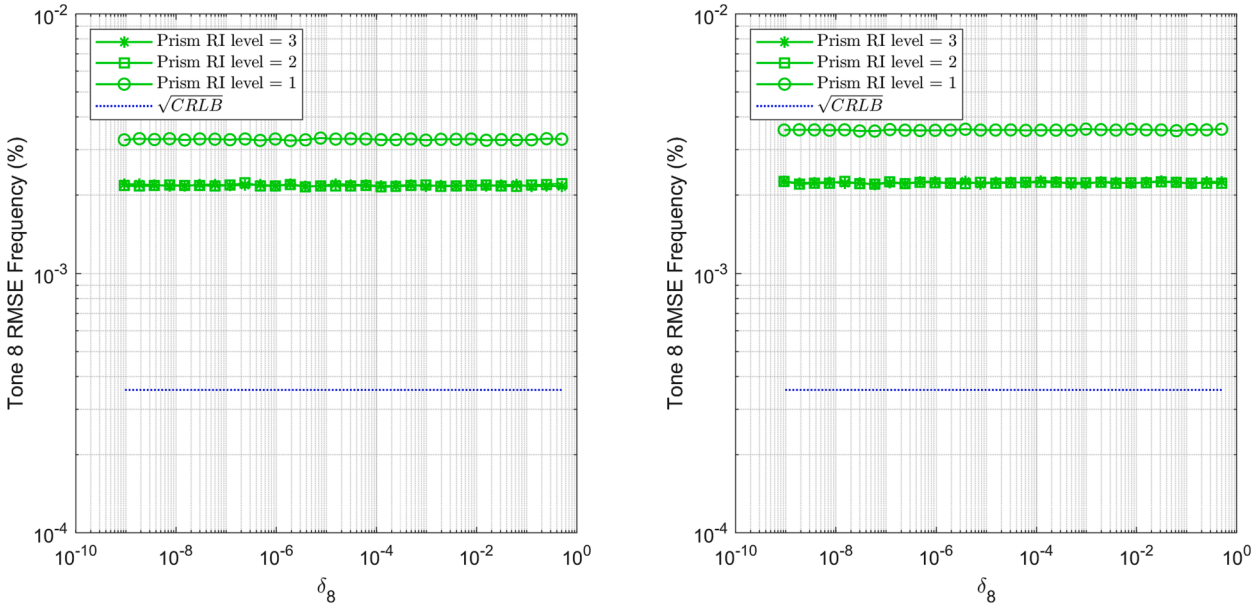

Fig. 28. FFT frequency errors for Tone 8 against $|\delta_1|$ for (left) $\delta_{j>1} = 0$; (right) $\delta_{j>1} \neq 0$; 1e-10 V s.d. white noise added; each result is based on 10,000 simulations. Compare with Fig. 11 (left) and 12 (right).

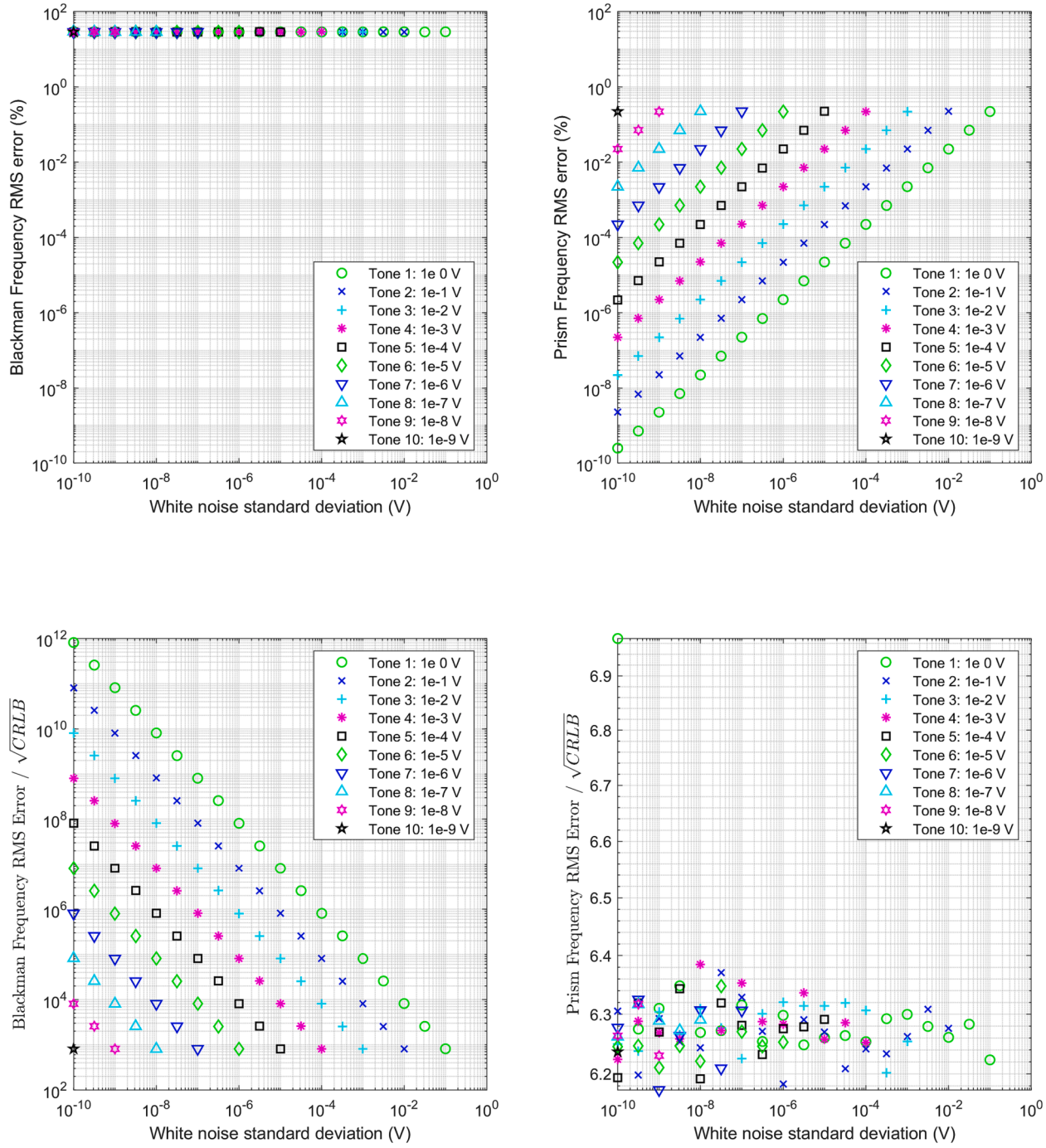

Fig. 29. FFT frequency RMS errors for all tones against varying white noise standard deviation. (top left) Blackman RMS errors; (top right) Prism RMS errors; (bottom left) Blackman RMS errors scaled by $\sqrt{CRLB}$; (bottom right) Prism RMS errors scaled by $\sqrt{CRLB}$; each points represented 10,000 simulations.

range. Scaled to the $\sqrt{CRLB}$(bottom left), the errors are no less than 1e3, rising to 1e12 for the highest tone-to-noise ratio. The Prism RMS errors (top right) show a linear dependency on the tone amplitude-to-noise ratio for all results. Scaled to the $\sqrt{CRLB}$(bottom left), the errors are approximately constant at around 6.3 $\sqrt{CRLB}$, across nine orders of magnitude of increasing noise. The only notable departure is the error for Tone 1 at the lowest noise std, which rises to approximately 7 $\sqrt{CRLB}$, suggesting a lower error saturation limit that might be addressed using a higher RI level.

Fig. 30 shows the results obtained for the amplitude estimates. The most significant source of error for the Blackman method is also δ, so the RMS errors (top left) are relatively constant multiples (approximately 5%) of the corresponding tone amplitude, irrespective of noise standard deviation. The exceptions are the lowest two tones which remain hidden after the application of the Blackman window, resulting in higher proportional errors. Scaled to the $\sqrt{CRLB}$(bottom left), the Blackman errors have a minimum of approximately 100 $\sqrt{CRLB}$, rising to around 1e11 $\sqrt{CRLB}$ for the highest tone-to-noise ratio. The Prism RMS errors (top right) once again show a linear dependency on the tone amplitude-to-noise ratio. On the $\sqrt{CRLB}$ scale (bottom left), the errors are steady at ~ 2.54 $\sqrt{CRLB}$ for all results, with a slightly higher result for Tone 1 at the lowest noise std.

Fig. 31 shows the results for the phase estimates, which are more nuanced. For high tone-to-noise ratios, above approximately 100, the

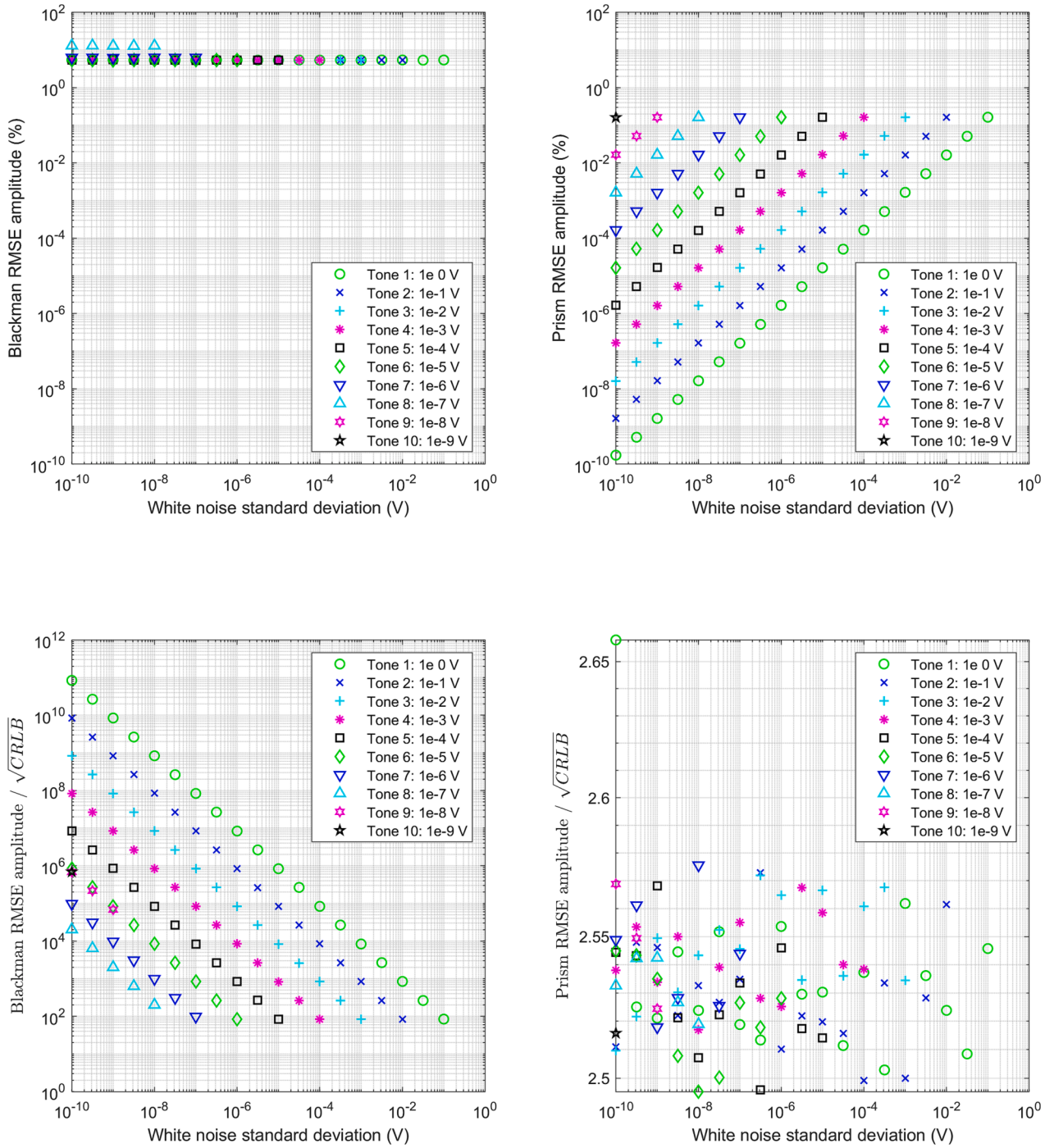

Fig. 30. FFT amplitude RMS errors for all tones against varying white noise standard deviation. (top left) Blackman RMS errors; (top right) Prism RMS errors; (bottom left) Blackman RMS errors scaled by $\sqrt{CRLB}$; (bottom right) Prism RMS errors scaled by $\sqrt{CRLB}$; each points represented 10,000 simulations.

Blackman phase errors (top left) have reasonably low values (in the range 1e-4 to 1e-2 %) and show little variation with the level of noise. At lower amplitude-to-noise ratios, the RMS errors develop a linear relationship with the amplitude-to-noise. This can be explained with reference to the $\sqrt{CRLB}$ ratio (bottom left): with increasing noise, this ratio drops rapidly towards one, reaching a lower limit of around 1.4. Once this lowest value is reached, the relationship between phase error and noise level becomes linear (upper left). The Prism RMS phase errors (top right), like those for frequency and amplitude, show a linear dependency on the amplitude-to-noise ratio for all results. Scaled to the $\sqrt{CRLB}$(bottom left), the errors are more or less constant at around 2.5 $\sqrt{CRLB}$, other than for the highest amplitude-to-noise ratio results.

Significantly, at high noise levels, some Blackman phase errors are lower than the Prism equivalents. Comparing the two top graphs in Fig. 30, no Blackman errors exceed 2e-2 % for Tones 1 – 6 whereas several of the Prism errors exceed this value. Equivalently, some Blackman high noise results having a $\sqrt{CRLB}$ ratio as low as 1.4, while the corresponding Prism ratios are around 2.5 or higher. This better Blackman performance can be explained by two effects: the relative insensitivity of phase errors to variations in δ, and the relatively wide bandwidth of the Prism window function passband compared to that of the Blackman (Fig. 6). The 12 layer, high attenuation Prism filter is

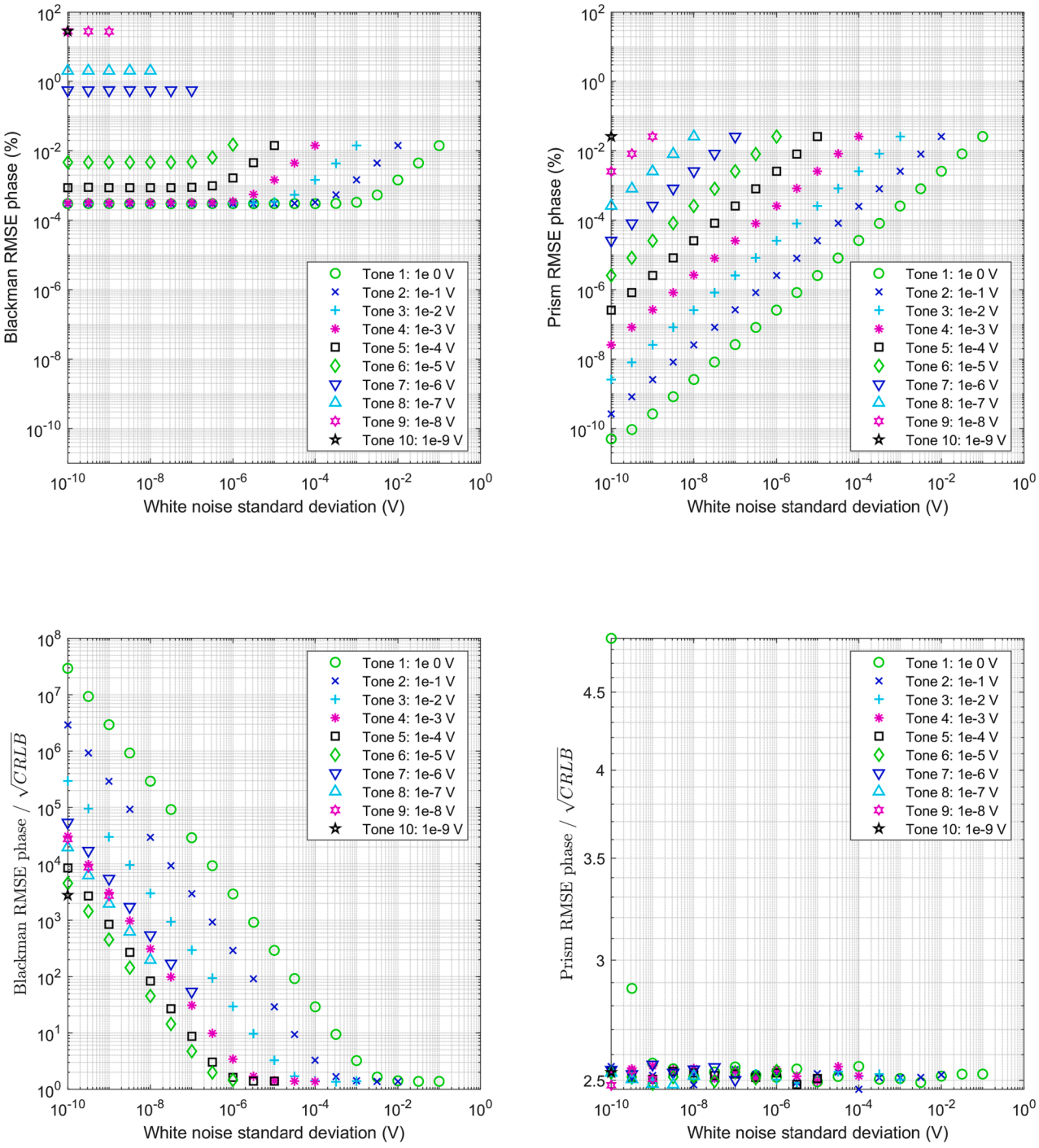

Fig. 31. FFT phase RMS errors for all tones against varying white noise standard deviation. (top left) Blackman RMS errors; (top right) Prism RMS errors; (bottom left) Blackman RMS errors scaled by $\sqrt{CRLB}$; (bottom right) Prism RMS errors scaled by $\sqrt{CRLB}$; each points represented 10,000 simulations.

designed for the detection of low amplitude components in a low noise environment; at higher noise levels an alternative design with fewer layers, less attenuation, and a narrower passband, would provide better performance with a lower $\sqrt{CRLB}$ ratio, even without the additional calculation step of IpFFT.

7. Summary, conclusions and further work

This paper has presented a new approach to calculating the Fast Fourier Transform using Prism Signal Processing. It includes the following novel aspects:

- Two windowing functions are used, enabling the simple and direct calculation of tone frequency, and thereby amplitude and phase.
- The windowing functions are the non-analytic and non-smooth convolutional equivalents of a Prism network implementing Romberg Integration to provide high precision outputs.
- Current FFT techniques focus on the bin: windowing functions are designed to reduce spectral leakage to as narrow a band as possible, while peak location is concerned with identifying the nearest bin and, with more sophisticated methods, calculating the value of δ within the bin. In the Prism FFT the focus is the lobe; any peak

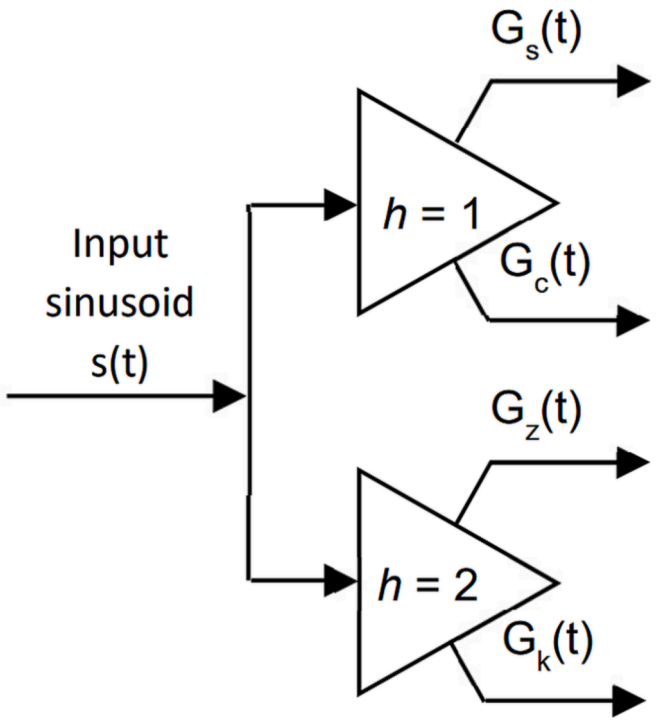

Fig. 32. Tracking of a pure sinusoid input s(t) using two Prisms.

isolated within the lobe can be located to high precision via a simple calculation.

In the simulation used throughout this paper, the Prism FFT has outperformed conventional windowing functions by several orders of magnitude in most, but not all, examples. Limitations include the following:

- By using two windowing functions, the computational cost of the Prism FFT is (at least) double that of a conventional FFT calculation. However, there is scope for reducing this cost. While for clarity the *Gs1* and *Gs2* windowing functions have been given equal prominence in this paper, in practice most of the required information is derived from *Gs1;* the only role of *Gs2* is in the calculation of *r*. Accordingly, the use of *Gs2* could be restricted to performing frequency-specific 'interrogations' (eqns (7) – (9)) around potential peaks located in the *Gs1* spectrum. If the expected number of peaks is less than log(*n*) then the full FFT calculation of the *Gs2* windowed data may be neither efficient nor necessary.
- An inherent feature of Prism signal processing is zero gain at zero Hz; the windowing functions have peak gains offset from zero Hz, which generates double peaks in the raw FFT spectra. This has a number of side-effects. It complicates peak location, particularly when there are several in close proximity. It also requires double peaks to be hidden via dead zones in Prism FFT plots.
- The Prism windowing functions have lower attenuation rates compared with those of conventional WCW windows; accordingly, for poor signal-to-noise ratios, Prism performance may not always improve on that of a WCW method (e.g. in Fig. 28). This issue can be managed by appropriate window function design (e.g. fewer Prism layers providing a narrower passband for lower SNR data sets).
- The Prism lobe is wide – for the 12 layer windowing functions used in the example, spectral tones are assumed to be separated by at least 48 Hz in order to facility a simple and high precision calculation. Alternative, narrower Prism filter designs may be developed, and/or further Prism techniques such as dynamic notch filtering ([22,23]) may be used to provide additional discrimination within an individual lobe.
- There is a minimum practical Prism FFT window length. Some IpFFT techniques can operate on signal lengths as short as 128 samples (e.g. [15]). By contrast, to provide reasonable RI performance, the minimum value of *samples_per_m* is 32. The corresponding length of a 12 layer lowpass filter would be approximately 840 samples. In such a design, however, the dead-zone around each peak would be a significant proportion of the full signal bandwidth, and few peaks could be detected. Design tradeoffs such as using fewer layers in the low pass filter may be considered, but as general guidance the Prism FFT would be difficult to use effectively on data sets$<$1000 samples. For shorter data sets and/or where real-time tracking is required, the original recursive FIR Prism implementation may be more suitable [4].

While this paper has only provided simulation results, the Prism FFT technique has been applied to several instrumentation applications, where high precision results are obtained. For example, in [9] it is used to analyse data generated in a prototype Coriolis mass flowmeter, operating in two modes of vibration. The resulting spectra are feature-rich, with many components arising from harmonics and beat frequencies of the two modes. A Coriolis meter typically has two sensors monitoring the flowtube vibration in different positions; in [9] the Prism spectra from the two sensors are compared. Peak frequency estimates show agreement to within 10 nHz for the highest amplitude components. Such precision supports accurate identification of the source of each peak, which for the experimental data presented, including the vibration frequency of an adjacent flowmeter.

Overall, the Prism FFT technique is proposed as a simple, low cost, but precise alternative to the WCW methods currently employed throughout science and engineering. While it does not perform to the CRLB limits, for the examples provided it reaches to within an order of magnitude of the CRLB for all parameters. The precision and numerical stability of the technique is demonstrated by the delivery of the same CRLB performance for peaks with nine orders of magnitude variation in amplitude over nine orders of magnitude variation in additive white noise. This is likely to be acceptable precision for many applications, while providing an excellent starting point (e.g. via an initial estimate of δ) for more sophisticated analysis where the highest precision is required.

A particular attraction is its methodological simplicity: equations (6) – (10) are readily understood and implemented, while most of the mathematical sophistication is restricted to the design of the windowing functions *Gs1* and *Gs2*. Provision of the numerical values of the *Gs1* and *Gs2* coefficients for a particular window length, and the associated value of *samples_per_m*, is sufficient to enable the spectral analysis of any suitable data set. Accordingly, it is hoped the Prism FFT may prove beneficial in a wide range of scientific and engineering applications. For example, its delivery of high precision FFT results over a wide range of signal-to-noise ratios may encourage the development of improved instrumentation, for example through the use of higher precision ADCs [24].

There are many potential extensions to this technique. These include:

- The detection and accurate location of multiple peaks within the Prism lobe. Preliminary results suggest that, under the same assumptions of the current simulation example, two peaks separated by only 3 Hz can be readily and precisely located. This technique will be described in a future paper.
- The Sparse Fourier Transform [1]; here the low cost and high frequency and phase precision of the Prism FFT could support high levels of data reduction.
- The introduction of Romberg Integration techniques into other filter design methods.

A simple Matlab implementation of the Prism FFT calculation is provided in the supplementary material to this paper.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Example code is attached.

Acknowledgements

This paper is dedicated to the memory of John Wilmott, the author's undergraduate and DPhil supervisor, who passed away on 2nd January 2023. He inspired a deep interest in computational and numerical methods which is reflected in this paper.

Appendix A:. Calculation of Frequency, amplitude and phase from Prism outputs

This Appendix explains how the FAP parameters of an assumed pure sinusoid (isolated by prior pre-filtering as shown in Fig. 22) can be calculated where the sinusoid provides the input to two Prisms, as illustrated in Fig. 32.

The two Prisms have a common characteristic frequency m but harmonic number h values of 1 and 2 respectively. For notational convenience, the outputs of the $h = 1$ Prism are labelled simply $G_s(t)$ and $G_c(t)$, while the outputs of the $h = 2$ Prism are labelled $G_z(t)$ and $G_k(t)$. The input signal s(t) is:

$$s(t) = A\sin(\varphi(t)) \tag{A1}$$

$$\varphi(t) = 2\pi ft + \varphi_o \tag{A2}$$

where t is time, A is amplitude, $\varphi(t)$ is the instantaneous phase, f is the frequency and φ_o is the initial phase at $t = 0$. Assuming A and f are constant, then, as described in [4], the Prism outputs are given by the analytic expressions:

$$G_s^h(t) = A\mathrm{sinc}^2(r)\frac{r^2}{r^2 - h^2}sin(\varphi(t) - 2\pi r) \tag{A3}$$

and

$$G_c^h(t) = A\mathrm{sinc}^2(r)\frac{hr}{r^2 - h^2}cos(\varphi(t) - 2\pi r) \tag{A4}$$

where $r = f/m$ is the dimensionless frequency corresponding to f, and where it is assumed that $r \in [0,1]$.

Specifically, for the values $h = 1$ and 2 respectively, we obtain:

$$G_s(t) = A\mathrm{sinc}^2(r)\frac{r^2}{r^2 - 1}sin(\varphi(t) - 2\pi r) \tag{A5}$$

$$G_c(t) = A\mathrm{sinc}^2(r)\frac{r}{r^2 - 1}\cos(\varphi(t) - 2\pi r) \tag{A6}$$

$$G_z(t) = A\mathrm{sinc}^2(r)\frac{r^2}{r^2 - 4}sin(\varphi(t) - 2\pi r) \tag{A7}$$

$$G_k(t) = A\mathrm{sinc}^2(r)\frac{2r}{r^2 - 4}\cos(\varphi(t) - 2\pi r) \tag{A8}$$

To track the FAP parameters, the first step is to calculate an estimate of r, via an intermediary variable x, an amplitude ratio, defined as:

$$x(t) = \sqrt{\frac{4G_z^2(t) + G_k^2(t)}{G_s^2(t) + G_c^2(t)}} \tag{A9}$$

Substituting the analytical results above:

$$x(t) = \frac{2(r^2 - 1)}{(r^2 - 4)} \tag{A10}$$

Accordingly, during a tracking operation r can be estimated using:

$$\widehat{r}(t) = \sqrt{\frac{4x(t) - 2}{x(t) - 2}} \tag{A11}$$

from which corresponding estimates of frequency, amplitude and mid-window phase of the tracked component can be calculated as follows:

$$\widehat{f} = \widehat{r}m \tag{A12}$$

$$\widehat{A} = \left[\frac{\pi}{sin(\pi\widehat{r})}\right]^2\left[1 - \widehat{r}^2\right]\sqrt{G_s^2 + \widehat{r}^2G_c^2} \tag{A13}$$

$$\widehat{\varphi} = \mathrm{atan2}(-G_s, \widehat{r}G_c) \tag{A14}$$

In the Prism FFT calculation (Fig. 21), a slightly different calculation of r is used. The FFT calculation efficiently generates both cosine (real) and sine (imaginary) outputs, only two windowed data sets are required. The real and imaginary results from the Gs1 and Gs2 windows are orthogonal but

have the same gains. For any particular FFT bin frequency in the vicinity of a peak of amplitude A, the Gs1 and Gs1 real and imaginary values are equivalent to:

$$G_{s\_FFT} = Asinc^2(r)\frac{r^2}{r^2-1}sin(\varphi_{mid} - 2\pi r) \tag{A15}$$

$$G_{c\_FFT} = Asinc^2(r)\frac{r^2}{r^2-1}\cos(\varphi_{mid} - 2\pi r) \tag{A16}$$

$$G_{z\_FFT} = Asinc^2(r)\frac{r^2}{r^2-4}sin(\varphi_{mid} - 2\pi r) \tag{A17}$$

$$G_{k\_FFT} = Asinc^2(r)\frac{r^2}{r^2-4}\cos(\varphi_{mid} - 2\pi r) \tag{A18}$$

The Gs1 and Gs2 amplitudes are

$$A_1 = \sqrt{G_{s\_FFT}^2 + G_{c\_FFT}^2} = Asinc^2(r)\frac{r^2}{r^2-1} \tag{A19}$$

$$A_2 = \sqrt{G_{z\_FFT}^2 + G_{k\_FFT}^2} = Asinc^2(r)\frac{r^2}{r^2-4} \tag{A20}$$

Accordingly, the amplitude ratio x in this case is

$$x = \frac{A_2}{A_1} = \frac{(r^2-1)}{(r^2-4)} \tag{A21}$$

so r may be estimated using:

$$\widehat{r}\,(t) = \sqrt{\frac{4x-1}{x-1}} \tag{A22}$$

The subsequent calculations of frequency, amplitude and phase are straightforward once the value of r has been established.

Appendix B. Supplementary material

Supplementary data to this article can be found online at https://doi.org/10.1016/j.measurement.2023.113372.

References

[1] D. Schneider, A faster fast Fourier transform, IEEE Spectr. (2012).
[2] P. Duhamel, M. Vetterli, "Fast Fourier Transforms: a tutorial review and a state of the art, Signal Process. 19 (1990).
[3] M.P. Henry, Spectral analysis techniques using Prism signal processing, Measurement 169 (2021).
[4] M.P. Henry, "The Prism: Recursive FIR Signal Processing for Instrumentation Applications" IEEE Trans, Instr. Meas. 69 (2020).
[5] J. Dutka, Richardson Extrapolation and Romberg Integration, Hist. Math. 11 (1984) 3–21.
[6] M.P. Henry, Building Billion Tap Filters, IEEE Ind. Electron. Mag. (Nov 2020).
[7] Henry, MP. "Low Pass, Low Cost Prism Filters", 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, Rome, June 2020.
[8] M.P. Henry, An ultra-precise fast fourier transform, Science Talks 4 (2022), 100097.
[9] M.P. Henry, Enhanced measurement validation via ultra-precise spectral analysis, Science Talks 5 (2022), 100094.
[10] C. Offelli, D. Petri, Interpolation techniques for real-time multifrequency waveform analysis, IEEE Trans. Instrum. Meas. 39 (1) (Feb. 1990) 106–111.
[11] Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform", Proc. IEEE, 66 (1978), 51-83.
[12] A. Nuttall, Some windows with very good sidelobe behavior, IEEE Trans. Acoust. Speech Signal Process. 29 (1) (February 1981) 84–91.
[13] R. Lyons, Understanding Digital Signal Processing, 3rd Edition,, Prentice Hall, 2011.
[14] E. Aboutanios, B. Mulgrew, Iterative Frequency Estimation by Interpolation on Fourier Coefficients, IEEE Transactions on Signal Pocessing 53 (2005) 1237–1242.
[15] C. Liguori, A. Paolillo, A. Pignotti, Estimation of Signal Parameters in the Frequency Domain in the Presence of Harmonic Interference: A Comparative Analysis, IEEE Trans. Instrum. Meas. 55 (2006) 562–569.
[16] C. Liguori, A. Paolillo, IPFFTC-Based Procedure for Hidden Tone Detection, IEEE Trans. Instrum. Meas. 56 (2007) 133–139.
[17] D. Belega, D. Dallet, "Multifrequency signal analysis by Interpolated DFT method with maximum sidelobe decay windows, Measurement 42 (2009) 420–426.
[18] D. Belega, D. Petri, D. Dallet, Accurate amplitude and phase estimation of noisy sine-waves via two-point interpolated DTFT algorithms, Measurement 127 (2018).
[19] D. Belega, D. Petri, Influence of the noise on DFT-based sine-wave frequency and amplitude Estimators, Measurement 137 (2019) 527–534.
[20] D. Belega, D. Petri, Fast procedures for accurate parameter estimation of sine-waves affected by noise and harmonic distortion, Digital Signal Process. 114 (2021), 103035.
[21] Davis, P.J, Rabinowitz, R, Methods of Numerical Integration", Courier Corporation, 2007, ISBN 0486453391.
[22] M.P. Henry, F. Leach, M. Davy, O. Bushuev, M.S. Tombs, F.B. Zhou, S. Karout, The Prism: Efficient Signal Processing for the Internet of Things, IEEE Ind. Electron. Mag. (Dec 2017) 22–32.
[23] M.P. Henry, F.B. Zhou, M.S. Tombs, F. Leach, M. Davy, M. Malladi, Prism Signal Processing of Coriolis Meter Data for Gasoline Fuel Injection Monitoring, Flow Meas. Instrum. (Oct 2019).
[24] G. Betta, C. Liguori, A. Pietrosanto, Propagation of uncertainty in a discrete Fourier transform algorithm, Measurement 27 (4) (2000) 231–239.
[25] C. Offelli, D. Petri, The influence of windowing on the accuracy of multifrequency signal parameter estimation, IEEE Trans. Instrum. Meas. 41 (2) (1992) pp.