Copyright Statement

Investigation into digital audio equaliser systems and the effects of arithmetic and transform errors on performance

by

Robin John Clark

A thesis submitted to the University of Plymouth
In partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

Department of Communication and Electronic Engineering
Faculty of Technology

In collaboration with
Allen & Heath Limited

April 2001

Investigation into digital audio equaliser systems and the effects of arithmetic and transform errors on performance

Robin John Clark

Abstract

Discrete-time audio equalisers introduce a variety of undesirable artefacts into audio mixing systems, namely, distortions caused by finite wordlength constraints, frequency response distortion due to coefficient calculation and signal disturbances that arise from real-time coefficient update. An understanding of these artefacts is important in the design of computationally affordable, good quality equalisers. A detailed investigation into these artefacts using various forms of arithmetic, filter frequency response, input excitation and sampling frequencies is described in this thesis.

Novel coefficient calculation techniques, based on the matched z-transform (MZT) were developed to minimise filter response distortion and computation for on-line implementation. It was found that MZT-based filter responses can approximate more closely to s-plane filters, than BZT-based filters, with an affordable increase in computation load. Frequency response distortions and prewarping/correction schemes at higher sampling frequencies (96 and 192 kHz) were also assessed.

An environment for emulating fractional quantisation in fixed and floating point arithmetic was developed. Various key filter topologies were emulated in fixed and floating point arithmetic using various input stimuli and frequency responses. The work provides detailed objective information and an understanding of the behaviour of key topologies in fixed and floating point arithmetic and the effects of input excitation and sampling frequency.

Signal disturbance behaviour in key filter topologies during coefficient update was investigated through the implementation of various coefficient update scenarios. Input stimuli and specific frequency response changes that produce worst-case disturbances were identified, providing an analytical understanding of disturbance behaviour in various topologies. Existing parameter and coefficient interpolation algorithms were implemented and assessed under finite wordlength arithmetic. The disturbance behaviour of various topologies at higher sampling frequencies was examined.

The work contributes to the understanding of artefacts in audio equaliser implementation. The study of artefacts at the sampling frequencies of 48, 96 and 192 kHz has implications in the assessment of equaliser performance at higher sampling frequencies.

List of Contents

6 Topology behaviour during coefficient update 152

List of Figures

List of Tables

List of Abbreviations and Glossary

AES	Audio Engineering Society
bell	filter type in audio equalisers
BH4	Blackman-Harris 4 windowing function
boost	gain setting in a filter
BZT	bilinear z-transform
cut	attenuation setting in a filter
dB	decibel
dBFS	decibel full-scale
dc	direct current (zero Hertz)
DF1	direct form 1 filter topology
DF2	direct form 2 filter topology
DF1T	direct form 1 transposed filter topology
DF2T	direct form 2 transposed filter topology
DSP	digital signal processing
Fc	tuned frequency (corner or centre frequency)
FFT	fast Fourier transform
Fs	sampling frequency
G	gain parameter in dB
HF	high frequency
LF	low frequency
MZT	matched z-transform
Q	Quality factor (bandwidth)
RMS	root mean square
shelving	filter type in audio equalisers
THD+N	total harmonic distortion plus noise
Ts	sampling interval

Acknowledgments

I would like to thank my Director of studies Professor Emmanuel Ifeachor for his excellent guidance, encouragement and patience throughout this project. I would also like to thank my second supervisor Glenn Rogers for his support and continuous enthusiasm in my research.

I would also like to thank the following people:
Peter Van Eetvelt for his tutorials in developing closed form expressions and function approximation. Silvio Gehri for suggesting the concept of averaging coefficient sets. Ian Dennis and Julian Dunn for discussions in spectrum analysis and the principles of intermodulation distortion measurements. Rhonda Wilson, Udo Zölzer and James Moorer for answering my questions on their work referenced in this thesis. Mike Williamson for his encouragement during the writing of this thesis.

The research was partly funded through an industrial fellowship awarded by the Royal Commission for the Exhibition of 1851. This award has provided an ideal opportunity for British industry to benefit from an academic research project.

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.

This study was partly funded through an industrial fellowship awarded by the Royal Commission for the Exhibition of 1851. The study was carried out in collaboration with Allen and Heath.

Publications

Clark R. J., Ifeachor E. C., Rogers G. M., The study of arithmetic and wordlength requirements for digital audio filtering hardware. Preprint 4100, 99th Audio Engineering Society Convention , 1995.

Clark R. J., Ifeachor E. C., Rogers G. M., Real-time equaliser coefficient realisation with minimised computational load and distortion. Preprint 4360, 101st Audio Engineering Society Convention , 1996.

Clark R. J., Ifeachor E. C., Rogers G. M., Van Eetvelt P. W. J., Techniques for generating digital equalizer coefficients. Journal of the Audio Engineering Society, vol. 48, no. 4, April 2000.

Signed …….

Date ……..

1 Introduction

1.1 Frequency response equalisation in audio mixing systems

Frequency response equalisation is an important aspect of all audio mixing applications such as public address, recording, broadcast and live performance. Frequency response equalisation is typically performed on every audio channel in mixing systems. A typical audio channel equaliser, termed 'parametric equaliser', consists of three to six cascaded stages of filtering. Typically, each filter stage is used to provide control of a particular spectral region of the audio signal. Various filter types are selectable for each stage, for example, low pass, high pass, peak (bell) and shelving filter functions. A user interface enables real time control of the parameters of the filters (for example centre frequency, Q factor and Gain). By manipulating the filter parameters the user can modify the frequency response of an audio channel to provide desirable subjective or objective spectral changes to the audio signal. Subjective changes are made to 'enhance' or to create an 'effect' in a particular spectral region of the audio signal. Objective modifications to the frequency response are typically used to control a particular spectral band in the audio signal due to noise interference or for transmission purposes.

The commercial success of the storage of audio information on digital media (compact disc, digital audio tape and hard disc recording) has led to increased use of digital audio in mixing applications. Furthermore, digital audio mixing systems are becoming popular,

due to digital audio interfacing (Audio Engineering Society, 1992) and the benefits of computer control which allows, for example, snap-shot audio scene change at the touch of a button. However, the high cost of digital mixer technology has in the past led to inferior performance, compared to analogue mixers. Factors that contributed to this include: poor ergonomics due to design constraints of the digital control surface; noise and distortion components due to inadequate audio signal wordlengths in the analogue to digital conversion and signal processing; audio bandwidth limitations due to the adopted sampling frequencies of 44.1 and 48 kHz.

1.2 Advances in digital audio equalisation

Digital signal processing techniques have been exploited to implement novel equalisation schemes that were not feasible in analogue systems. Linear phase, finite impulse response (FIR) filters have been used for equalisation applications (Kraght, 1992; Lian and Lim 1993). Novel speaker equalisation systems using high order frequency response equalisation curves to correct speaker frequency responses have been developed (Greenfield, 1991; Hawksford, 1997). Digital adaptive filtering techniques have been used in echo cancellation and acoustic feedback attenuation to improve intelligibility in public address applications (Kamerling et al, 1998). Despite these advances in digital audio equalisation, there is still a requirement to replicate analogue parametric equalisers with minimum phase characteristics.

Recent advances in digital signal processor (DSP) technology have made floating point devices available to the audio industry. Various floating point formats have been implemented. For example, a twos-complement binary coded format (Texas Instruments, 1992) and a sign magnitude binary coded format (Analog Devices, 1997) are available. In addition, fixed point processors that enable efficient double precision arithmetic (Motorola, 1999) are emerging. However there is still no simple way of specifying

internal signal processing resolution in commercial digital audio systems. Audio systems utilise a variety of wordlengths, e.g 24, 32, 40, 48 and 56 bits, and employ single, extended and double precision fixed or floating point arithmetic. Specifications of commercial systems often detail internal operating wordlengths at different parts of the arithmetic unit. In addition, numerous implementation techniques are used to realise operational digital filters. These filters make it difficult to compare arithmetic specifications of digital equaliser systems.

The higher sampling frequencies of 96 kHz and 192 kHz were standardised in the Audio Engineering Society (1999). The introduction of the higher sampling frequencies has caused debate about human auditory perception beyond 20 kHz. Due to the design constraints of digital audio equipment, it is difficult to quantify whether the extended audio bandwidth (beyond 20 kHz), facilitated by the higher sampling rates, is the sole reason for improved audio reproduction. Some digital equalisers operating at a sampling rate of 96 kHz have been reported to sound inferior to analogue equalisers and digital equalisers operating at a sampling rate of 48 kHz. Problems associated with digital equaliser quality are not always associated with the constraints of the Nyquist frequency limitation. Common criticisms of digital equalisation include noise and harmonic distortion products, transient distortion during parameter change, and high frequency response.

A primary aim of this project is to investigate the distortions associated with the efficient implementation of discrete-time equalisers, paying particular attention to system sampling rates, finite wordlength arithmetic and the coefficient calculation process.

1.3 Research issues in digital equaliser design

Figure 1-1 depicts the key parts of a digital equaliser system. In practice the mixing system operator specifies a desired frequency response by manipulating the user interface

(control surface). This produces a set of filter parameters which are used to calculate appropriate digital filter coefficients. The discrete-time varying filter implementation applies the specified frequency response to the audio signal.

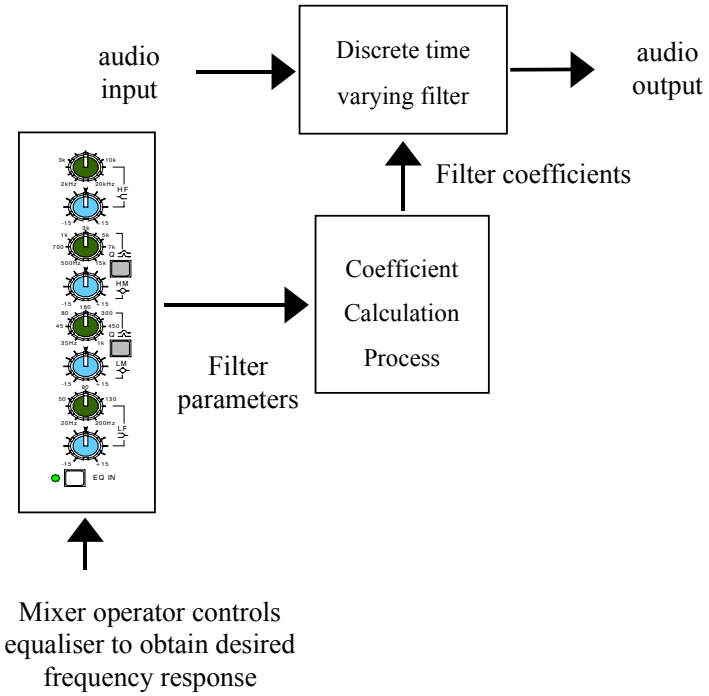

Figure 1-1 A digital equaliser system

1.3.1 Coefficient calculation techniques

On-line digital equaliser coefficient calculation provides greater parameter resolution and filter type selection than off-line coefficient generation schemes (look-up tables). However, this increases computational load and can produce inferior filter responses compared to off-line coefficient calculation. With the increasing power of DSP technology, it is feasible and beneficial to implement both on-line coefficient calculation and the filtering functions required for equalisation on the same processor. In this case, it is necessary to optimise the computational load for coefficient calculation and minimise the distortion in the frequency response. Frequency response distortion is taken here to mean the difference between the magnitude and phase responses of the target filter (the

ideal s-plane response) and that of the resulting discrete-time filter. Techniques for on-line calculation of equaliser coefficients have been described extensively. In techniques based on the Matched z-transform (MZT) (M^{c}Nally, 1979; Hirata, 1980), distortions due to reflections about the sampling frequency were not considered. In Bilinear z-transform (BZT) based techniques, a number of pre-warping schemes have been developed to reduce the response distortion that is inherent in the BZT (Moorer, 1983; White, 1985; Shpak, 1992; Bristow-Johnson, 1994; Orfanidis, 1996). However, the response distortions produced by the different BZT based techniques have not been compared.

1.3.2 Finite wordlength noise analysis of filters

The three main finite wordlength effects associated with digital filter implementation are coefficient quantisation, arithmetic overflow and arithmetic quantisation.

Coefficient quantisation is necessary because of the finite wordlength of the coefficient storage elements. Coefficient quantisation affects the pole and zero positions on the z-plane, leading to a distortion of the frequency response of the filter. This distortion is filter topology dependent, since topology sensitivity to coefficient quantisation varies. Direct form topologies are highly sensitive to coefficient quantisation compared to coupled forms, state variable, ladder and lattice structures. Typically in direct form topologies filters tuned to low frequencies with respect to the sampling frequency produce large response distortions, examples are given in Wise (1998).

Arithmetic quantisation and overflow both result from finite wordlength arithmetic operations. The multiplication and accumulation of state variables can lead to an increase in wordlength. If an arithmetic result is stored in memory at a wordlength less than the result, then quantisation or overflow can occur. Quantisation errors in filter topologies act as noise sources and the overall noise characteristics of a topology greatly depends on where quantisation takes place in the topology. Arithmetic overflow in fixed point

implementations leads to large limit cycle signal behaviour (periodic, highly correlated, often self-sustained oscillations). Scaling is typically employed to avoid overflow, but this compromises the signal to noise ratio of the filter.

The Direct form 1 topology (DF1) with error feedback may be used to combat quantisation noise in high quality audio applications using 24 bit fixed point arithmetic (Dattorro, 1988). The DF1 is shown to be immune from accumulator overflow using modulo wrap-round techniques and capable of producing low frequency tuned filters using double precision coefficients to minimise the direct form high coefficient sensitivity. Dattorro, 1988 suggests that DF1 is the most suitable topology for fixed point professional audio applications. In Wilson (1993), fixed point noise models for DF1 (using error feedback) and the Gold-Rader coupled form are developed. The DF1 is shown to be computationally more efficient and produce less quantisation noise than the scaled Gold-Rader structure (using scaling to avoid overflow). Zölzer (1991) develops fixed point quantisation noise models for the DF1 (with and without error feedback), and unscaled coupled form topologies (Gold-Rader, Kingsbury and Zölzer). The Zölzer topology was found to produce superior dynamic range figures for low frequency filters.

Ladder and lattice allpass filters can be used to realise audio equaliser filters (Regalia, 1988 ; Massie, 1993). The ladder uses L_2 norm scaling at each accumulator node to combat overflow, Massie (1993). However L_2 norm scaling does not eliminate the possibility of overflow, since the scaling is based on average power not peak amplitude. Cabot (1992) details a hybrid digital filter structure using the zero part of the direct form topology and implements the pole transfer function through the use of the state-space topology or 'normal form' as described in Mullis and Roberts (1976). This topology is reported to produce quantisation noise and stability characteristics similar to the state-space topology, but produces a more efficient coefficient calculation method than that of the state-space topology.

Theoretical noise models for cascaded and parallel DF1 structures, implemented in floating point arithmetic are developed in Liu and Kaneko (1969). This model has been used in comparison to a DF1 fixed point noise model (Weinstein and Oppenheim, 1969). It concludes that 'floating point arithmetic leads to a lower noise to signal ratio than fixed point if the mantissa is equal in length to the fixed point word'. Despite the development of fixed and floating point noise models, no work has studied and compared the noise and distortions of actual filter implementations in fixed and floating point arithmetic or assessed the influences of sampling frequency.

1.3.3 Filter coefficient update and resulting disturbances

Discrete time-varying filters are commonly used to allow user control over filter types and parameters in digital equalisers. However, time-varying digital filters are susceptible to audible transient distortion (disturbances), which can be problematic in audio systems. Disturbances due to a filter state change during the configuration of an audio system installation can damage speakers and distress the human ear. The production of noticeable disturbance as a result of filter state change could hinder a live performance.

Much work has been done in the investigation and optimisation of filter systems to accommodate state change with minimal audible disturbance. Mourjopoulos et al (1990) investigated optimal filter parameter interpolation rates with the goal of minimising audible disturbances. Bell and shelving filter functions were implemented using an allpass topology embedded in a gain structure. Frequency parameter changes under sinusoidal input were found to produce worst case disturbances. Hanna (1994) examined optimal parameter update rates for a bell filter, using the DF1 topology. Music and sinusoids were used as audio test programme. Parameter update intervals of 1ms were found to produce inaudible disturbances in virtually all cases.

Zölzer et al (1993) examined optimal switching strategies for parameter interpolation for shelving, bell, low and high pass filters. Strategies for parameter morphing between two filter types relies on an intermediate unity gain (flat) frequency response then changing filter type and morphing onto the target response. The work reported minimal acoustic disturbance for parameter update intervals in the region of 1 to 40 ms. The work also describes a technique using an attenuator to reduce signal level during filter state change. This produces audible amplitude modulation during the filter state change.

Rosenthal (1997) describes a technique to reduce disturbance using two filters implemented in parallel. One filter produces the current frequency response whilst the other filter implements the new target response. Once the target filter has settled, its output level is ramped up and the other filter's output level is ramped down. This incurs the implementation costs of two filters and produces unsuitable amplitude modulation for slow settling filters (low frequency, high Q filters). In Välimäki (1995) a transient eliminator system is described. This system implements a secondary pole transfer function in parallel with the primary filter. The secondary 'pole only' filter is fed the coefficients at an earlier point in time than the primary filter. Once the 'pole only' filter has settled its state variables are transferred to the pole section of the primary filter, minimising the coefficient update disturbance. This technique is computational exhaustive and therefore not suitable for efficient implementation.

In Ding and Rossum (1995) coefficient interpolation techniques are developed which closely simulate the logarithmic nature of frequency and gain filter parameters. This technique dispenses with the need for a complex parameter to coefficient mapping. However, the technique only applies to particular filter types and is bound to one filter topology. Furthermore, the interpolated/extrapolated states do not produce identical responses to a known ideal filter function. Therefore, the use of a target coefficient set

may be needed to realise exact filter settings. The technique is desirable in the application of musical synthesis.

In Moorer (1999) audio disturbances for DF1, the allpass ladder structure and a state-variable topology are measured using white noise input excitation. The filter response change, used in the measurements, was a linear sweep of the centre frequency of a bell filter (from 44.1 Hz to 441 Hz, constant peak gain, 18 dB, and constant bandwidth 22.05 Hz). The DF1 was shown to produce disturbance magnitudes of over 10. The allpass ladder and state-variable structures produced disturbance magnitudes of over 1.5.

Most of the previous work used parameter interpolation. Techniques exist that reduce disturbances by direct interpolation of coefficient sets, from an initial stable set of coefficients to a target set over a pre-determined period. Despite this work in the optimisation of coefficient update in filters there is no work describing the actual disturbance mechanisms in the various filter topologies and the effects of signal sampling frequency on disturbance.

1.4 Statement of problem

Frequency response distortions are introduced by the s to z-plane mapping in on-line coefficient calculation schemes. No work exists that analytically compares frequency response distortions and the computational load introduced by existing s to z plane mapping techniques for the various filter types required (low and high pass, shelving and bell filters). No research has studied the effects of higher sampling frequencies on response distortion. Furthermore there is a need to investigate novel mappings with minimal response distortion and computational load.

The benefits of important topologies are diminished by the need for scaling to prevent overflow in fixed point implementations. These important topologies can be implemented without scaling in floating point arithmetic. There is a need to investigate and compare

the behaviour of these topologies in floating point implementation. Some topologies are immune to accumulator overflow in fixed point implementations through the use of twos-complement modulo wrap-round. There is a need to compare the behaviour of these filter topologies in fixed and floating point implementations. Work to date has relied on theoretical quantisation noise models in the examination of filter topology behaviour. There is a need to investigate the behaviour of filter topologies under finite wordlength constraints, using actual signals as input stimuli. This may highlight further distortions and non-linear behaviour in discrete-time filters implemented in fixed and floating point arithmetic. No work to date, known to the author, has assessed the finite wordlength effects of the various topologies operating at higher sampling rates.

There appears to be no research describing the disturbance mechanisms and resulting signal behaviour in the popular equaliser filter topologies under coefficient update. Mourjopoulos et al (1990) and Hanna (1994) suggest that the type of input excitation has a large effect on the resulting disturbance. However no work to date has examined the disturbance effects under various input excitations. No work to date has studied the disturbance effects of the various different filter response changes possible in a digital equaliser. Furthermore, no comparison has been made of the coefficient interpolation schemes devised to reduce coefficient update disturbance. No research, known to the author, has investigated the actual behaviour of the coefficient interpolation schemes under finite wordlength arithmetic or the effects of higher signal sampling rates on coefficient update disturbance. The above research issues need to be addressed for the successful development of efficient high quality audio equalisers.

1.5 Project aim and objectives

The primary aim of the project is to investigate distortions associated with the efficient implementation of discrete-time equalisers. Specific objectives of the project are to:

- Investigate and develop novel techniques to minimise distortions during coefficient calculation paying particular attention to computational efficiency. The expected outcome is a knowledge base of the performance of various s to z-plane mapping techniques which are suitable for coefficient calculation for real time varying filters incurring minimal static frequency response distortion.

- Assess the impact of factors such as wordlength, type of arithmetic and input excitation on the performance (or noise behaviour) of digital equalisers. This will involve developing an environment for emulating discrete time domain filters with variable wordlength and arithmetic. The environment will be developed in Mathcad, a visual and graphical mathematical design software package, (Mathsoft, 1998). Various filter topologies will be implemented and noise analysis under various input excitations will be performed. The effects of higher sampling rates will be examined. This will provide an understanding of the behaviour of filter topologies under fixed and floating point arithmetic with varying wordlengths.

- Investigate signal disturbance behaviour for various filter topologies under coefficient update. The effects of various input stimuli and filter settings on signal disturbance will be examined. This is expected to produce an understanding of the relationship between input excitation, frequency response, filter topology and resulting signal disturbance. The performance of the various interpolation techniques in the minimisation of signal disturbances will be examined as well as the sensitivity of the various interpolators to finite wordlength arithmetic and sampling rate.

1.6 Outline of thesis

The basic theory of real-time digital audio equaliser systems commonly associated with digital audio mixing systems is described in Chapter 2. Various audio filter types, existing

coefficient calculation techniques and update methods are described. The various filter topologies studied in this work are described in detail.

The work on digital equaliser coefficient calculation techniques is described in Chapter 3. This work examines existing s to z-plane mappings and develops novel mapping techniques with the aim of minimal response distortion and computational load. The effects of higher sampling frequencies on response distortion are also examined. Optimal mapping techniques are suggested for each the sampling frequencies considered.

Chapter 4 describes the development of finite wordlength arithmetic functions for use in the filter topology emulation environment. These functions emulate the finite wordlength behaviour of fixed and floating point arithmetic, for any given fractional wordlength and binary coding scheme. Arithmetic tests were performed, finding the emulated arithmetic identical (bit exact) to a specific DSP platform (Appendix A).

Chapter 5 describes an investigation into filter topology behaviour under finite wordlength arithmetic. Filter topologies are implemented in the discrete-time domain, in fixed and floating point arithmetic, of varying wordlengths. Distortions generated by various different input stimuli are presented. The noise behaviour for filter topologies operating at higher sampling frequencies is also investigated.

Chapter 6 describes the investigation of topology behaviour under coefficient update. Various input stimuli and frequency response changes are used to examine the disturbance mechanisms in discrete-time filter topology implementations. The effectiveness of parameter and coefficient interpolation schemes is also assessed. Furthermore disturbance effects at higher sampling frequencies are presented. Chapter 7 reviews the work done, suggesting future work and presents the conclusions of the project.

2 Basic theory of real-time digital audio equaliser systems

2.1 Introduction

Parametric equalisation is normally found on every audio channel in a mixing system. The equaliser is a group of user controllable cascaded filters. The mixing system operator (user) manipulates the filter types and parameters, in real-time, to produce a desired frequency response. This chapter provides a background into real-time user controllable digital audio equalisers. There are three main aspects in digital equaliser systems - filter specification and coefficient realisation, topology implementation under finite wordlength arithmetic and the management of coefficient update for real time control. These three aspects are the main topics of this chapter.

2.2 Audio Filter Equaliser types

Filter types commonly found in equaliser systems are bell (peaking), low frequency (LF) shelving, high frequency (HF) shelving, notch, low and high pass second order functions. This section discusses their basic magnitude frequency response attributes, the associated filter parameter controls and their application in audio equalisation. All of these filter functions are primarily considered to be magnitude frequency equalisers with a non-linear minimum phase response.

2.2.1 Bell filter

Figure 2-1 shows typical bell filter magnitude frequency response curves. The three essential user controllable parameters are the peak gain tuned frequency (Fc); the peak gain or attenuation (G) and the bandwidth/Q factor (Q). The tuned frequency, Fc, determines the maximum peak or minimum trough in the magnitude frequency response and has an operable range between 20 Hz to 20 kHz. The amount of peak or trough (boost or cut) is user controlled by a gain control parameter, G. Typical gain settings are ±18 dB. If G is positive then a gain is applied producing a boost at the tuned frequency. If G is negative then attenuation is applied, producing a cut at the tuned frequency. On either side of the tuned frequency the response slope tends towards unity gain. The selectivity is controlled by the Q factor (Q) or bandwidth control (typical Q settings range from 0.1 to 10). There are many equaliser design variations leading to different relationships between Q and gain G, (Tromans, 1995; Bohn, 1986). Many analogue equaliser designs have a non-constant Q relationship with variation in gain, G. This was historically for cost reasons, however this does produce low Q (wide bandwidth) for small values of G. This is often said to be useful for subtle changes in gain. Constant Q equalisers use a Q factor that is independent of gain, G; and provides accurate control of gain and bandwidth. Furthermore, there are two important variants of constant Q equalisers, asymmetric and symmetric Q. These variants differ in response for the cut case (attenuation at tuned frequency). The asymmetric variant produces a constant bandwidth at the –3 dB gain points. The symmetric variant defines bandwidth at 3 dB less than the tuned frequency gain. This produces a symmetrical magnitude frequency response with the boost case and can be used to correct earlier equalisation, by simply negating the gain setting. In this thesis the constant Q boost variant, with a symmetrical cut response, shown in Figure 2-1,

is used. Further details about the relationships between Q and bandwidth can be found in the literature (Moorer, 1983; White, 1985; Bristow-Johnson , 1994; Tromans, 1995).

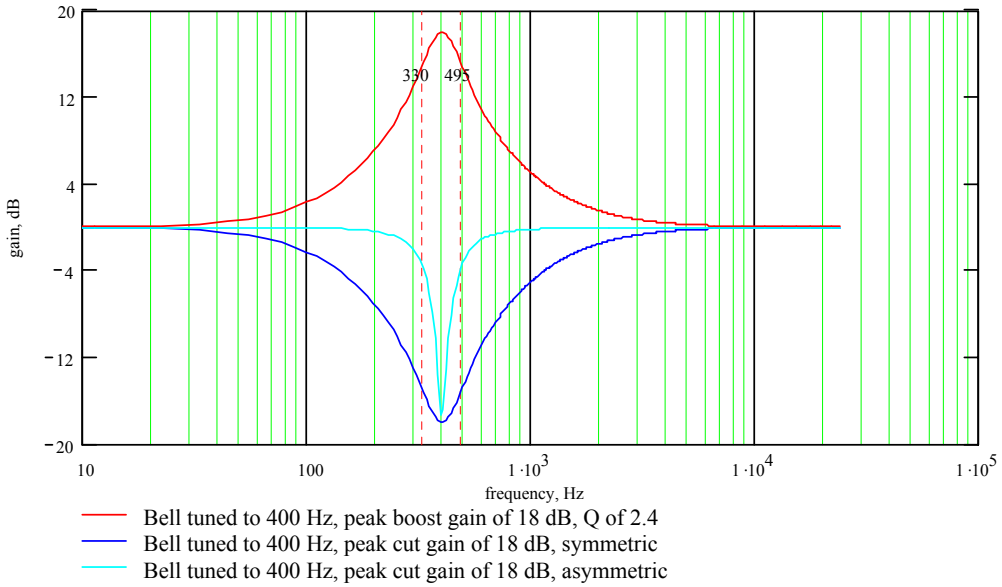

Figure 2-1 Magnitude frequency response of typical bell filters

2.2.2 Shelving filters

Low frequency (LF) and high frequency (HF) shelving filters are used to boost or cut LF and HF frequency regions respectively. Figure 2-2 shows typical magnitude frequency response curves for the LF and HF shelving filter functions. The LF shelf provides user controlled boost or cut gain from dc (zero Hertz) up to the corner (tuned) frequency, which may be defined as the frequency at which the gain is one dB less than the maximum gain. As frequency increases the magnitude response slope tends towards unity gain. The HF shelf provides unity gain at dc and user controllable gain (cut or boost) at high frequencies. Both LF and HF shelving filters have corner frequency controls providing the user with the ability to control the frequencies that are affected by the gain control. The slope in the magnitude response between the maximum gain and unity is sometimes user controllable and typically between three to nine dB per octave. Historically, first

order shelving functions were used as treble and bass controls, providing a gentle slope, typically less than four dB per octave. For the purposes of this work only second order shelving functions are considered.

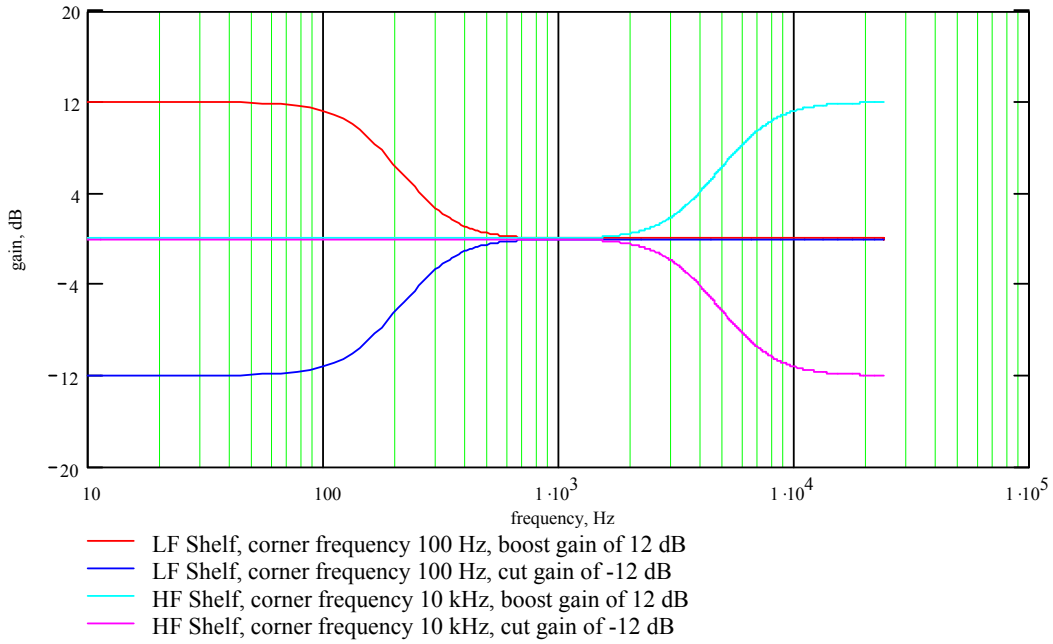

Figure 2-2 Magnitude frequency response of LF and HF shelving filters

2.2.3 Notch filter

The notch filter is a rare function in a mixing system, since the bell cut filter with a high Q setting can provide almost an identical response. However, digital equalisers are increasingly being used in many novel equalisation applications. Notch filters are useful in acoustic feedback, resonance elimination. The notch filter has a controlled centre frequency and produces a theoretical zero transmission at this tuned frequency. The bandwidth of a notch filter is specified by the two –3 dB gain points. Bandwidths are typically narrow, e.g 5 to 20 Hz. Alternatively bandwidths can be dependent on the filters tuned frequency - 1/3rd to 1/12th octave are typical. The magnitude response of a 20 Hz tuned notch filter, with a –3 dB bandwidth of 7.5 Hz is shown in Figure 2-3.

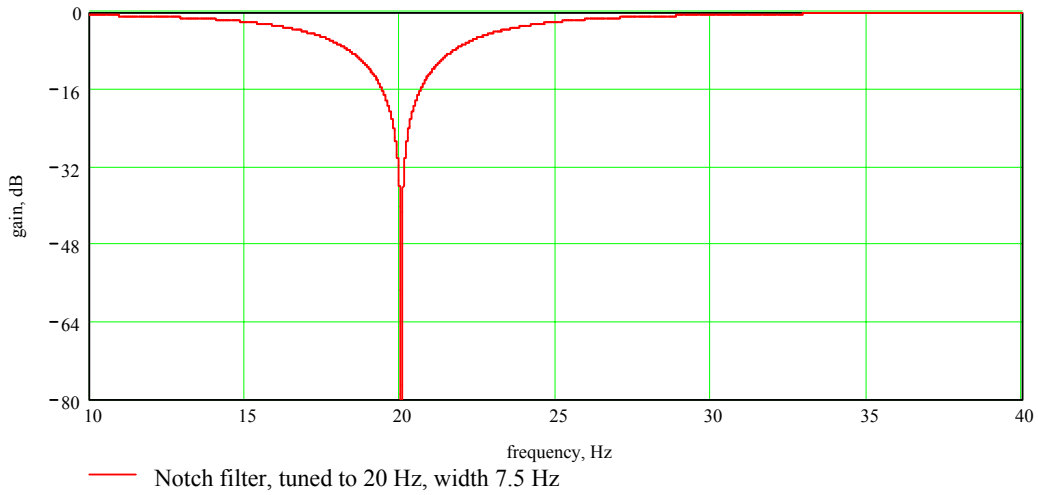

Figure 2-3 Magnitude frequency response of a notch filter

2.2.4 Low and High pass filters

Audio equalisers typically include low and high pass filters to eliminate noise from the audio channel. These are fixed slope (Q), unity gain filters with one tuned frequency control, defined at the –3dB edge frequency. In addition to this, digital audio mixing systems include low and high pass filter functions for basic speaker cross-over applications. Cross-over filters are usually high order functions, ranging from 6 to 48 dB per octave with various slope and phase responses (Butterworth, Bessel and Linkwitz-Riley). The decomposition of such high order slope responses leads to a cascade of second order filters, where the manipulation of each of the filter's Q factor can produce any of the desired responses. Since these decomposed Q factors are nominally in a region between 0.58 to 0.8, a Butterworth response (0.7071) is used for the purposes of this work. Figure 2-4 shows low and high pass filter functions, tuned to 400 Hz with a Butterworth slope response.

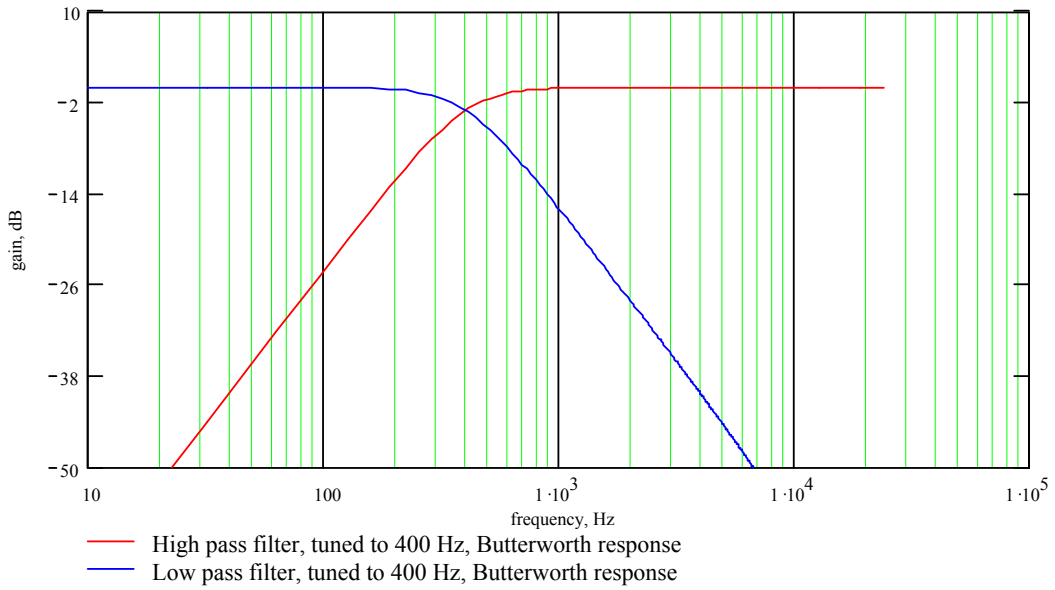

Figure 2-4 Magnitude frequency response of LF and HF pass filters

2.3 S-plane to z-plane transforms

Generalised s-plane expressions for the bell, shelving and pass filter functions are given in Equations (2-1) to (2-5), where Ω_c equals $2\pi Fc$, and Fc represents the tuned frequency of the filter. For convenience, the variables A and B are functions of Gain, G and Q factor (bandwidth or slope). Expressing these parameters as generalised variables, A and B, provides a comprehensive solution for coefficient calculation, whereby A and B can be substituted for a particular Gain-Q relationship, as discussed in Section 2.2.

$$H_{bell}(s) = \frac{s^2 + A\,\Omega_c\, s + {\Omega_c}^2}{s^2 + B\,\Omega_c\, s + {\Omega_c}^2} \tag{2-1}$$

$$H_{lfshelf}(s) = \frac{s^2 + 2A\,\Omega_c\, s + A^2\,{\Omega_c}^2}{s^2 + 2B\,\Omega_c\, s + B^2\,{\Omega_c}^2} \tag{2-2}$$

$$H_{hfshelf}(s) = \frac{A^2 s^2 + 2A\Omega_c s + \Omega_c^2}{B^2 s^2 + 2B\Omega_c s + \Omega_c^2} \tag{2-3}$$

$$H_{lowpass}(s) = \frac{\Omega_c^2}{s^2 + \frac{\Omega_c}{Q} s + \Omega_c^2} \tag{2-4}$$

$$H_{highpass}(s) = \frac{s^2}{s^2 + \frac{\Omega_c}{Q} s + \Omega_c^2} \tag{2-5}$$

The standard z-transform, matched z-transform and bilinear z-transform are three techniques used to convert s-plane transfer functions to z-plane transfer functions. Standard z-transforms (SZT) are time invariant and preserve the filter's impulse or step response (often referred to as impulse and step invariant). Whilst the SZT preserves the filter's time response, the resulting frequency response suffers from large errors that cannot be corrected (Clark et al, 1996).

'Direct pole and zero placement' on the z-plane is an alternative and simple efficient method of realising some filter functions such as notch filters, Equation (2-6), where Ts denotes the sampling interval. However, the bell, shelving and pass filters all require a transformation technique to realise a transfer function in the z domain.

$$H_{notch}(z) = \frac{1 - 2\cos(\Omega_c T_s) z^{-1} + z^{-2}}{1 - 2r\cos(\Omega_c T_s) z^{-1} + r^2 z^{-2}} \tag{2-6}$$

2.3.1 Bilinear z-transform (BZT)

The basis of the BZT is a 'tanh' function. Using standard exponential identities and mapping from the s-plane to the z-plane by making the substitution $z = e^{S.Ts}$ leads to the

BZT 's to z' mapping, given in Equation (2-7). The BZT band-limits the frequency response of the filter to within the Nyquist frequency and so avoids the problem of fold-over aliasing. However, the band-limiting effect of the BZT distorts the frequency response of the filter - the so-called warping effect. Various techniques exist that preserve (pre-warp) specific response attributes or parameters to counteract the warping effects of the BZT. Equation (2-8) shows a standard technique for compensating the warping effect at a given frequency, Fc. The pre-warped frequency, $F_{prewarped}$, then replaces Fc in the z-plane transfer function.

$$s = \frac{2}{Ts} \cdot \frac{z-1}{z+1} \tag{2-7}$$

$$F_{prewarped} = \tan\left[\pi \frac{Fc}{Fs}\right] \frac{Fs}{\pi} \tag{2-8}$$

2.3.2 Matched z-transform (MZT)

In the MZT method, real and complex s-plane poles (or zeros) are mapped onto the z-plane using substitutions given in Equations (2-9) and (2-10) respectively. The MZT maps s-plane poles and zeros directly onto the z-plane with no band limiting at the Nyquist frequency. The resulting discrete-time filter suffers from two forms of response distortion. An overall gain shift (dc offset) and an image response distortion towards the Nyquist frequency. The MZT has been used to generate shelving and bell functions (M^{c}Nally, 1979; Hirata, 1980), however the resulting response distortions were not explored.

$$s + a \rightarrow 1 - z^{-1}\, e^{-aTs} \tag{2-9}$$

$$(s + a - jb)\cdot(s + a + jb) \rightarrow 1 - 2\,z^{-1}\, e^{-a\,Ts}\, \cos(b\,Ts) + e^{-2\,a\,Ts}\, z^{-2} \tag{2-10}$$

2.4 Implementation of discrete-time filters

The s to z-plane mapping techniques discussed in Section 2.3 produce z-plane transfer functions of the general form given in Equation (2-11). This can be implemented through the discrete-time domain difference equation, (2-12). A direct implementation of this difference equation produces the Direct Form 1 (DF1) topology, shown in Figure 2-5. The DF1 topology is a 'zero before pole' topology whereby its zero transfer function operates directly on the current input sample, x_i , the previous two input sample instances, x_{i-1} and x_{i-2} . The zero transfer function feeds the single accumulator. The accumulator feeds the output, y_i , and the pole transfer function. The pole transfer function recursive paths, that is the previous output sample (y_{i-1}) and second from previous output samples (y_{i-2}), feedback into the accumulator.

$$H_{notch}(z) = \frac{a0 + a1\cdot z^{-1} + a2\cdot z^{-2}}{1 + b1\cdot z^{-1} + b2\cdot z^{-2}} \tag{2-11}$$

$$y_i = a0\cdot x_i + a1\cdot x_{i-1} + a2\cdot x_{i-2} - b1\cdot y_{i-1} - b2\cdot y_{i-2} \tag{2-12}$$

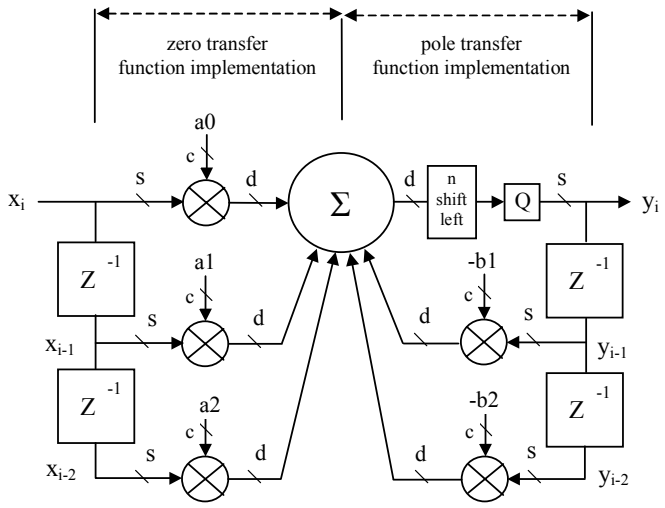

Figure 2-5 Direct Form 1 (DF1) second order filter topology, 's' and 'd' denote single and double precision wordlengths, 'c' denotes the wordlength of the coefficients.

2.4.1 Fixed point data formats

Fixed point, 24 bit processing is widely used in audio processing applications. Furthermore in (Analog Devices, 1997) a DSP platform capable of 32 bit fixed and floating point arithmetic is described. Both 24 and 32 bit fixed point wordlengths are considered in this work, Figure 2-6. Fixed point arithmetic typically uses twos-complement and not sign magnitude binary representation, due to multiplier efficiency. Fixed point DSP's typically use fractional representation for the inputs to the multiplier (fractional normalised inputs). For example, a 24 bit twos-complement fractional format uses 23 fractional bits and one sign bit (1.23). The numerical range of twos-complement 1.23 format is $(1\text{-}2^{-23})$ to –1. The fixed point multiplication product requires twice the number of fractional bits, in addition to the sign bit. For example, multiplying two 1.23 numbers produces a number format of 1.46. If the result is represented with less fractional bits a quantisation error is introduced. Fixed point accumulation does not produce any increase in fractional wordlength. Therefore two 1.23 numbers can be summed to produce a fractional result of 1.23 without any fractional quantisation. However, fixed point

accumulation can produce integer wordlength growth and as a result most accumulator implementations provide some headroom integer bits (guard bits).

Figure 2-5 shows the data path wordlengths for the DF1 assuming single precision fixed point implementation. Single precision implementation means that the state variables are stored at the same precision as the multiplier input and data bus wordlengths of the processor. Single precision wordlengths are denoted by 's' in the DF1 topology, Figure 2-5. The product accumulation is performed at double precision wordlength, denoted by 'd'. This ensures that no fractional quantisation is necessary in the summation of the products. Thus one quantisation point exists at the accumulator output where the data is quantised from double to single precision for state variable storage. This quantisation noise source is greatly amplified by filters with high gain in the pole transfer function. Consequently in critical filtering applications the use of first and second order error feedback to reduce quantisation noise in 24 bit implementations has been required (Wilson, 1993). Alternatively, modern fixed point processors provide efficient instructions to facilitate double precision multiplication (Motorola, 1999). This can eliminate the quantisation noise source at the accumulator output and eliminate high pole gain noise problems.

Fixed point filter topologies rely on coefficient scaling techniques to ensure all coefficients are within the numerical range of the arithmetic (typically fractional). DF1 coefficient scaling is performed by dividing the coefficient set by a global scaling factor, 2^n. This scaling factor ensures all coefficients are within the numerical range. The coefficient scaling is compensated by multiplying the accumulator output by the scaling factor (n arithmetic shifts left). The scaled coefficients and arithmetic shifting operate on double precision data and therefore do not incur any additional quantisation noise in a single precision implementation.

A major disadvantage of fixed point arithmetic is accumulator overflow. Overflow can be alleviated through scaling, with the penalty of a reduction in signal to noise ratio. There are various scaling strategies that may be used, for example L_1, L_2 and L_∞ norms. An excellent review of these is given in (Massie, 1993). In principle the most conservative scaling strategy, L_∞ norm, should be used for professional audio equipment, since overflow must be prevented. However, the actual scaling scheme is filter response dependent and to obtain optimal scaling at all times scaling factors must be calculated for every filter setting. Some topologies can utilise twos-complement modulo wraparound schemes (the Jackson rule), (Jackson et al, 1968; Dattorro, 1988) to prevent intermediate accumulator overflow. This produces infinite headroom for intermediate accumulation results if the filter is stable and does not produce a final accumulator result outside the first modulo. Topologies that rely on scaling for fixed point implementation are typically avoided in professional audio equaliser applications.

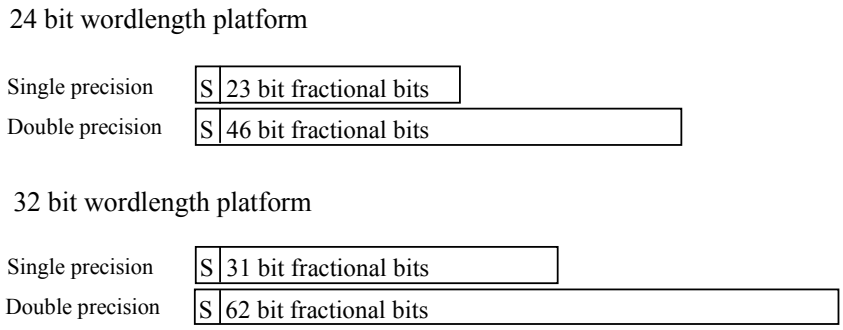

Figure 2-6 Fixed point wordlength formats

2.4.2 Floating point data formats

There are two floating point implementations typically found in signal processing platforms (Analog Devices, 1997; Texas Instruments, 1992). The first floating point platform (Texas Instruments, 1992) uses twos-complement binary coding. This scheme uses a twos-complement mantissa and exponent, providing an exponent range of $2^{\pm 127}$.

The second floating point scheme has been standardised by (ANSI/IEEE754, 1985) and (ANSI/IEEE854, 1987), here referred to in this thesis as IEEE floating point. This uses sign magnitude binary coding. The exponent for the IEEE format uses an unsigned eight bit field, biased by 127, providing a range of $2^{\pm 127}$. Despite existing definitions for double precision floating point, it is rare for existing DSP platforms to support double precision arithmetic. The two floating point platforms considered in this work provide an extended precision format, shown in Figure 2-7. Extended precision formats provide higher fractional resolution in the mantissa, without an increase in exponent range. It is also common that the product registers and addition/subtraction are implemented in extended precision. Since extended precision does not provide twice the fractional bits in the mantissa as the single precision, it is possible to introduce quantisation in the product and addition registers. Therefore extended precision implementation (storing state variables in extended precision) does not eliminate quantisation.

The pole transfer function of an audio filter can produce gains in the region of 120 dB, Figure 5-2. The operating range of an eight bit exponent (1500 dB) facilitates the implementation of any topology without arithmetic or coefficient scaling to avoid overflow. However, floating point arithmetic still uses a finite wordlength mantissa and relies on normalised data for arithmetic operations. Therefore quantisation noise still exists within filter structures.

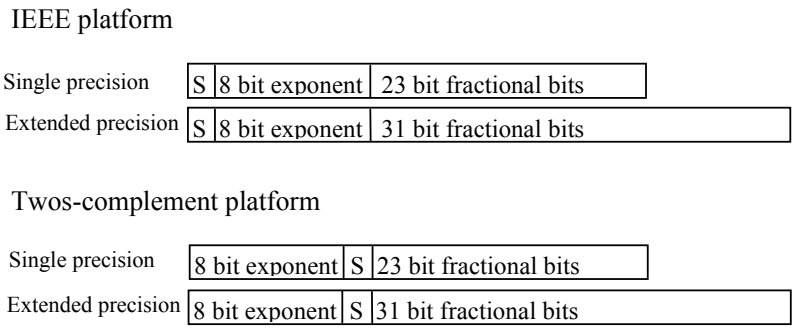

Figure 2-7 Floating point formats

2.4.3 Coefficient sensitivity

Coefficients are typically quantised to the multiplicand wordlength, denoted 'c', in Figure 2-5. This produces static quantisation errors in the coefficient set distorting the pole zero placements and the filter's frequency response. Coefficient sensitivity to quantisation is topology dependent and is widely researched (Dattorro, 1988; Wilson, 1993; Wise, 1998). Direct Form topologies produce the worst coefficient quantisation effects on frequency response. Direct Form topologies produce poor frequency responses for high Q filters tuned to low frequencies. At a sampling frequency of 48 kHz using 24 bit coefficients, direct form topologies cannot reproduce accurate frequency responses for high pass filter tuned to less than 10 Hz. As suggested in (Wise, 1998) alternative topologies can reproduce critical case filter responses. Furthermore the use 32 bit or 24 bit double precision (46 fraction bits) provide ample coefficient resolution for direct form audio filtering implementations, even at higher sampling rates such as 192 kHz.

2.5 Alternative filter topologies for implementing discrete-time filters

Alternative filter topologies to the DF1 are introduced in this section. Topology diagrams are given for each topology studied. Quantisation points for single precision implementation are shown on each of the topology diagrams. Two topologies that have not been studied in this work are the Agarwal-Burrus structure (Agarwal-Burrus, 1975) and the Ding-Rossum structure (Ding and Rossum, 1995). The Agarwal-Burrus structure replaces unit delays with integrators, which are reported to produce dc instability in (Moorer, 1999). The Ding-Rossum structure is reported to be suitable for particular filter types and does not suit this work's objective of a filter structure suitable for many filter types, as described in Section 2.2.

2.5.1 Direct Form 2 (Canonical Form)

Figure 2-8 shows the DF2 topology diagram. The topology is often referred to as the canonical form since it only requires two memory locations. The topology is 'pole before zero' and has one quantisation error source for single precision implementation. Consequently the error transfer function contains poles and zeros. The topology cannot utilise modulo wrap-round to avoid accumulator overflow and relies on input scaling to avoid overflow in fixed point implementations. Therefore in this work the topology is implemented under floating point arithmetic.

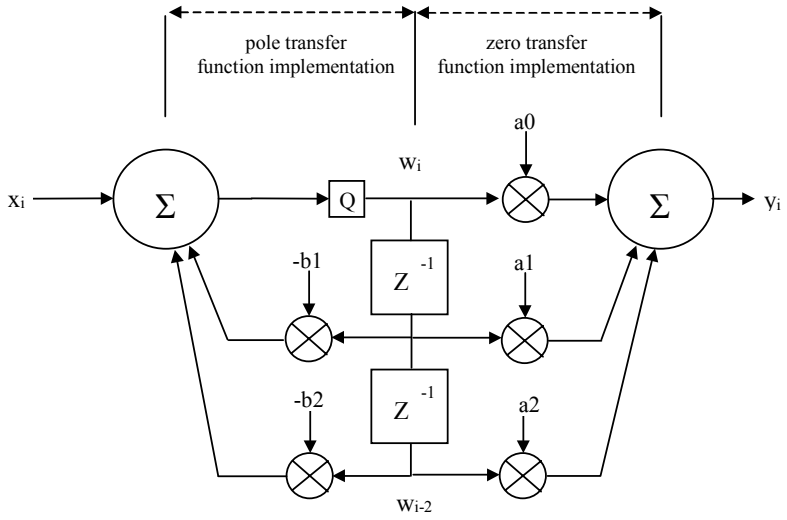

Figure 2-8 Direct Form 2 filter topology

2.5.2 Transposed Direct Form 1 (DF1T)

The transposed Direct Form 1 (DF1T) is a pole before zero topology and more similar to the DF2 than the DF1, Figure 2-9. The topology has four potential quantisation points in single precision implementation. The DF1T relies on scaling to prevent accumulator overflow in fixed point implementations. In this work the topology is implemented in floating point arithmetic.

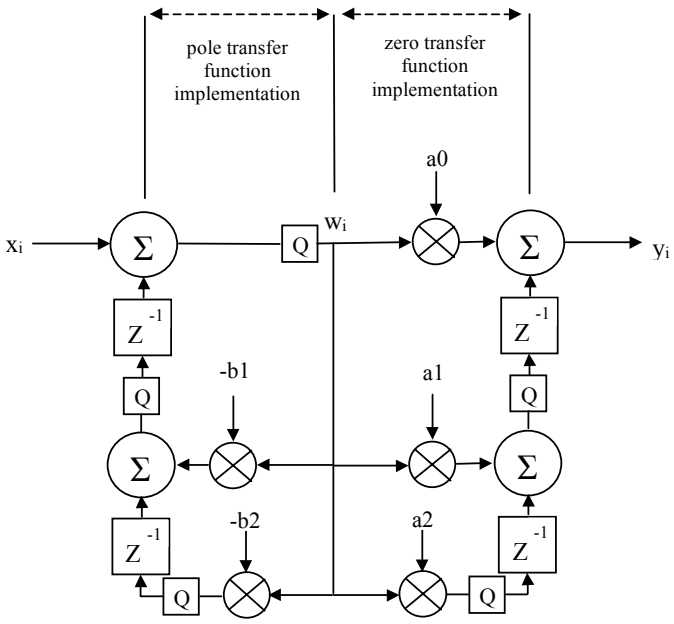

Figure 2-9 Transposed Direct Form 1 (DF1T) filter topology

2.5.3 Transposed Direct Form 2 (DF2T)

The transposed Direct Form 2 (DF2T) is a zero before pole topology and more similar to the DF1 than the DF2, Figure 2-10. The topology has three potential quantisation points in single precision implementation. The DF1T can utilise modulo wrap-round to avoid accumulator overflow in fixed point implementations. In this work the topology is implemented in fixed and floating point arithmetic.

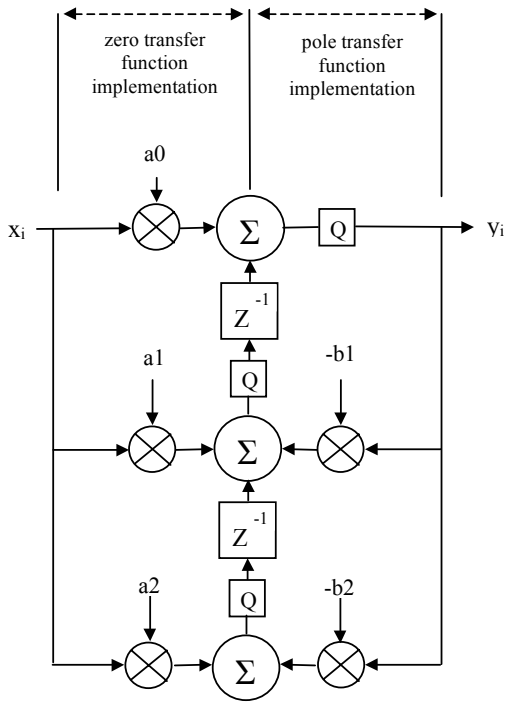

Figure 2-10 Transposed Direct Form 2 (DF2T) filter topology

2.5.4 Coupled form topologies (Gold-Rader, Zölzer, Kingsbury)

The Gold-Rader (Gold and Rader, 1967), Kingsbury (Kingsbury, 1972) and the Zölzer (Zölzer, 1991) are all 'coupled form' topologies, Figure 2-11, Figure 2-12 and Figure 2-13. Using the zero transfer function implementation of the Direct Form, coupled with different implementations of the pole transfer function produces low coefficient sensitivity properties for the pole placements. The topologies require scaling to avoid accumulator overflow in fixed point implementation. Theoretical quantisation noise models are developed in (Wilson, 1993; Zölzer, 1991). (Wilson, 1993) compares the scaled Gold-Rader to the DF1 in fixed point implementation. The low frequency performance of the Gold-Rader is found to be similar to DF1 with error shaping. (Zolzer, 1991) develops theoretical noise models for the three topologies, assuming no scaling and documents theoretical signal to noise ratios for the three topologies for different tuned frequencies assuming a Q factor of 0.7071. This work considers the unscaled topologies implemented in floating point arithmetic. Coefficient realisation is given in Appendix D.

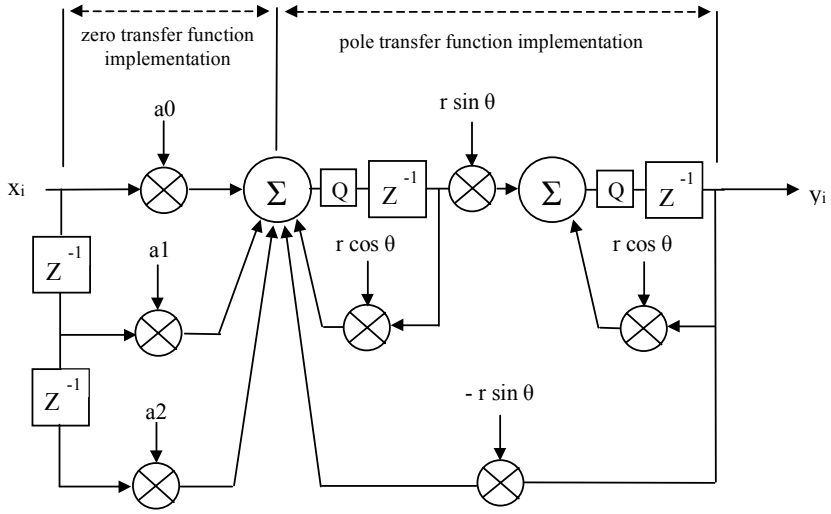

Figure 2-11 Gold Rader filter topology

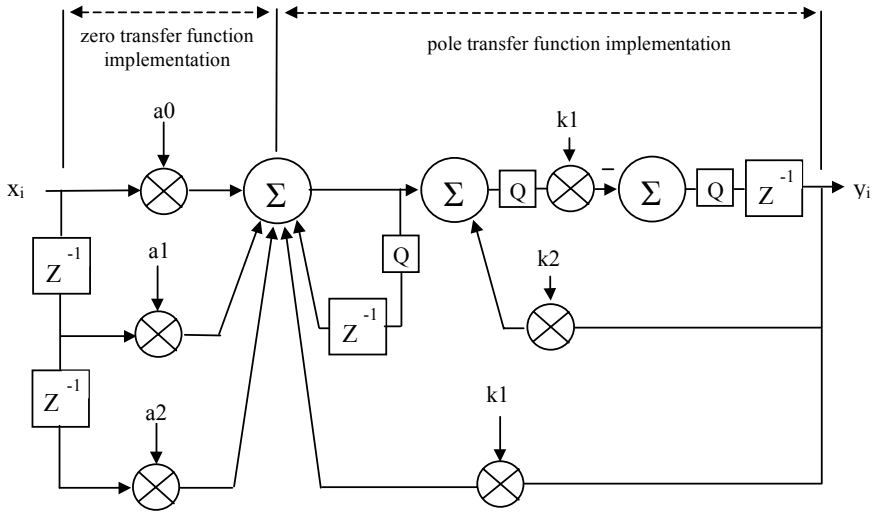

Figure 2-12 Kingsbury filter topology

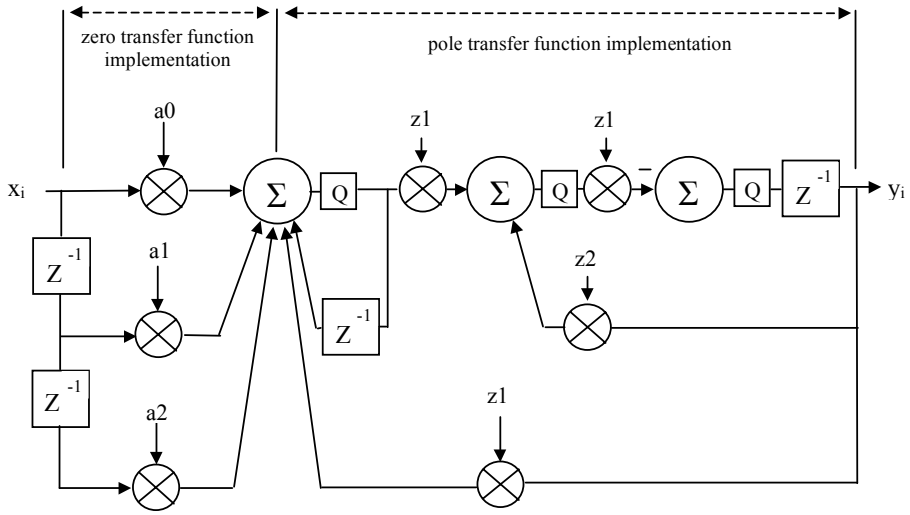

Figure 2-13 Zölzer filter topology

2.5.5 State-space topology

Figure 2-14 shows the state-space topology diagram. The topology is intrinsically scaled to the L_2 norm to reduce overflow in fixed point implementation. Mullis and Roberts (1976) shows that the topology produces excellent noise performance in fixed point implementations, since its noise characteristics are only dependent on filter bandwidth, not frequency. It is also stated that L_2 scaling produces near optimal noise characteristics within the topology. However L_2 scaling does not eliminate accumulator overflow therefore the state-space topology is implemented in floating point arithmetic in this work.

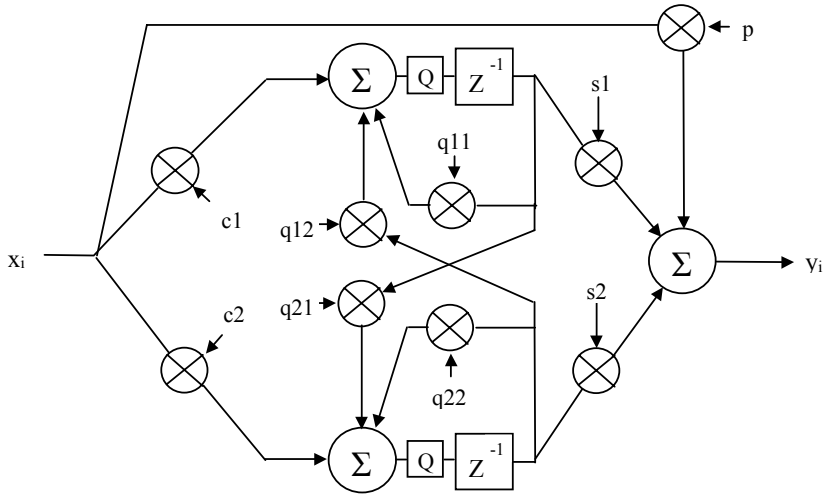

Figure 2-14 State-space filter topology

2.5.6 State-space hybrid (Cabot) topology

The state-space hybrid (Cabot) topology, Figure 2-15, uses the zero transfer function of the direct form and couples that with the pole transfer function implementation of the state-space topology, (Cabot, 1992). The hybrid topology is less complex than the state-space topology. Coefficient calculation is simplified compared to that of the state-space topology. In Cabot (1992) it is claimed that is produces truncation noise and stability characteristics like those of the state-space form. In this work this topology is implemented in fixed and floating point.

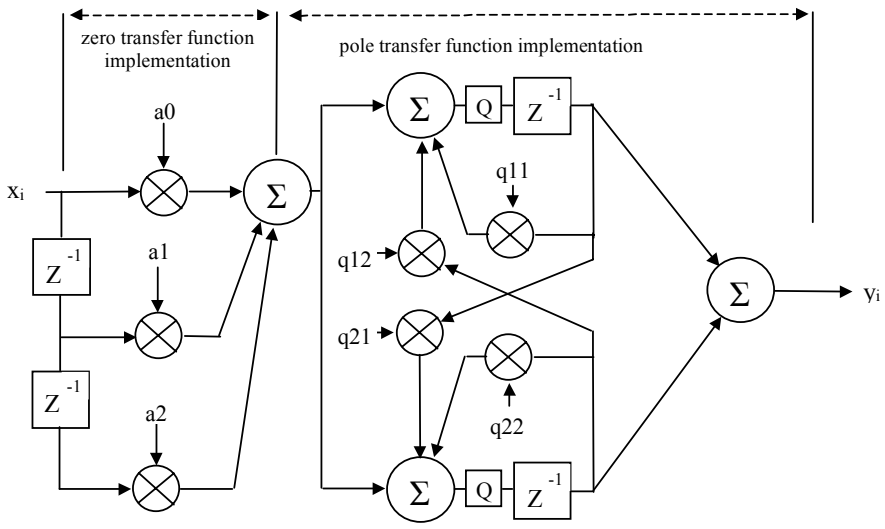

Figure 2-15 State-space hybrid (Cabot) filter topology

2.5.7 Ladder and lattice structures

All of the filter types discussed in Section 2.2 can be realised using allpass filters embedded in gain stages (Regalia et al, 1988). Allpass implementations produce efficient one to one parameter to coefficient mappings, for some filter types and parameters. However this work focuses on generic filter functions, eliminating this advantage. In Massie (1993) the implementation of allpass gain structures using lattice and ladder allpass filters for audio equalisation is described. Both the lattice and ladder allpass topologies are 'pole before zero', see Figure 2-16 and Figure 2-17. The second order structures are two nested first order functions. The ladder uses L_2 scaling at internal accumulation nodes, however L_2 scaling is not immune from overflow (Massie, 1993). The lattice structure uses no scaling and is prone to accumulation overflow under fixed point arithmetic. The ladder can also be used as a non-allpass filter by appending a zero transfer function to the allpass network as shown in Figure 2-18. This method is used for implementing audio filters in (Moorer, 1999) and will be referred to in this work as the Moorer (ladder) implementation. The use of a gain stage with the allpass ladder and

lattice topologies will be referred to as Massie implementations. All implementations will use floating point arithmetic.

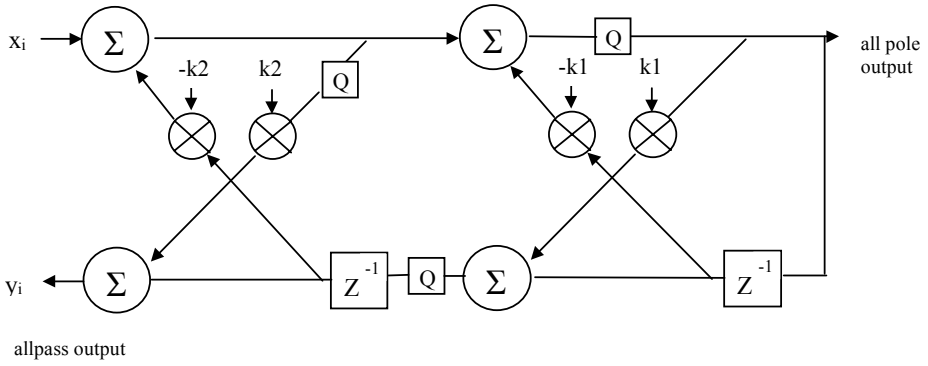

Figure 2-16 Allpass filter, lattice topology

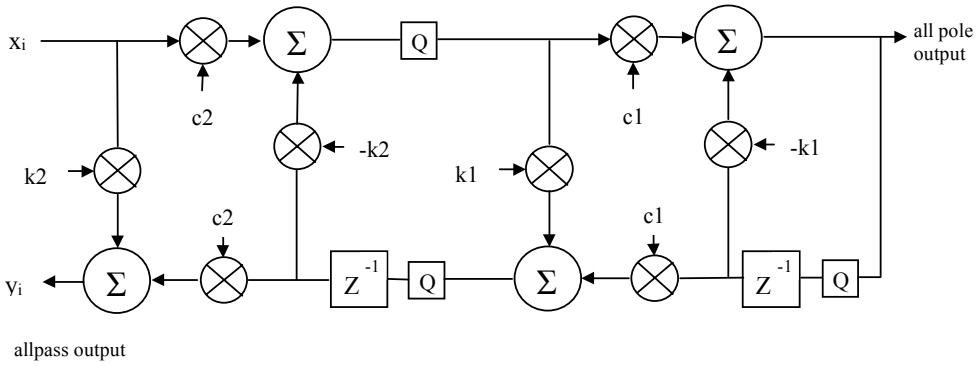

Figure 2-17 Allpass filter, ladder topology

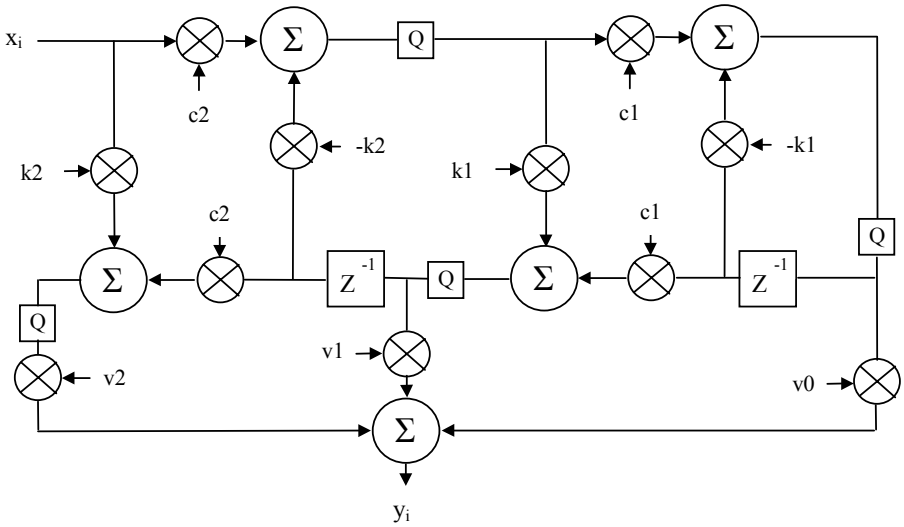

Figure 2-18 Ladder filter with appending zeros (Moorer)

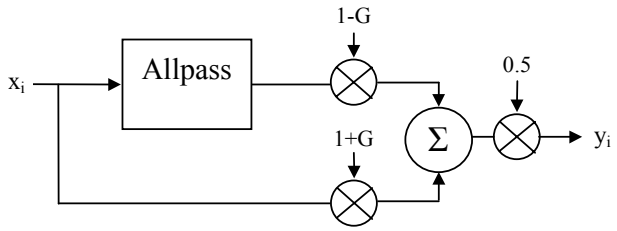

Figure 2-19 Structure for realising filter functions using an allpass filter (Regalia)

2.6 Real-time user controllable filter systems

Figure 2-20 shows a diagram of a real time user controllable digital equaliser system. There are four components in the system: the human interface (control surface), a coefficient calculation process, a parameter or coefficient interpolation section and the time-varying filter.

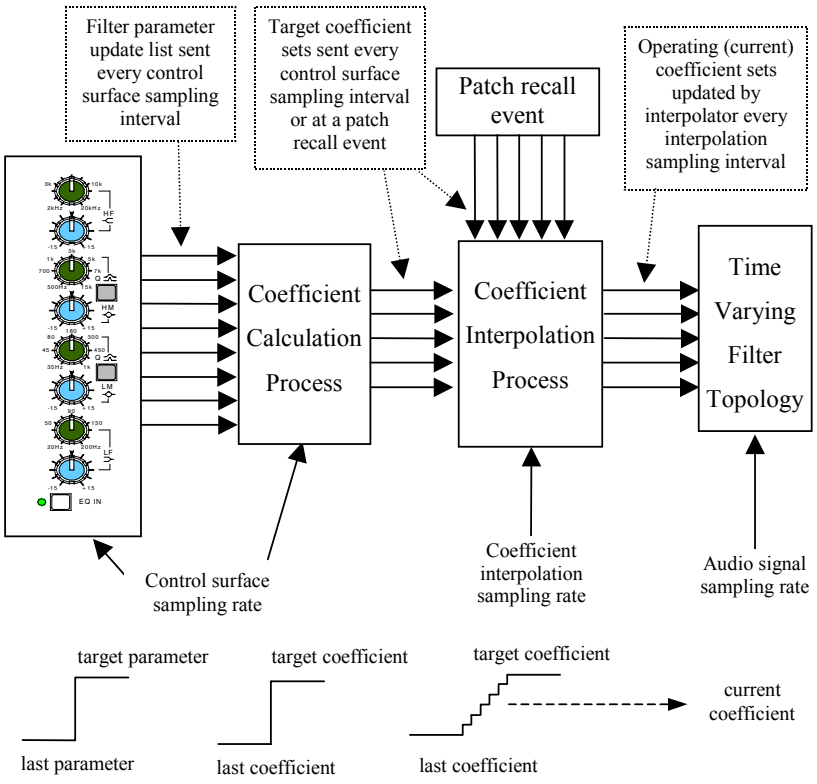

Figure 2-20 Real-time user controllable equaliser system (using coefficient interpolation)

2.6.1 Control surface

The control surface is a collection of controls which allows the user to select filter type and manipulate filter parameters in real-time. There are various ways a control can change; through user manipulation, a pre-programmed scene change (patch recall), or an automated performance change controlled by a sequencer. The equaliser system has to accommodate single and continuous filter control changes, across multiple channels of audio equalisers. The control surface sampling rate is the rate at which all associated filter parameters are sampled and subsequently updated. A control sampling rate in the region of 25 frames per second (an interval of 40ms) is widely accepted as a suitable control sampling rate, based upon visual perception of 'instantaneous' and 'smooth' control (SMPTE[1] frame rates). The audio equipment manufacturing industry tends to provide faster control sampling rates for critical audio controls such as audio level and channel mute controls. This is due to audio engineers requiring faster time responses (typically 100 frames per second) during musical reproduction arrangements. However filtering controls are not considered as critical and the more typical sampling rate of 25 frames per second is widely adopted.

2.6.2 Coefficient calculation

The second system component is the coefficient calculation section which maps filter parameters to a set of filter coefficients. On-line parameter to coefficient mapping is computationally intensive and needs to be accurate to avoid frequency response errors. Look-up tables can be used to provide a minimal computational load. However, the use of look-up tables restricts the potential number of filter types and parameter settings available to the user. Implementing look-up table schemes with extensive parameter settings and

[1] Society of Motion Picture and Television Engineers

filter types can also lead to the requirement of costly memory. For this reason, various on-line coefficient generation schemes are discussed in Chapter 3.

2.6.3 Time-varying filter topology

A time-varying filter is a filter structure with time varying coefficients. Equation (2-13) gives the discrete time domain expression for the DF1 topology, where the coefficient set {a0, a1, a2, b1, b2} varies as a function of time, sample to sample, where *i* denotes the sample instant. Any step change in the set of coefficients {a0, a1, a2, b1, b2} in time causes a filter state change. The time varying-filter topology is required to realise many different filter types with many different parameter settings. Therefore any filter structures that are limited to certain filtering functions have to be discounted.

$$y_i = a0_i \cdot x_i + a1_i \cdot x_{i-1} + a2_i \cdot x_{i-2} - b1_i \cdot y_{i-1} - b2_i \cdot y_{i-2} \tag{2-13}$$

State changes in discrete-time varying filters produce signal disturbances at the filter output. Like any discrete-time control system, the disturbance magnitude is proportional to the step change magnitude. By increasing the update (sampling) rate of the coefficients sets presented to the time varying filter the magnitude of the filter state change is reduced. Parameter and coefficient interpolators are two alternative processes that reduce filter state change disturbance in this way. These two different interpolation techniques are discussed in the following two sections.

2.6.4 Coefficient interpolation

A user controllable filtering system using coefficient interpolation is depicted in Figure 2-20. This scheme performs the parameter to coefficient mapping, then increases the coefficient update (sampling) rate. Therefore the computationally intensive parameter to

coefficient mapping is performed at the slower, control surface sampling rate. However, the parameter to coefficient mapping is non-linear and the intermediate coefficient sets produced by the interpolator do not reproduce known intermediate filter responses. There are three interpolation techniques commonly used as coefficient interpolators in audio systems; linear, exponential and sinusoidal interpolation.

Linear interpolation between a 'start' and 'target' coefficient, over N samples, can be described by Equation (2-14). The number of samples N equals the desired interpolation time, I_{period}, multiplied by the interpolation sample rate, Fs.

$$c_i \leftarrow c_{i-1} + \delta$$

where,

$$\delta \leftarrow \frac{(\text{target} - \text{start})}{N}$$

$$N \leftarrow I_{period} \cdot Fs \qquad (2\text{-}14)$$

A typical linear interpolation is shown in Figure 2-21. Each interpolation sample within an interpolation period is of a fixed step-size, δ. Updating the step size, δ, will determine exactly how the interpolator will behave in the next interpolation period. The implementation of the algorithm needs to ensure that the interpolation stops at the target and does not end up extrapolating. This can be done by forcing the increment, δ, equal to zero after N samples or by disabling the interpolation completely after this period.

A first order exponential interpolation is described by Equation (2-15), where, k, is the damping factor. The damping factor, k, is a function of the interpolator sampling rate, Fs and the time constant, τ. The time constant, τ, is the time required for the interpolator to reach two thirds of its final value. A typical exponential interpolation is shown in Figure 2-21. At the start of the interpolation the step-sizes are large, at the end of the

interpolation the step-sizes tend towards zero. By updating the target variable every interpolation period the interpolation tends towards the new defined target.

$$c_i \leftarrow c_{i-1} + k \cdot \left(target - c_{i-1} \right)$$

where,

$$k \leftarrow 1 - e^{-\frac{1}{Fs \cdot \tau}} \qquad (2\text{-}15)$$

Sinusoidal interpolation of audio filter coefficients is implemented in Rimell and Hawksford (1996). An example of sinusoidal interpolation is shown in Figure 2-21. The step size is largest in the middle of the interpolation period and tends towards zero at the start and end of the interpolation period. There are numerous methods of implementation. One technique is shown in Equation (2-16). The interpolation is controlled at the start point through variables, dv, ddv and Const, Equation (2-16). At the end of the interpolation period the variables, dv, ddv and Const are set to zero to disable the interpolation at the end target value.

$$\begin{cases} v_i \leftarrow v_{i-1} + dv_{i-1} \\ dv_i \leftarrow dv_{i-1} + ddv_{i-1} \\ ddv_i \leftarrow ddv_{i-1} + Const \end{cases}$$

where the follow variables are initialised
at the beginning of the interpolation period

$$dv_{initial} \leftarrow (target - start) \cdot \frac{(3 \cdot N - 2)}{N^3}$$

$$ddv_{initial} \leftarrow \frac{6 \cdot (target - start) \cdot (N - 2)}{N^3}$$

$$Const \leftarrow \frac{-12 \cdot (target - start)}{N^3}$$

(2-16)

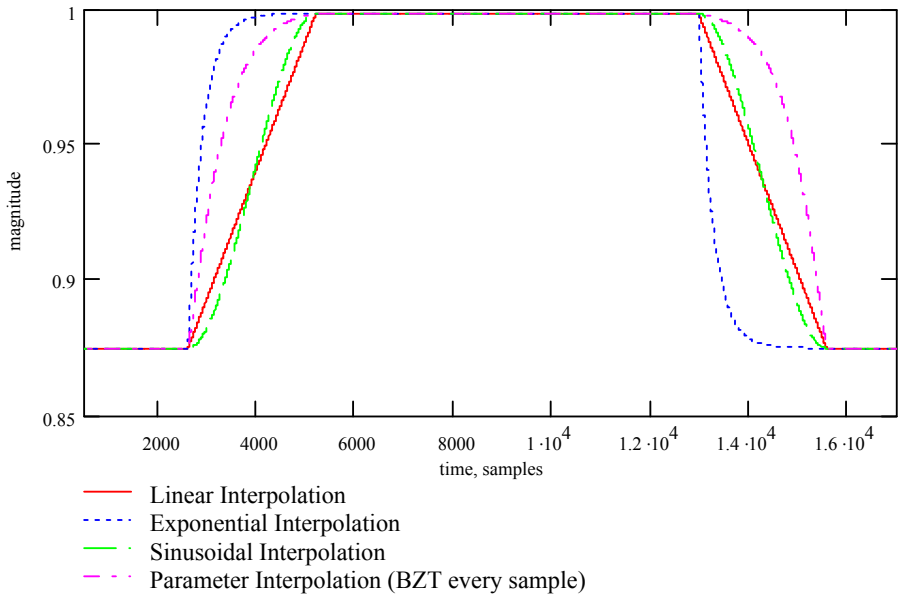

Figure 2-21 Typical coefficient interpolation functions, operating on a variable changing between 0.87 and 0.99.

The interpolation period for all of the techniques is set to equal the control surface sampling interval, typically 30 to 50 ms (Section 2.6.1). This results in a smooth data interpolation between control surface sampling intervals. The sinusoidal and linear interpolators are tuned through the constant, N (number of samples per interpolation period). Tuning the exponential interpolator is more complex. Setting the time constant, τ, to the control surface interval, I_{period}, results in the interpolation reaching two thirds of its travel in the interval. Setting, τ, to say 15 times I_{period} means the interpolation completes prematurely with large step changes. The selection of τ, within the constraints of this work, are discussed further in Chapter 6.

The final value error can be defined as the difference between the target value and the actual value produced by the interpolator at the end of the interpolation period. The error is an effect of the interpolator implementation under finite wordlength arithmetic. Final value error in coefficients can produce large frequency response errors in the target filter. 'Clamping' techniques are used to ensure that the interpolator reaches the target

coefficient at the end of the period, effectively reducing the final value error to zero. Clamping linear and sinusoidal interpolators can be performed by detecting the interpolation period has completed and then replacing the final value in the interpolator with the target coefficient. The exponential interpolator clamp is best implemented by detecting the interpolator has reached steady state (the interpolator is no longer changing the coefficient) and then replacing the current interpolator value with the target. This produces a small step change at the end of the interpolation but ensures zero steady state error.

2.6.5 Parameter interpolation

A user controllable filtering system using parameter interpolation is shown in Figure 2-22. This method performs the interpolation (increase in sampling rate) in the parameter domain, prior to the parameter to coefficient mapping. This ensures that every intermediate set of coefficients presented to the time varying filter is a proper known set of coefficients, which would produce the associated intermediate frequency response. There two disadvantages of parameter interpolation. Operating the parameter to coefficient mapping process at the higher sampling rate incurs a considerable increase in computational load. Since the parameter to coefficient mapping process is computationally intensive. The second disadvantage concerns the problem of filter type switching. Parameter interpolation schemes are easily implemented in conditions where parameter settings change between two known states. However if the filter state change involves a filter type change, there is still a potential discontinuity. Techniques of parameter management for interpolating between filter types are discussed in (Zölzer, 1993). However these techniques involve an allpass intermediate state.

2.6.6 Sub-sampled interpolators

Implementing multiple band equalisation across multiple audio channels with real time parameter control potentially involves a high number of coefficients needing to be updated per interpolator sample period. Therefore operating the interpolators at the signal sampling frequency is not computationally economical. Coefficient interpolators operating at the signal sampling rate consumes as much computational resource as the filter. Operating parameter interpolator schemes at the signal sampling frequency would be exhaustive due to the parameter to coefficient mapping. Coefficient and parameter interpolation schemes are commonly sub-sampled reducing the interpolation processing load exerted on the DSP. Investigations into the audible effects of parameter interpolation sampling rates are detailed in (Mourjopoulos et al, 1990; Hanna, 1994).

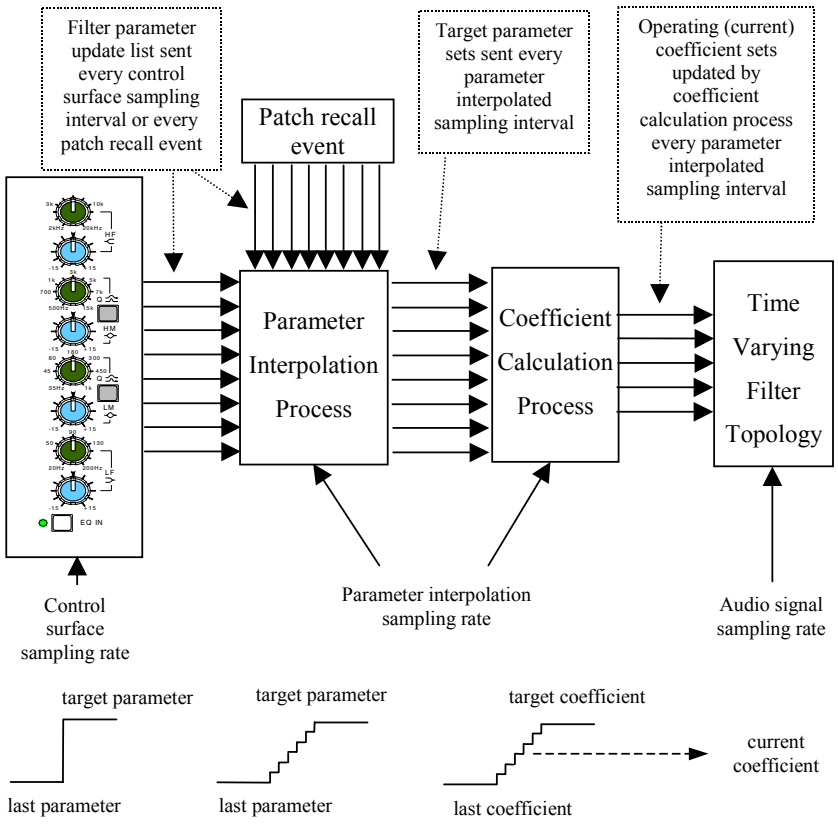

Figure 2-22 Real-time user controllable equaliser system (using parameter interpolation)

2.7 Summary

The purpose of this chapter has been to present the basic theory of real-time digital audio equalisers, which underpins the work presented in subsequent chapters. Audio equaliser filter types and associated control parameters have been specified. Existing transform methods of mapping continuous-time (s-plane) transfer functions into discrete-time (z-plane) transfer functions have been introduced. The implementation of the filter transfer function as a discrete-time difference equation under fixed and floating point arithmetic formats has been discussed. Effects of quantisation, caused by finite wordlength arithmetic are introduced with the discussion of several filter topologies that are studied in this work. Finally an overview of schemes that are used to provide real time user controllable parameters for discrete-time varying filters is given.

3 Techniques for calculating digital equaliser coefficients

3.1 Introduction

As discussed in Chapter 2, Section 2.6.2, on-line coefficient calculation in digital equaliser systems can provide greater parameter resolution and filter type selection than off-line techniques. Offline techniques incur extensive and costly memory for coefficient look-up table requirements. On-line techniques increase computational load and produce inferior discrete-time frequency responses compared to off-line coefficient generation techniques. With the increasing power of digital signal processing (DSP) technology, it is feasible and beneficial to implement both on-line coefficient calculation and the filter processing algorithms on the same processor. Thus it is necessary to optimise the computational load for coefficient calculation and minimise the frequency response distortion. Frequency response distortion is the difference between the desired frequency response (the ideal s-plane response) and the resulting discrete-time frequency response.

Techniques for on-line coefficient calculation using the bilinear z-transform (BZT) and matched z-transform (MZT) are well researched. However, the issue of frequency response distortion has not been fully addressed. In techniques based on the MZT (M^c Nally, 1979; Hirata, 1980), distortion due to aliasing errors was not considered. In BZT based techniques, a number of pre-warping schemes have been developed to minimise the inherent frequency response distortion (Moorer, 1983; White, 1985; Shpak, 1992; Bristow-Johnson, 1994; Orfanidis, 1996). However the response distortions produced by

the different schemes have not been compared. At higher sampling rates, such as 96 and 192 kHz, the nature of the response distortion will change because the Nyquist frequency is substantially higher than equaliser tuned frequencies.

In this chapter, techniques for reducing response distortions due to aliasing in the MZT methods are presented. The resulting response distortions are compared to those produced by existing BZT pre-warping schemes. These on-line coefficient calculation techniques are evaluated in terms of frequency response distortion and computational load at the sampling frequencies of 48, 96 and 192 kHz. The filter types considered are the LF and HF pass, LF and HF shelving and bell functions, as described in Chapter 2, Section 2.2 .

Notch filters are realised through direct pole and zero placements onto the z-plane. This method produces no response distortion at the centre frequency of the notch, ignoring finite wordlength coefficient sensitivity. Notch filters are typically narrow bandwidth filters and do not suffer from response distortion towards the Nyquist frequency since their magnitude response is unity gain in this region. Consequently notch filters are not considered further in this chapter.

3.2 Frequency response distortion in BZT-based equalisers

The BZT mapping of s-plane transfer functions into z-plane transfer functions is described in Chapter 2, Section 2.3.1. This section reviews the frequency response distortions resulting from the use of the BZT to realise a z-plane transfer function from the s-plane ideal. This section specifically examines discrete-time transfer function distortions assuming an operating sampling frequency of 48 kHz.

3.2.1 LF and HF shelving filters

The BZT provides a linear s-plane to z-plane mapping for LF tuned poles and zeros. Consequently, the LF shelving filter is not sensitive to the BZT warping effects since its

response is mainly concentrated in the LF region. LF shelves with typical frequency settings (tuned corner frequency, Fc is less than 1 kHz) can be implemented using no pre-warping, avoiding the tangent function. However, for higher tuned frequency settings (2 to 3 kHz) simply pre-warping Fc, Equation (2-8), does minimise response distortion further, albeit slightly. Above one tenth of the sampling frequency, the frequency response distortion increases (increased slope roll-off). The slope roll-off distortion for an extreme LF shelving filter setting (Fc equal to 2 kHz) is shown in Figure 3-1. The resulting phase response is shown in Figure 3-2.

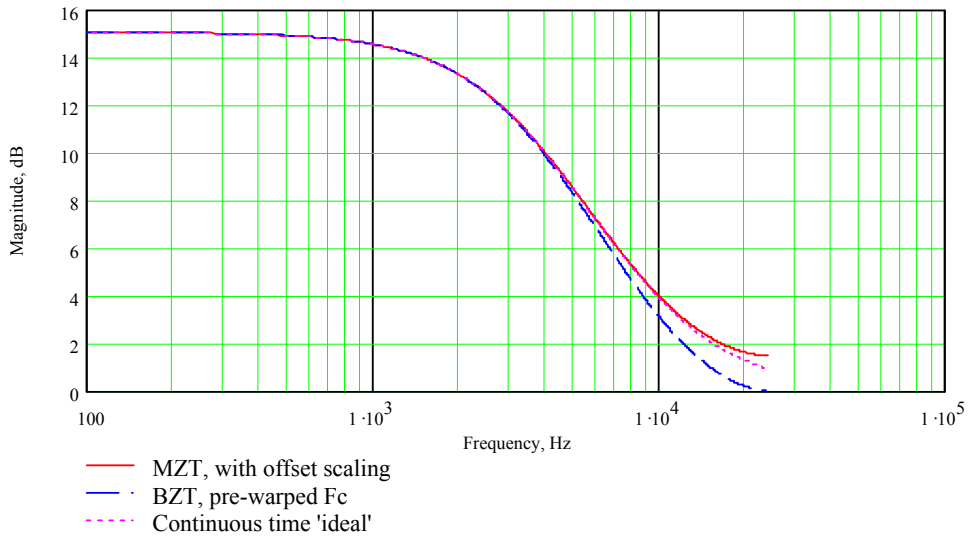

Figure 3-1 Magnitude response comparisons, LF shelving filter, Fc=2 kHz, boost gain=15 dB, Fs=48 kHz.

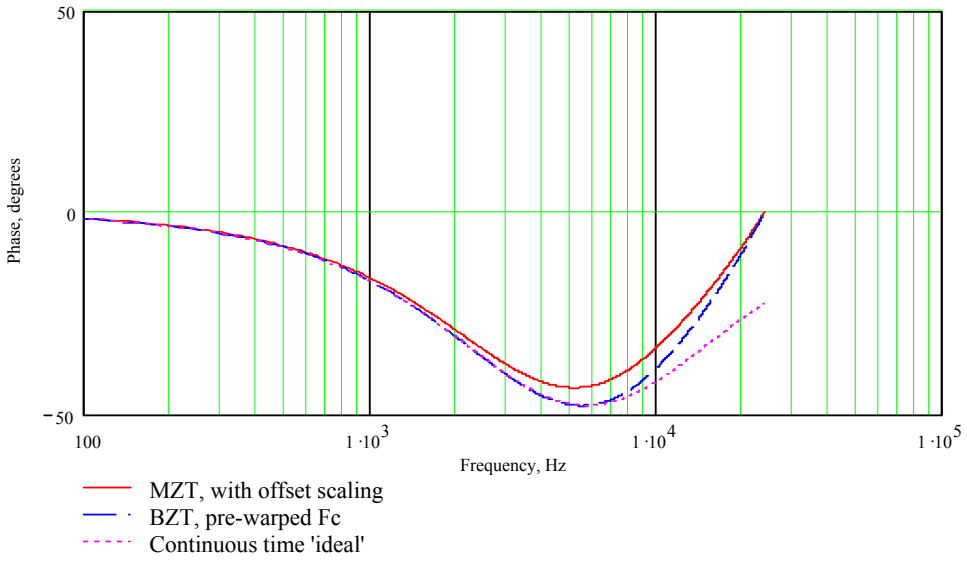

Figure 3-2 Phase response comparisons, LF shelving filter, Fc=2 kHz, boost gain=15 dB, Fs=48 kHz.

The HF shelving filter is sensitive to magnitude slope distortion since the slope is inherently in the HF region, where the BZT does not provide a linear mapping. An intuitive simple technique for reducing slope distortion is to reduce the Q factor (slope) as a function of frequency. This can be conveniently achieved by pre-warping the Q factor by the same warping factor as that of the cut-off frequency as shown in Equation (3-1). This is also computationally efficient since only one pre-warping factor needs to be calculated, as shown in Equations (2-8) and (3-1). The magnitude and phase response of the HF shelving filter, using the Fc/Q pre-warping method, is shown in Figures 3-3 and 3-4. The HF shelving filter can also be implemented without pre-warping although the slope distortion is greatly increased.

$$Q_{prewarped} = Q \cdot \frac{Fc}{\tan(\pi \cdot \frac{Fc}{Fs}) \cdot \frac{Fs}{\pi}} \qquad (3\text{-}1)$$

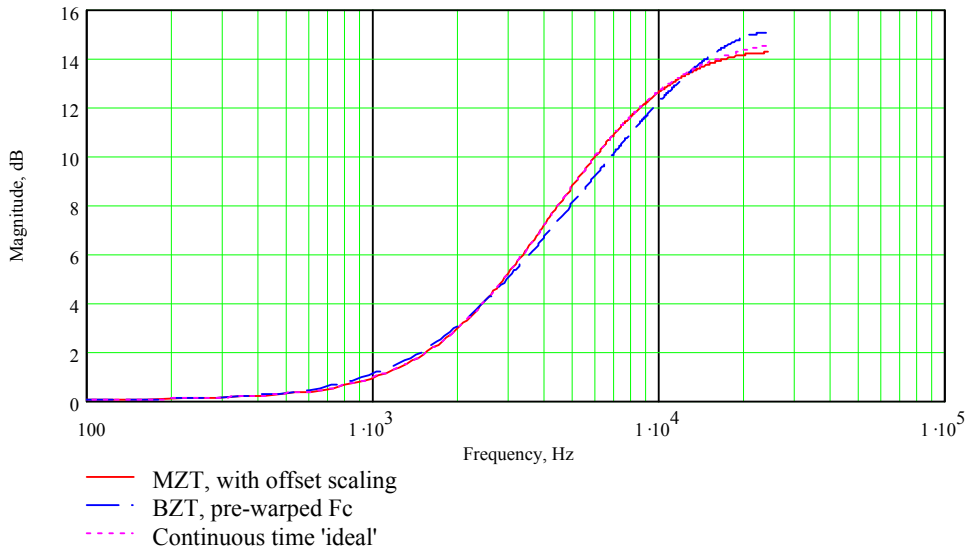

Figure 3-3 Magnitude response comparisons, HF shelving filter, Fc=12 kHz, boost gain=15 dB, Fs=48 kHz.

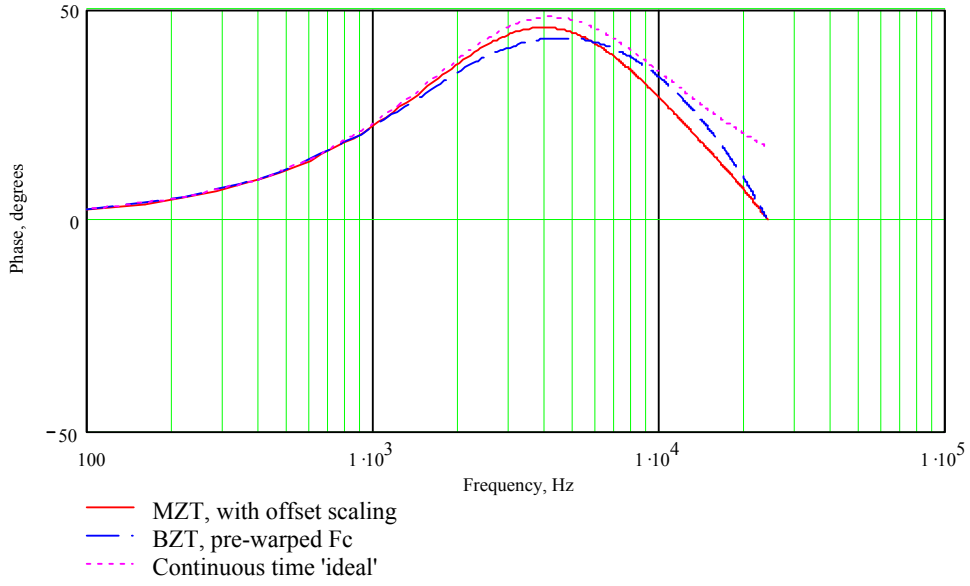

Figure 3-4 Phase response comparisons, HF shelving filter, Fc=12 kHz, boost gain=15 dB, Fs=48 kHz.

3.2.2 Low and high pass filters

Most tuned frequency settings for low and high pass filters (LPF and HPF) are small with respect to the Nyquist frequency. Consequently these filters suffer little response distortion caused by the s to z-plane mapping technique. This is due to the high frequency

response being flat at unity gain for HPF and at a low attenuation level for the LPF. However in speaker cross-over applications, these filters are both tuned to higher frequencies specifically for the high to mid frequency cross-over filter. An extreme example of a high to mid cross-over frequency would be in the region of 10 kHz. Figure 3-5 shows the response distortion for various BZT schemes realising a 10 kHz, Butterworth HPF. The BZT using no prewarping produces a 1.5 dB magnitude error at the cross-over frequency (10 kHz). The BZT (prewarping Fc) preserves the tuned frequency Fc, however the response suffers from large distortion in the slope. The BZT (prewarping Fc/Q) successfully preserves the slope, however the Fc pre-warping appears ineffective, generating a 1.5 dB error at the tuned frequency.

Figure 3-6 shows the response distortion for various BZT schemes realising a 10 kHz, Butterworth LPF. None of the schemes managed to preserve the slope in the response, due to the band-limiting nature of the BZT approaching the Nyquist frequency. The BZT using no prewarping produces a 1.5 dB magnitude error at the cross-over frequency (10 kHz). The BZT (prewarping Fc) preserves the tuned frequency Fc. The BZT (prewarping Fc/Q) does not preserve the slope or the tuned frequency, Fc. The importance of slope distortion for low and high pass filters is complex. An increase in slope (Q) is a form of response distortion, however in some applications the increase in slope can be beneficial.

A fourth order LPF and HPF Linkwitz-Riley response uses two cascaded second order Butterworth functions tuned to the same frequency (Linkwitz, 1976). One of the most important features of the Linkwitz-Riley filter is that the summed response of the LPF and HPF is flat. The BZT (using no prewarping) produces LPF and HPF that sum to a flat response, despite the large error at the tuned frequency. The BZT (prewarping Fc) method also produces LPF and HPF that provide a flat summed response. This is because the BZT warping effects for the LPF and HPF response are symmetrical across the frequency axis. However the BZT (prewarping Fc/Q) produces a non symmetrical frequency

response distortion between the LPF and HPF filters. This results in the summed response not being flat, Figure 3-7, the summed response is -2.7 dB for a tuned frequency of 10 kHz.

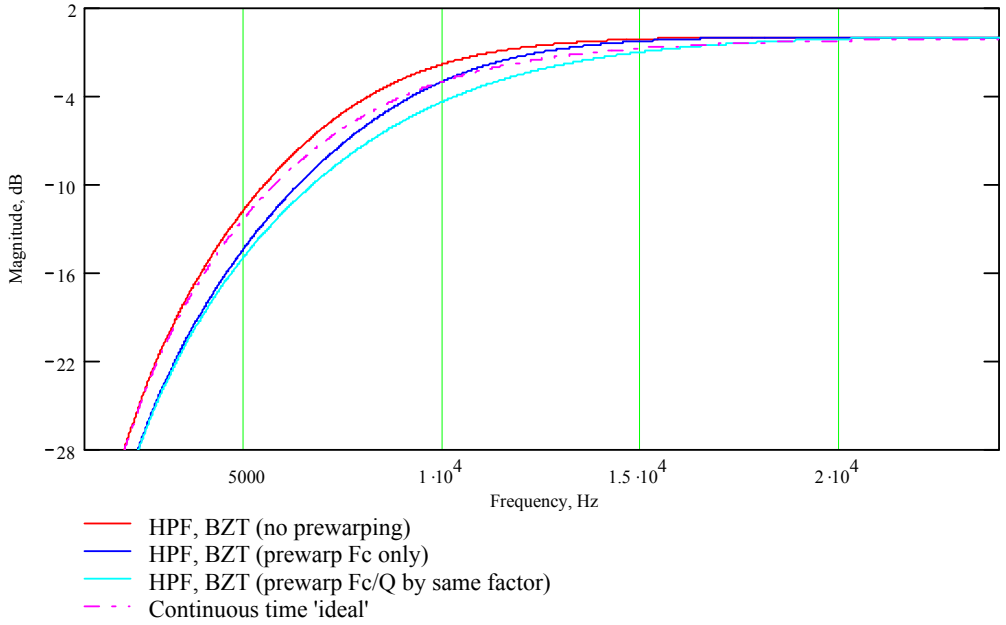

Figure 3-5 Magnitude response comparisons, high pass filter, Fc=10 kHz, Butterworth response, Fs=48 kHz.

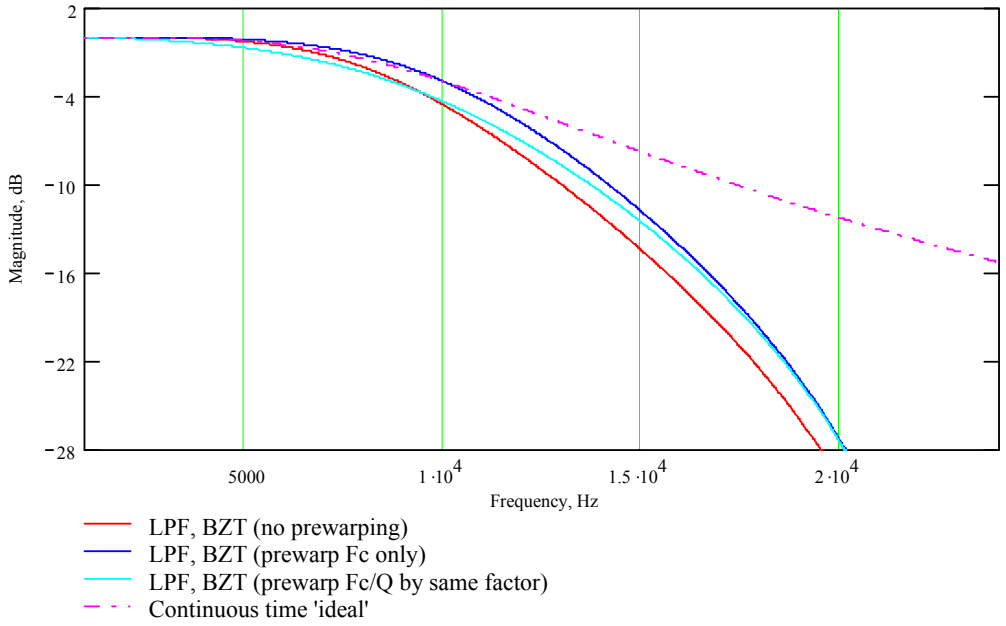

Figure 3-6 Magnitude response comparisons, low pass filter, Fc=10 kHz, Butterworth response, Fs=48 kHz.

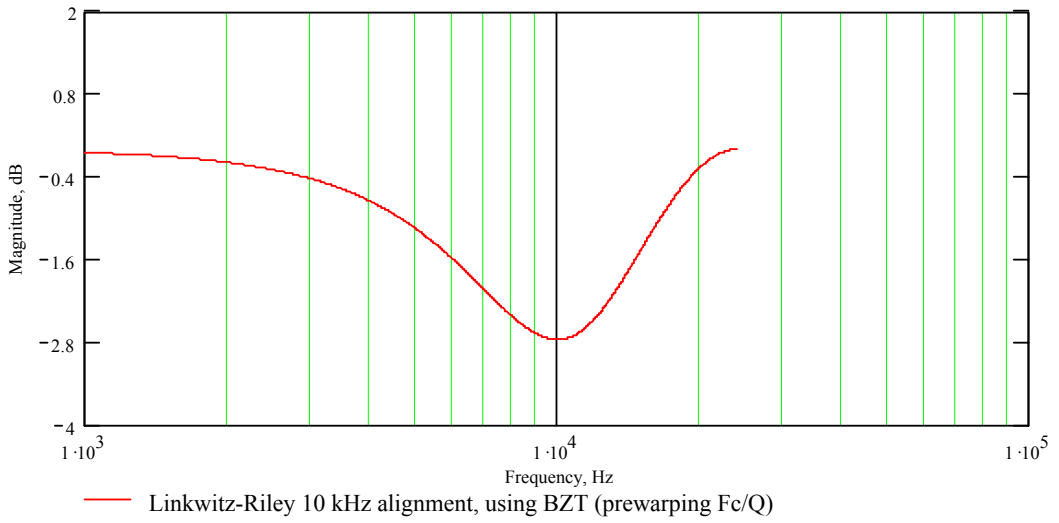

Figure 3-7 Summed magnitude response low and high pass Linkwitz-Riley alignment. BZT based filters using prewarping for Fc and Q. Fc=10 kHz, Fs=48 kHz.

3.2.3 Bell filters

A variety of techniques for minimising warping effects for the bell filter are available (Moorer, 1983; White, 1985; Shpak, 1992; Bristow-Johnson, 1994; Orfanidis, 1996). Pre-warping schemes that preserve Fc and either the –3dB lower, upper band-edge frequency or the bandwidth (BW) are developed by Shpak (1992). In (Moorer, 1983; Bristow-Johnson, 1994) BZT techniques that preserve Fc and half-gain BW are described. In Orfanidis (1996) a scheme that pre-warps Fc, BW and preserves the gain at the Nyquist frequency is developed. Figure 3-8 shows an interesting comparison of the Fc/half-gain BW pre-warping (Moorer, 1983; Bristow-Johnson, 1994) and the Fc/-3dB BW/gain at Nyquist frequency preservation, described in (Orfanidis, 1996). Note that both bandwidth preservation techniques compromise the rising slope magnitude considerably, whilst attempting to preserve the bandwidth. The Fc/half-gain BW pre-warping (Moorer, 1983; Bristow-Johnson, 1994) heavily compromises the rising slope characteristics to preserve the wider half-gain BW. The Fc/BW/Nyquist gain preservation technique (Orfanidis, 1996) causes less distortion on the rising slope, since only the –3dB bandwidth is being

preserved. Furthermore the falling slope is less distorted owing to the Nyquist gain preservation. Figure 3-9 compares the Fc and lower band-edge pre-warping scheme (Shpak, 1992) with the Fc/Q pre-warping scheme (Clark et al, 1996). The magnitude responses are similar.

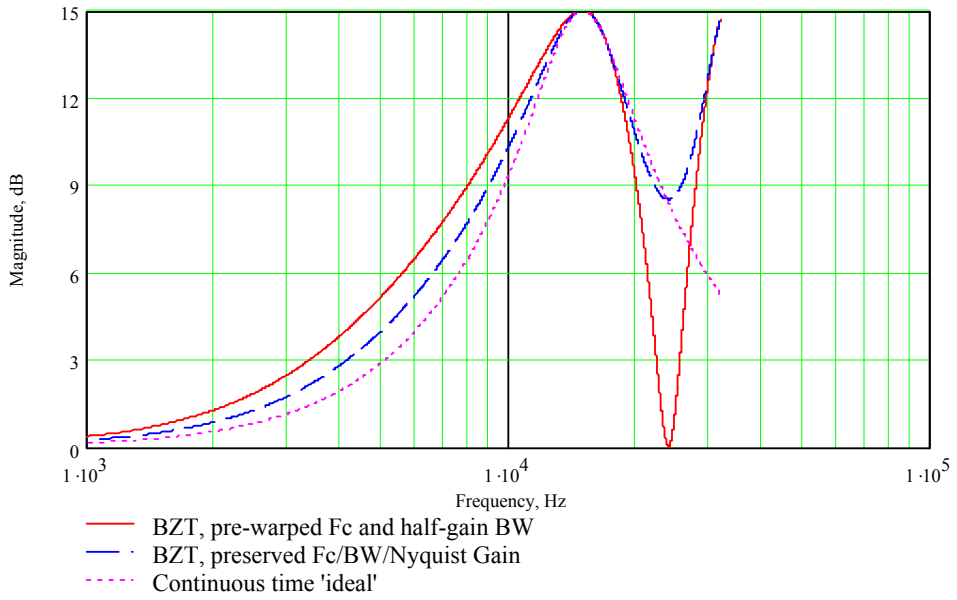

Figure 3-8 Magnitude response comparisons, bell filter, Fc=15 kHz, Q=2, boost gain=15 dB, Fs=48 kHz.

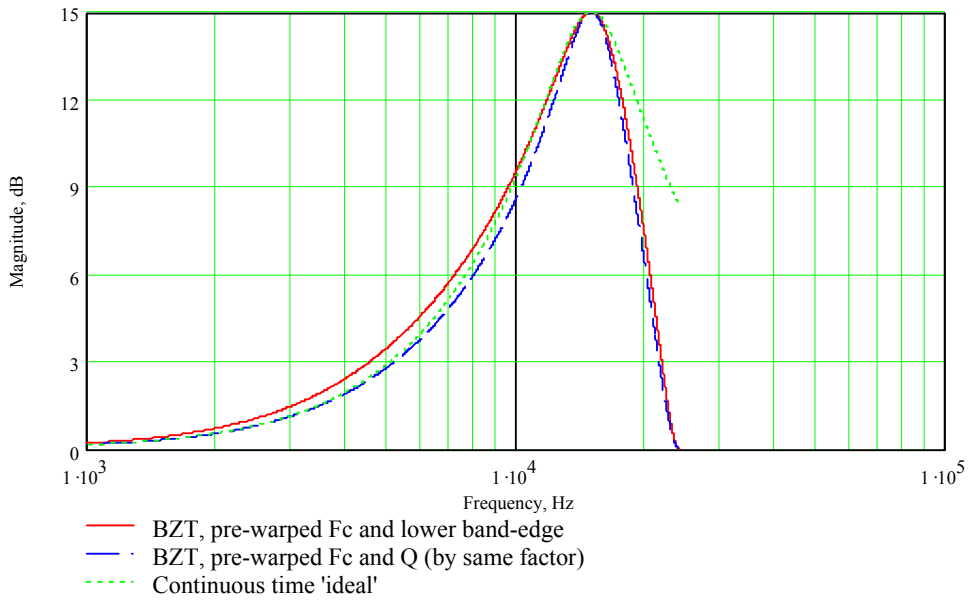

Figure 3-9 Magnitude response comparisons, bell filter, Fc=15 kHz, Q=2, boost gain=15 dB, Fs=48 kHz.

3.3 Frequency response distortion in MZT based filters

High pass and low pass filter functions are typically difficult to design using the MZT (Temes and Mitra, 1973), (Rabiner and Gold, 1975). Guard filters introducing additional zeros are typically required to produce a response that is similar to the s-plane prototype response. Therefore the MZT for low and high pass filter design is not considered in this work.

Direct application of the MZT typically produces gain-shifts (offsets) in the magnitude frequency response. In this section simple offset scaling methods are developed to correct the offsets in the magnitude frequency responses for the LF shelf, HF shelf and bell filters. Figure 3-10 shows a typical offset in the magnitude frequency response caused by the MZT in the LF shelving filter. This offset in the response can be simply corrected since the required gain at zero Hertz (dc) is known for all of the filter types. The bell and HF shelving filters both require unity gain at dc. In this case the offset can be corrected by the application of a scaling factor to the zeros (numerator) of the z-plane filter, to normalise the dc response to unity gain. The LF shelving filter requires a further stage of scaling since the gain at dc is user defined. Therefore a further gain factor needs then to be applied, in addition to the normalisation scale factor.

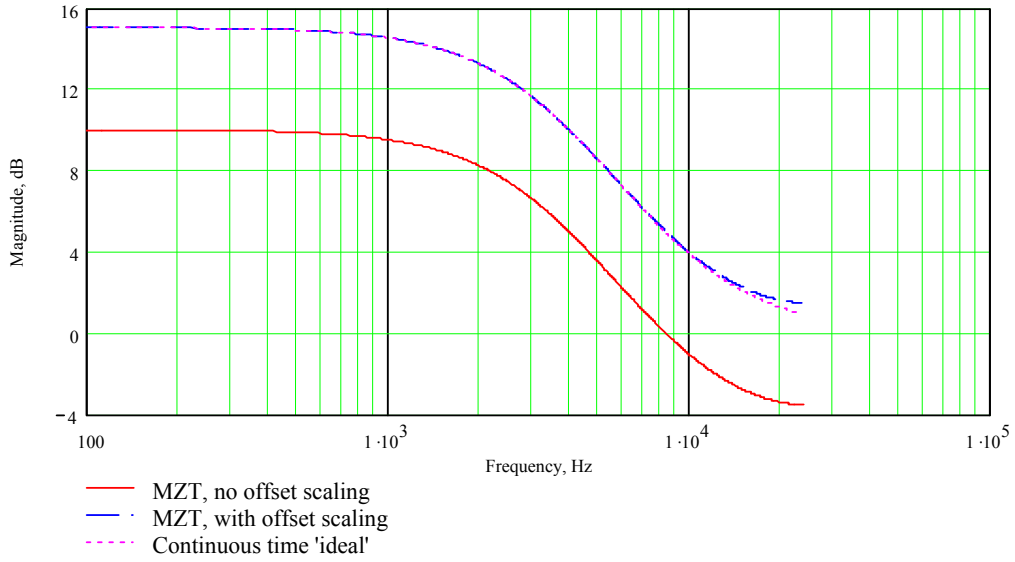

Figure 3-10 Magnitude response comparisons, LF shelving filter, Fc=2 kHz, boost gain=15 dB, Fs=48 kHz.

3.3.1 LF shelving filter

Critically damped, second order, LF Shelves, Equation (2-2) can be factored into perfect squares, to give real roots. Using Equation (2-9) we obtain the equivalent z-plane transfer function, given in Equation (3-2). Steeper slope, under-damped, shelving functions produce complex conjugate roots and require the complex mapping, Equation (2-10).

$$H_{lfshelf}(z) = \frac{1 - 2 \cdot e^{-A \cdot \Omega_c \cdot Ts} \cdot z^{-1} + (e^{-A \cdot \Omega_c \cdot Ts})^2 \cdot z^{-2}}{1 - 2 \cdot e^{-B \cdot \Omega_c \cdot Ts} \cdot z^{-1} + (e^{-B \cdot \Omega_c \cdot Ts})^2 \cdot z^{-2}} \tag{3-2}$$

The z-plane transfer function, given in Equation (3-2) is not scaled for offset correction. The normalisation scaling factor can be calculated by substituting z = 1 into Equation (3-2). Furthermore, the additional scaling required to achieve the user defined dc gain is simply $(A/B)^2$. This results in the total scaling factor given in Equation (3-3). The offset corrected magnitude response is shown in Figure 3-1.

$$Scale_{lfshelf} = \frac{(\frac{A}{B})^2}{\frac{1-2\cdot e^{-A\cdot\Omega_c\cdot Ts}+(e^{-A\cdot\Omega_c\cdot Ts})^2}{1-2\cdot e^{-B\cdot\Omega_c\cdot Ts}+(e^{-B\cdot\Omega_c\cdot Ts})^2}} \tag{3-3}$$

3.3.2 HF shelving filter

The critically damped HF shelving filter can use the real roots mapping, given in Equation (2-9). Variable Q (slope) shelving functions need to exploit the complex mapping, Equation (2-10). Applying the real roots mapping of Equation (2-9) to the factored form, Equation (3-4), gives the discrete-time transfer function given in Equation (3-5). The HF shelving filter offset scaling (dc normalisation), Equation (3-6), is shown as a numerator scaling factor of the discrete-time transfer function, Equation (3-5). The offset corrected magnitude and resulting phase response are shown in Figure 3-3 and Figure 3-4.

$$H_{hfshelf}(s) = \frac{(s+\frac{\Omega_c}{A})^2}{(s+\frac{\Omega_c}{B})^2}\cdot\frac{A^2}{B^2} \tag{3-4}$$

$$H_{hfshelf}(z) = \frac{\left[1-2\cdot e^{-\frac{\Omega_c}{A}\cdot Ts}\cdot z^{-1}+(e^{-\frac{\Omega_c}{A}\cdot Ts})^2\cdot z^{-2}\right]\cdot Scale_{hfshelf}}{1-2\cdot e^{-\frac{\Omega_c}{B}\cdot Ts}\cdot z^{-1}+(e^{-\frac{\Omega_c}{B}\cdot Ts})^2\cdot z^{-2}} \tag{3-5}$$

$$Scale_{hfshelf} = \frac{1-2\cdot e^{-\frac{\Omega_c}{B}\cdot Ts}+(e^{-\frac{\Omega_c}{B}\cdot Ts})^2}{1-2\cdot e^{-\frac{\Omega_c}{A}\cdot Ts}+(e^{-\frac{\Omega_c}{A}\cdot Ts})^2} \tag{3-6}$$

3.3.3 Bell filter

The s-plane transfer function for the bell filter, Equation (2-1), factorises to Equation (3-7). Applying the complex mapping given in Equation (2-10) leads to Equation (3-8). The

scaling factor, $Scale_{bell}$, Equation (3-9), is used to normalise the offset in the response, as a numerator scaling factor in Equation (3-8). Figure 3-11 and Figure 3-12 show the magnitude and phase response of the resulting offset corrected bell filter.

$$H_{bell}(s) = \frac{\left(s + \frac{A \cdot \Omega_c}{2}\right)^2 + [\Omega_c{}^2 - \left(\frac{A \cdot \Omega_c}{2}\right)^2]}{\left(s + \frac{B \cdot \Omega_c}{2}\right)^2 + [\Omega_c{}^2 - \left(\frac{B \cdot \Omega_c}{2}\right)^2]} \tag{3-7}$$

$$H_{bell}(z) = \frac{\left[1 - 2 \cdot e^{-\frac{A \cdot \Omega_c}{2} \cdot Ts} \cdot \cos\left[\sqrt{\Omega_c{}^2 - \left(\frac{A \cdot \Omega_c}{2}\right)^2} \cdot Ts\right] \cdot z^{-1} + \left(e^{-\frac{A \cdot \Omega_c}{2} \cdot Ts}\right)^2 \cdot z^{-2}\right] \cdot Scale_{bell}}{1 - 2 \cdot e^{-\frac{B \cdot \Omega_c}{2} \cdot Ts} \cdot \cos\left[\sqrt{\Omega_c{}^2 - \left(\frac{B \cdot \Omega_c}{2}\right)^2} \cdot Ts\right] \cdot z^{-1} + \left(e^{-\frac{B \cdot \Omega_c}{2} \cdot Ts}\right)^2 \cdot z^{-2}} \tag{3-8}$$

$$Scale_{bell} = \frac{1 - 2 \cdot e^{-\frac{B \cdot \Omega_c}{2} \cdot Ts} \cdot \cos\left[\sqrt{\Omega_c{}^2 - \left(\frac{B \cdot \Omega_c}{2}\right)^2} \cdot Ts\right] + \left(e^{-\frac{B \cdot \Omega_c}{2} \cdot Ts}\right)^2}{1 - 2 \cdot e^{-\frac{A \cdot \Omega_c}{2} \cdot Ts} \cdot \cos\left[\sqrt{\Omega_c{}^2 - \left(\frac{A \cdot \Omega_c}{2}\right)^2} \cdot Ts\right] + \left(e^{-\frac{A \cdot \Omega_c}{2} \cdot Ts}\right)^2} \tag{3-9}$$

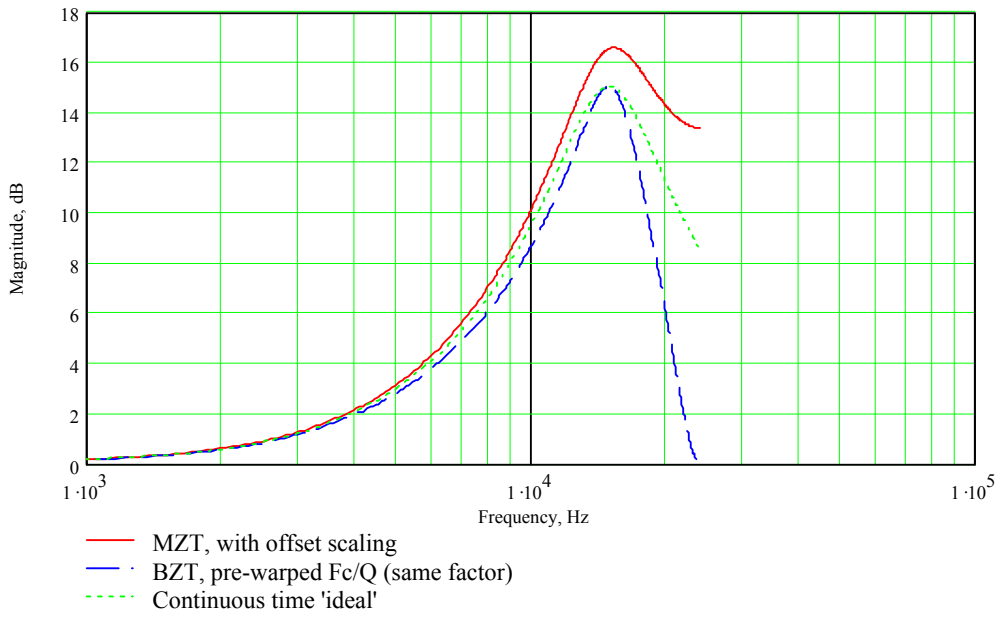

Figure 3-11 Magnitude response comparisons, bell filter, Fc=15 kHz, Q=2, boost gain=15 dB, Fs=48 kHz.

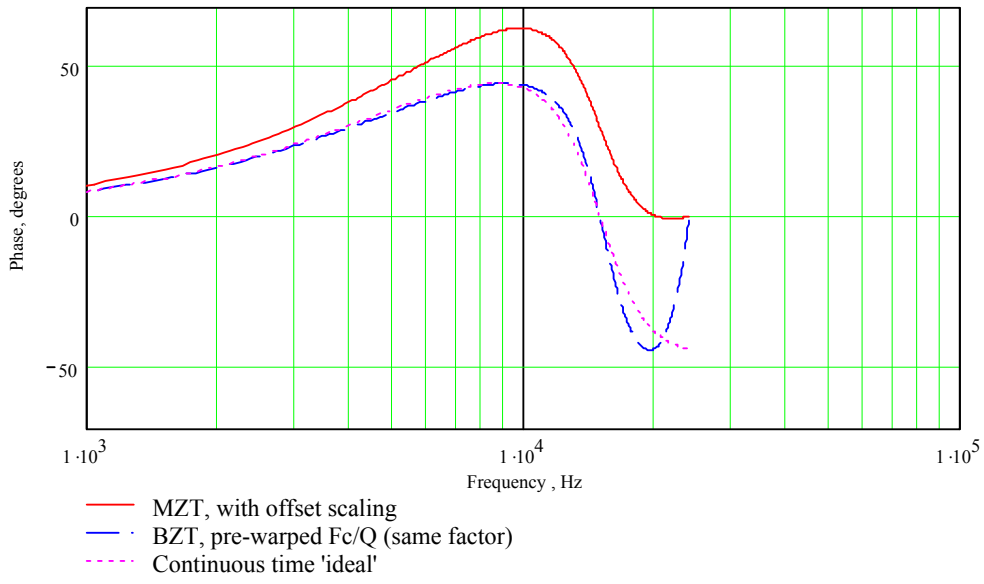

Figure 3-12 Phase response comparisons, bell filter, Fc=15 kHz, Q=2, boost gain=15 dB, Fs=48 kHz.

3.4 Comparison of existing transforms and techniques

3.4.1 LF and HF shelving filter response distortion

MZT offset scaling schemes have been developed to ensure the HF shelving and bell functions are unity gain at dc or the user prescribed gain at dc for the LF shelving filter.

For nominal frequency settings, the LF shelving filter suffers minimal response distortion through either the BZT or MZT. For a worse case scenario, for example Fc equal to two kHz, some response distortion is evident, Figure 3-1, and is worst using the BZT. HF shelving functions, using the BZT, do incur considerable response distortion in the slope of the magnitude frequency response. The MZT, using offset scaling, produces acceptably low amounts of response distortion for both the LF and HF shelving filters, Figure 3-1 to Figure 3-4. For this reason, techniques to improve the MZT response distortion for the LF and HF shelving filters, at the sampling frequency of 48 kHz, are not considered any further in this work.

3.4.2 BZT-based bell filter response distortion

BZT bell filter pre-warping techniques have been reviewed. The BZT bandwidth pre-warping techniques generally perform well, but at high frequency, lower Q settings, bandwidth preservation affects the response of the rising edge of the slope, Figure 3-8. The Fc/lower band-edge pre-warping scheme provides low response distortion and produces a similar response to the Fc/Q pre-warping scheme. However, both schemes suffer response distortion on the falling edge of the slope, Figure 3-9. The Fc/Q pre-warping scheme incurs less computational load than the lower band-edge pre-warping technique (Shpak, 1992). The latter requires additional trigonometric functions. The preservation of gain at the Nyquist frequency scheme (Orfanidis, 1996) resolves the response distortion on the falling edge of the response, Figure 3-8. However, the Nyquist frequency gain and bandwidth preservation scheme does incur large amounts of computational load and rising edge response distortion. Analysis of the z-plane pole and zero placements produced by the BZT, Figure 3-13, shows that, for real root solutions, the BZT has a pronounced zero at Nyquist (a pronounced pole for cut filters). This accentuates the decay in the falling slope of magnitude frequency response towards the

Nyquist frequency, Figure 3-8, Figure 3-9 and Figure 3-11. The presence of 'strong' poles and zeros in the z-plane increases the slope (and Q factor) of the filters magnitude frequency response.

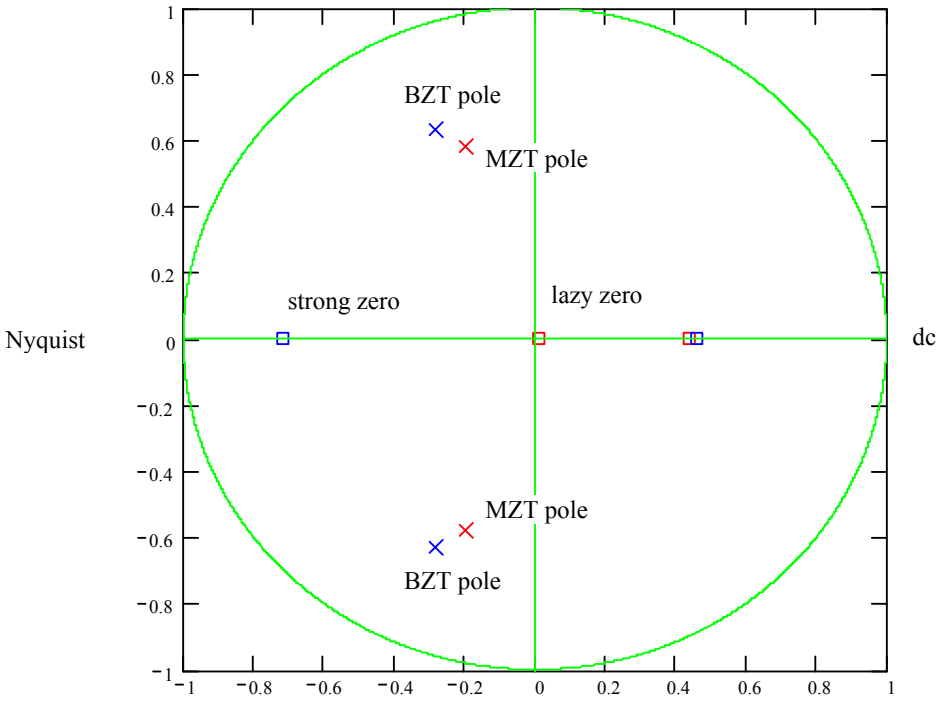

Figure 3-13 Z-plane pole/zero positions, bell filter, Fc=15 kHz, Q=2, boost gain=15 dB, Fs=48 kHz.

3.4.3 MZT-based bell filter response distortion

The offset scaling correction technique has been shown to improve the response distortion of the MZT bell to the extent that the MZT is useable. However at high frequencies and low to mid Q settings, the MZT tends to incur noticeable image distortion, approaching the Nyquist frequency, Figure 3-11. The image distortion can be characterised as a peak gain overshoot in the magnitude response. This is potentially a dangerous response distortion since excessive gain may result in overload clipping of the audio signal. Inspection of the MZT pole zero diagram, Figure 3-13, explains some common characteristics of the MZT. The real zero, in this example, is not placed at or towards Nyquist, but is positioned close to the origin on the dc axis and is therefore virtually

redundant and does not contribute to the overall response. For this reason this is referred to as the 'lazy' zero, Figure 3-13. Note, for the cut case filter the MZT would produce a 'lazy' pole.

3.5 Techniques to improve bell response distortion

3.5.1 Coefficient averaging

It is evident from Figure 3-11 that the MZT (using offset scaling) and BZT (using Fc/Q pre-warping) generate opposing response distortion towards the Nyquist frequency. Averaging the BZT and MZT coefficient sets, as shown in Equation (3-10), produces a resultant filter with a magnitude response that is the average of the BZT and MZT filters. Furthermore the resultant filter's magnitude response renders a good fit to the ideal response. Analysis of some real root pole zero placements produced by the averaged BZT/MZT shows that the MZT generated 'lazy' zero at dc and the BZT generated 'strong' zero at the Nyquist frequency are averaged to produce a 'moderately' positioned zero at the Nyquist frequency. The averaged zero placement is half way between the two zeros, along the dc-Nyquist axis. This has the effect of reducing the excessive peak gain caused by the MZT and the increased slope on the falling edge caused by the BZT. Figure 3-14 shows the resulting magnitude response for a bell filter using a BZT/MZT averaged coefficient set, which can be seen to be close to the ideal response. However the coefficient averaging technique has some side effects. In some cases the averaged response is worse than the individual BZT or MZT responses. At higher centre frequency settings, towards 20 kHz, the excessive peak gain overshoot of the MZT produces a peak gain overshoot in the averaged response. Excessive gain, as stated in Section 3.4.3, is not a desirable effect. At mid to high centre frequency settings, in the region of 8 kHz, the accentuated falling slope characteristic of the BZT is visible in the averaged response.

$$a_i = \frac{a_i(bzt) + a_i(mzt)}{2} \qquad i = 0,1,2$$

$$b_j = \frac{b_j(bzt) + b_j(mzt)}{2} \qquad j = 0,1 \tag{3-10}$$

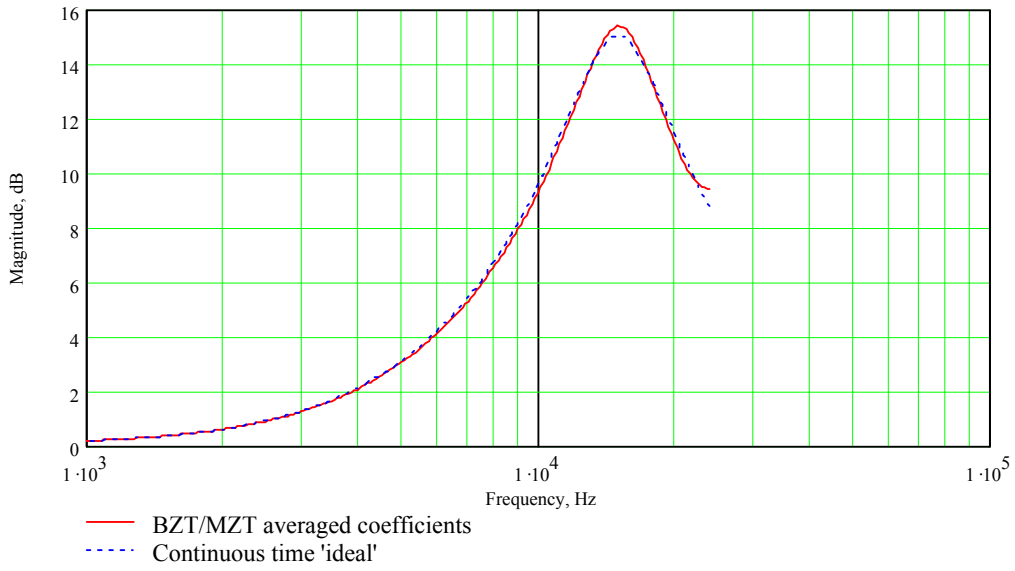

Figure 3-14 Magnitude response comparisons, bell filter, Fc=15 kHz, Q=2, boost gain=15 dB, Fs=48 kHz.

3.5.2 Root shifted MZT

Shifting the MZT generated 'lazy' zero from dc to a moderate position in the Nyquist frequency region of the dc - Nyquist axis combats the excessive peak gain problem, inherent in the MZT. For a boost gain setting the 'lazy' numerator root (zero) will be shifted onto the Nyquist section of the axis. For the cut case, the 'lazy' denominator root (pole) needs to be shifted onto the Nyquist section of the axis. The mechanism to shift the root is simple for the real roots case. The bell transfer function of Equation (3-8) can be rewritten assuming real unequal zero's, obtaining Equation (3-11). It is clear from Equation (3-11) that the 'lazy' root can be effectively shifted across the dc - Nyquist axis

by manipulating the variable 'zshift'. If 'zshift' is negative the 'lazy' zero at dc can be shifted across to the Nyquist region of the axis. The magnitude of 'zshift' determines the position on the axis. The zero requiring the shift is the root solution found from Equation (3-12). The transfer function in Equation (3-11) has to be scaled to combat the usual offset as previously described in Section 3.3.

$$H_{bell}(z) = \frac{1 - \left[e^{realzero1 \cdot Ts} + \left(e^{realzero2 \cdot Ts}\right) \cdot zshift\right] \cdot z^{-1} + \left[e^{realzero1 \cdot Ts} \cdot e^{realzero2 \cdot Ts} \cdot zshift\right] \cdot z^{-2}}{1 - 2 \cdot e^{-\frac{B \cdot \Omega_c}{2} \cdot Ts} \cdot \cos\left[\sqrt{{\Omega_c}^2 - \left(\frac{B \cdot \Omega_c}{2}\right)^2} \cdot Ts\right] \cdot z^{-1} + \left(e^{-\frac{B \cdot \Omega_c}{2} \cdot Ts}\right)^2 \cdot z^{-2}} \tag{3-11}$$

$$realzero2 = \frac{-A \cdot \Omega_c - \sqrt{\left(A^2 - 4\right) \cdot {\Omega_c}^2}}{2} \tag{3-12}$$

A closed form expression for 'zshift' can be realised, where Fc, Q and G are known. Variable 'zshift' can be solved taken that, Equation (3-11), once offset corrected, should equal the prescribed user gain G, at the peak frequency Fc. Figure 3-15 shows the resulting response distortion for a real roots shifted MZT example. The excessive peak gain overshoot has been corrected. However slight response distortion is visible on the rising edge slope. Although the closed form solution for 'zshift' will always produce a peak gain equal to G, the rest of the response can be compromised at extreme settings (typically as Fc approaches Nyquist frequency and at low Q settings).

If the discrete roots are complex then the zero and pole shifting principle has to apply a shifting factor to the complex conjugate pair, Equation (3-13). Various shifting principles have been tried, with the aim of reducing the image response distortion approaching the Nyquist frequency. Simply applying a single shifting parameter to the modulus of the conjugate pair does not solve the peak gain overshoot image response distortion effectively. Another shifting principle, which did reduce the peak gain overshoot in some

isolated cases, required shifting the modulus r, and the phase angle θ by a shifting factor, 'zs', Equations (3-14) and (3-15). No generic shifting principle was found for complex roots. Complex root shifting methods tended to produce response distortion at particular root positions, produce non-conjugate roots or did not produce a closed form solution for the shifting factor.

$$1 - 2 \cdot r \cdot \cos(\theta) \cdot z^{-1} + r^2 \cdot z^{-2} \tag{3-13}$$

$$1 - 2 \cdot e^{-a \cdot Ts} \cdot \cos(b \cdot Ts) \cdot z^{-1} + \left(e^{-a \cdot Ts}\right)^2 \cdot z^{-2} \tag{3-14}$$

$$a = \frac{A \cdot \Omega_c}{2} \cdot zs \qquad b = \sqrt{\Omega_c{}^2 - \left(\frac{A \cdot \Omega_c}{2} \cdot zs\right)^2} \tag{3-15}$$

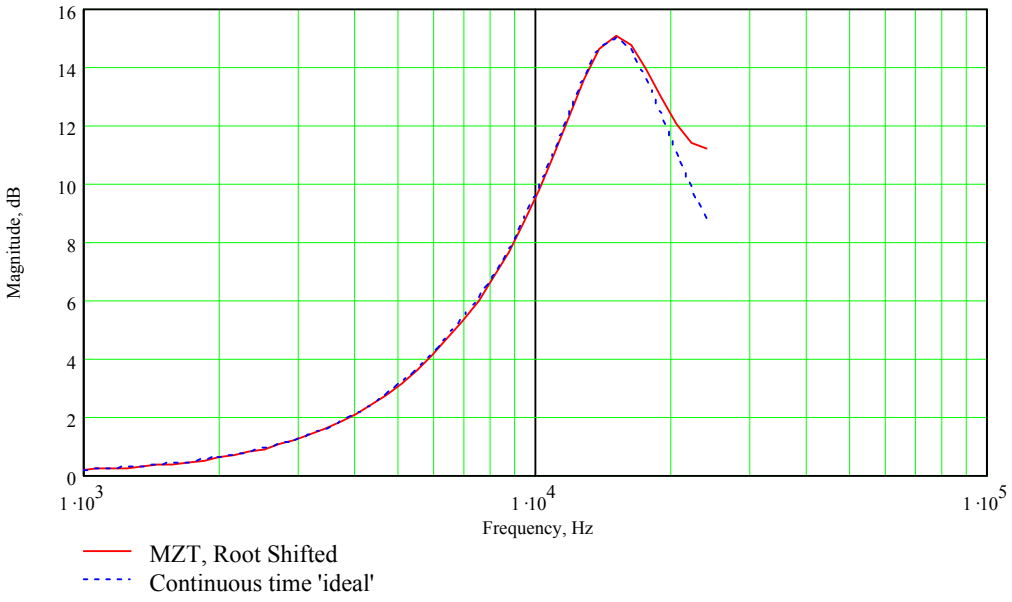

Figure 3-15 Magnitude response comparisons, bell filter, Fc=15 kHz, Q=2, boost gain=15 dB, Fs=48 kHz.

3.5.3 Peak gain pre-warping for MZT

In this work another technique is developed, a peak gain pre-warping, to counteract the excessive peak gain error inherent in the MZT. This is essentially a first order, peak gain error correction process. The actual peak gain of the basic MZT response can be obtained by substituting z, Equation (3-16), into the discrete-time transfer function, Equation (3-8). The peak gain error, Equation (3-17) is the difference between the actual peak gain and the desired peak gain, G. The peak gain error is then subtracted from the desired gain to form a pre-warped gain, see Equation (3-18). This pre-warped gain is then applied back into the MZT, providing a first order, peak gain error corrected response. The resulting response is free from peak gain overshoot, but typically suffers slope distortion, as shown in Figure 3-16. This is a generic solution, applicable to any given bell filter. This technique is thus preferred to the shifting process, Section 3.5.2, which only works for real root solutions. However the computational load is high since two implementations of the MZT mapping and offset scaling scheme are required.

$$z = e^{j \cdot 2 \cdot \pi \cdot Fc \cdot Ts} \tag{3-16}$$

$$PeakGainError = \left| Hbell\left(e^{\frac{j \cdot 2 \cdot \pi \cdot Fc}{Fs}} \right) \right| - G \tag{3-17}$$

$$PrewarpGain = G - PeakGainError \tag{3-18}$$

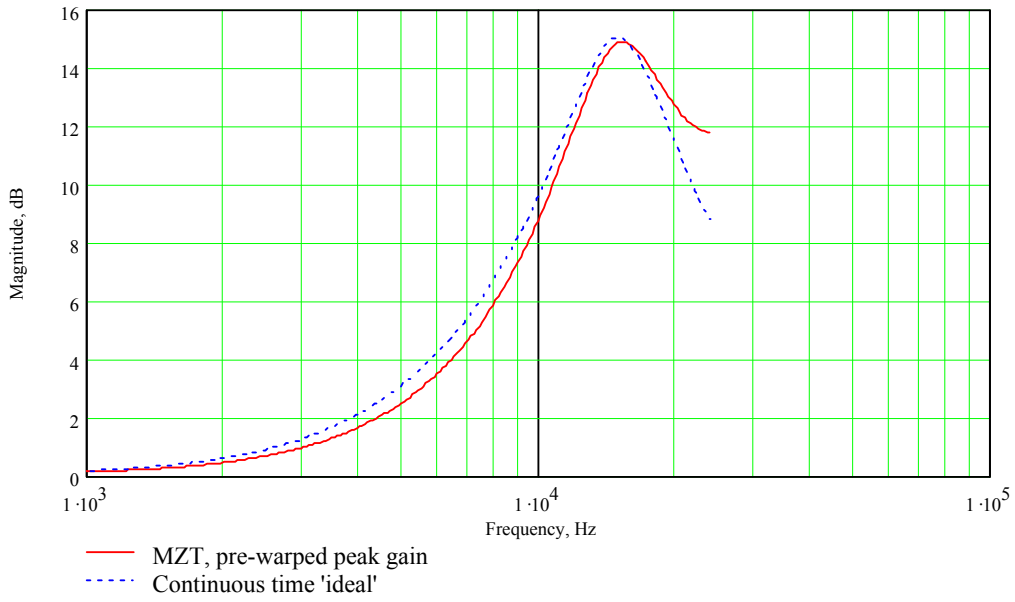

Figure 3-16 Magnitude response comparisons, bell filter, Fc=15 kHz, Q=2, boost gain=15 dB, Fs=48 kHz.

3.6 Assessment of frequency response distortion

Frequency response error curves, displaying the response differences (magnitude or phase) between the ideal s-plane response and that of a discrete-time filter, provide a useful way to compare transform schemes. Figure 3-17 shows the magnitude response errors for the BZT, MZT (offset scaled), averaged BZT/MZT and MZT (with peak gain pre-warping), for a bell filter, with Fc=15 kHz, Q=2, G=15 dB. The MZT peak frequency gain error (overshoot) is shown to be nearly 2 dB. The average BZT/MZT technique offers the lowest magnitude error, but does have some overshoot at the peak gain frequency. The MZT (with peak gain pre-warping) has no peak gain overshoot but displays some response error in the mid frequency range (less than 1dB error). Figure 3-18 depicts the magnitude error curve for a more typical filter setting, with the peak gain at 6 kHz. The MZT has approximately 0.25 dB overshoot at peak gain whereas the MZT (with peak gain pre-warping) has zero peak gain error. The BZT and averaged BZT/MZT incur more error on the falling slopes, which increases towards the Nyquist frequency.

The phase response distortion of the MZT and BZT methods with respect to the s-plane 'ideal' phase response can be effectively shown through a group delay error curve. That is the deviation in group delay of the MZT or BZT generated responses from the s-plane 'ideal' group delay response. The MZT and MZT (with peak gain pre-warping) produce extremely similar phase/group delay responses and for clarity only one 'MZT' plot is shown in Figure 3-19. It is clear from Figure 3-19, that the MZT phase error is linear, producing a flat group delay error. Whereas the BZT group delay error is smaller, but is a highly non-linear phase error (not a flat group delay error) from the ideal s-plane response.

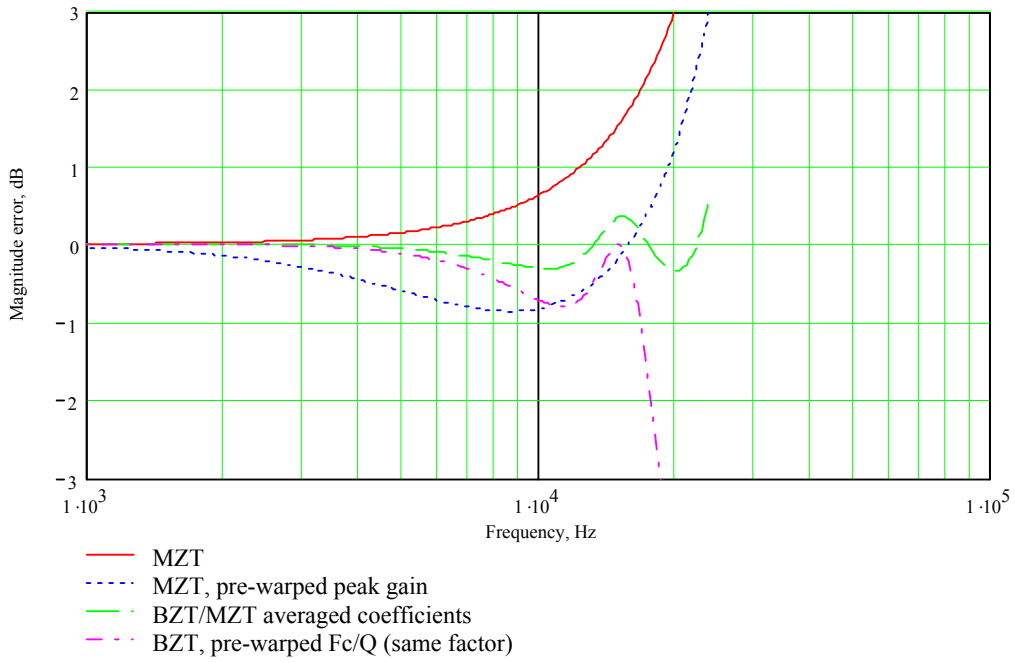

Figure 3-17 Magnitude response error comparisons, bell filter, Fc=15 kHz, Q=2, boost gain=15 dB, Fs=48 kHz.

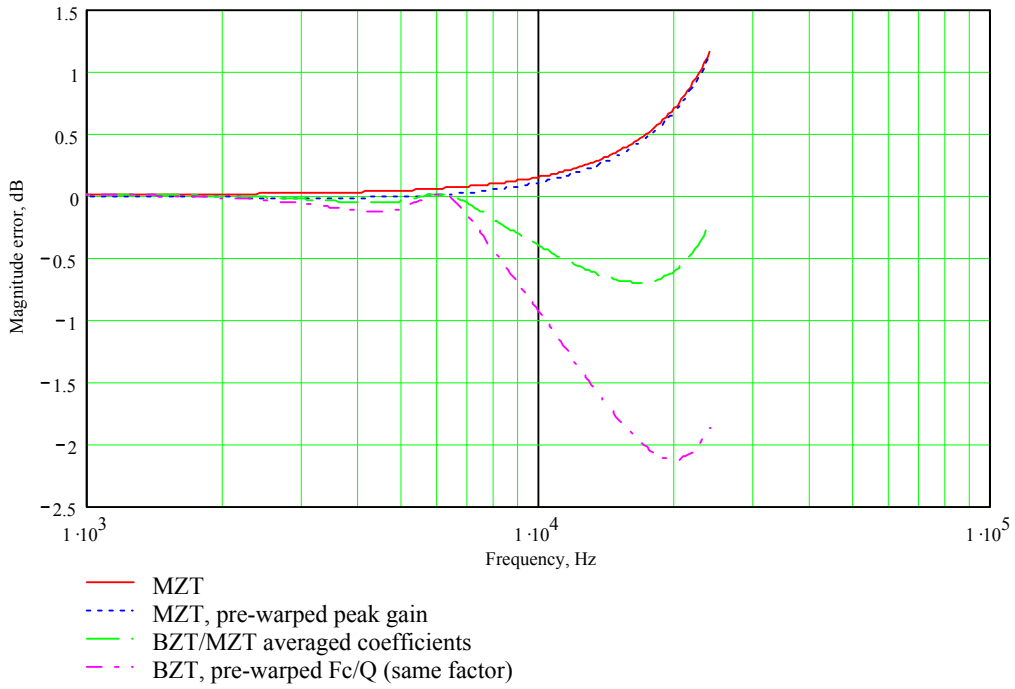

Figure 3-18 Magnitude response error comparison, bell filter, Fc=6 kHz, Q=2, boost gain=15 dB, Fs=48 kHz.

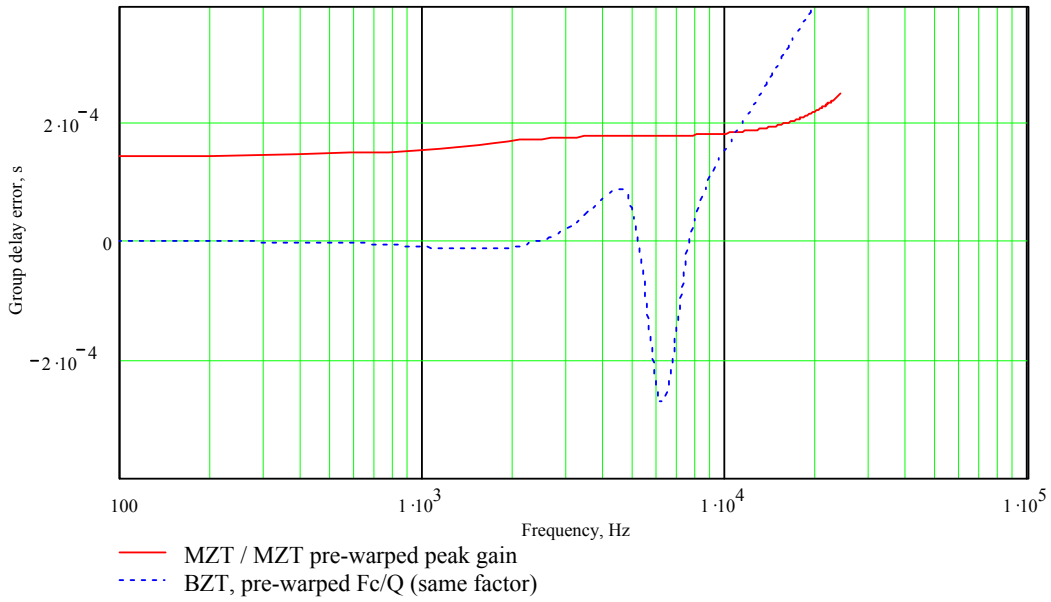

Figure 3-19 Group delay response error comparison, bell filter, Fc=6 kHz, Q=2, boost gain=15 dB, Fs=48 kHz.

3.7 Stability analysis and noise comparisons of target filters

For real and complex roots, the BZT pole positions are typically closer to the unit circle than the MZT pole positions, for high frequency tuned filters. The BZT typically places a

pronounced real root at the Nyquist frequency or a pair of 'strong' complex conjugate roots towards Nyquist. These roots are zeros for the boost gain case and poles for the cut gain case. Nevertheless, these pronounced 'strong' roots force the other roots in the system to be closer to the unit circle, as shown in Figure 3-13. As established, the MZT based methods do not produce the same pronounced 'strong' zero or pole at or towards the Nyquist frequency. Consequently, for high frequency tuned filters MZT pole positions are not as close to the unit circle as the BZT pole positions. These changes in pole positions have some bearing on the finite wordlength performance of the resulting filters. It should be noted that for the constant-Q boost case filter, the pole positions are not dependent on the amount of gain. Therefore the MZT and MZT (with peak gain pre-warping) produce the same pole positions for all boost settings (including the unity gain 0 dB setting).

3.7.1 Resilience to forced overflow oscillation

Pole positions and their susceptibility to forced overflow problems can be visualised through a stability triangle (Dattorro, 1988; Samueli and Willson, 1983). Low frequency tuned filters using the MZT and BZT produce identical pole positions and are both susceptible to forced overflow oscillations, producing pole positions in the lower right hand corner of the triangle. However, interesting stability triangle differences exist for high frequency tuned filters. Figure 3-20 shows BZT and MZT stability triangle pole positions, for a one third octave bell filter, Q=4.35, G=3 dB Fc=19 kHz. It is clear from Figure 3-20 that the BZT produces poles which are within the region where forced overflow oscillations can occur (inside the lower left hand side region). The MZT pole positions, for the same setting, are such that forced overflow oscillations are not possible. For all boost gain settings and the unity gain 'flat' case, it can be shown that for high frequency peak gain and Q, the BZT pole positions are always closer to the forced overflow oscillation region than the MZT pole positions. This suggests that for high

frequency tuned, high Q settings BZT based filters can be more susceptible to forced overflow oscillations than MZT based designs. This is true for all boost settings, including low gain filters and the unity gain case.

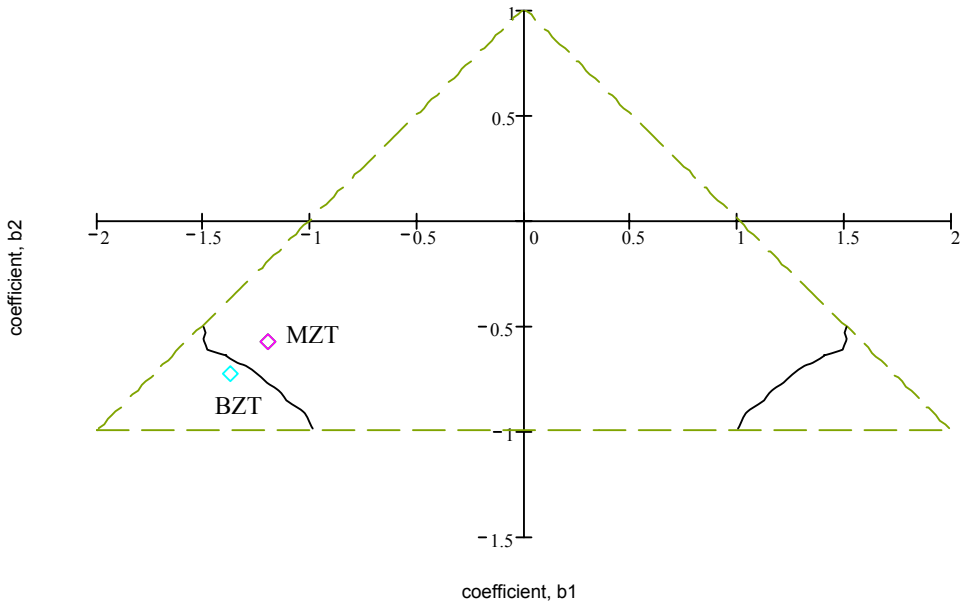

Figure 3-20 Stability triangle, bell filter, Fc = 19 kHz, Q=4.35 (1/3octave). Lower corners are regions of forced overflow oscillation, Fs=48 kHz.

3.7.2 Noise performance

For some filter topologies the pole transfer function is solely responsible for the overall noise gain, (Dattorro, 1988; Wilson, 1993; Clark et al, 1996). Figure 3-21 shows the pole magnitude responses for a MZT and BZT derived bell filter, Fc of 19 kHz, one third octave (Q=4.35). The pole responses are the same for all boost settings (including the 0 dB unity gain case). It should be noted that the BZT pole response produces nearly 4 dB of additional peak gain than the MZT pole response. The pole responses for the same filter in the cut case also show that the BZT produces more noise gain towards Nyquist than the MZT techniques. It is suggested that distortion and noise measurements of a BZT based high frequency tuned filter could be marginally worse than the MZT based filter, even for the unity gain 'flat' setting.

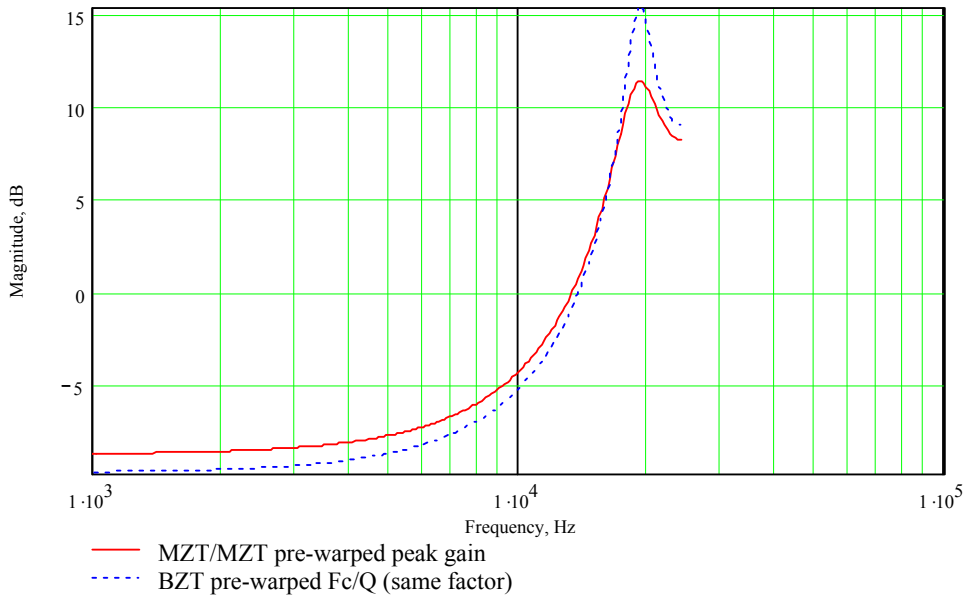

Figure 3-21 Pole transfer function comparisons, bell Filter, Fc=19 kHz, Q=4.35 (1/3octave), boost gain and unity gain (flat) case, Fs=48 kHz

3.8 Effects of higher sampling frequencies on response distortion

The emergence of higher system sampling frequencies, 96 and 192 kHz, is a testimony to the importance of imperfections in transient and frequency response up to and above 20 kHz. However, the tuned frequencies of analogue equalisers do not typically exceed 20 kHz. Consequently, the use of a higher sampling frequency reduces response distortion since the Nyquist frequency is well beyond the range of the tuned frequencies in the filter. The impact of the higher sampling rates on response distortions for the various coefficient generation techniques needs to be evaluated. This will establish whether the computational overhead of the various pre-warping schemes can be justified at higher sampling frequencies.

One obvious computational saving in the MZT method is the removal of the offset scaling correction. However, this is not feasible since the offset at higher sampling frequencies is still significant. At a sampling frequency of 192 kHz the offset for a bell filter (Fc=15

kHz, Q=2, 15 dB boost) is an unacceptable 3 dB. Computational savings, which result in acceptable response distortion, can be made through the omission of the BZT pre-warping step, omitting Equations (2-8) and (3-1) in the BZT mapping. These effects are filter type and sampling frequency dependent and are discussed in the following sections.

3.8.1 High pass filter response distortion at 96 and 192 kHz

Figure 3-22 shows the response distortion errors for a Butterworth response HPF tuned to 10 kHz, using the various BZT techniques, operating at a sampling frequency of 96 kHz. For the BZT (using no pre-warping) the maximum error is at the tuned frequency and is shown to be approximately 0.3 dB. The BZT (prewarping Fc) produces a maximum slope distortion of approximately 0.6 dB. The BZT (prewarping Fc/Q) produces a maximum error at the tuned frequency of approximately 0.4 dB.

Figure 3-24 shows the response distortion errors for a Butterworth response HPF tuned to 10 kHz, using the various BZT techniques, operating at a sampling frequency of 192 kHz. For the BZT using no pre-warping the maximum error is at the tuned frequency and is shown to be less than 0.1 dB. The BZT (prewarping Fc) produces a maximum slope distortion of approximately 0.16 dB. The BZT (prewarping Fc/Q) produces a maximum error at the tuned frequency of less than 0.09 dB.

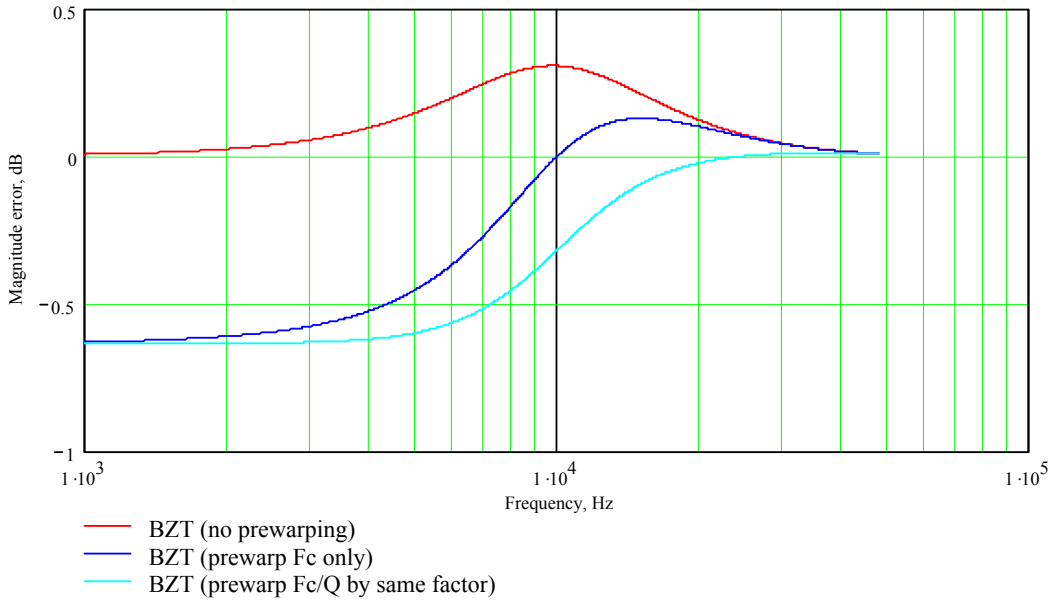

Figure 3-22 Magnitude response error comparison, high pass filter, Fc=10 kHz, Fs=96 kHz.

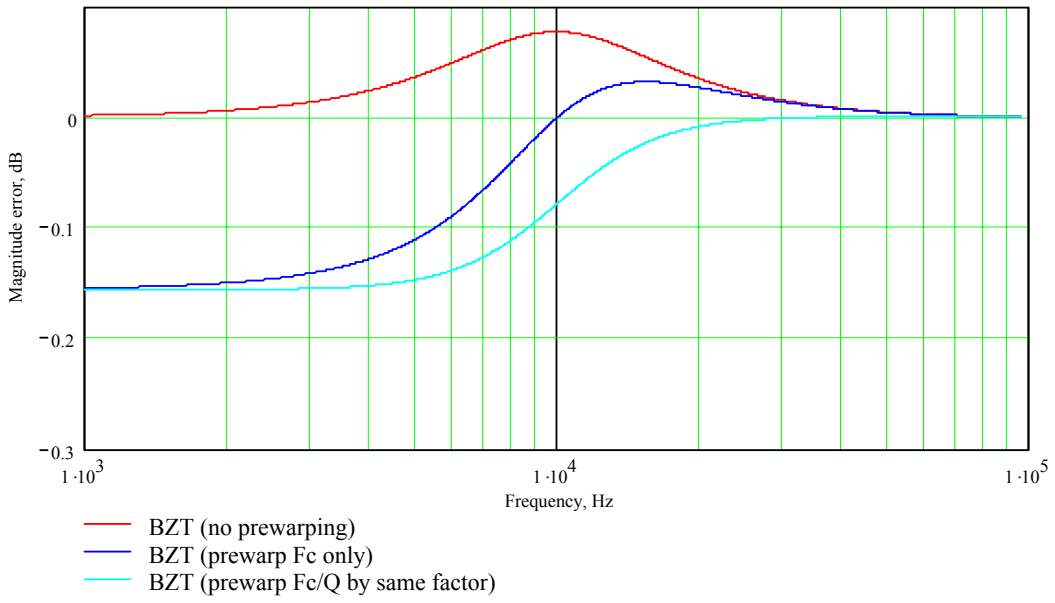

Figure 3-23 Magnitude response error comparison, high pass filter, Fc=10 kHz, Fs=192 kHz.

3.8.2 Low pass filter response distortion at 96 and 192 kHz

Figure 3-24 shows the response distortion errors for a Butterworth response LPF tuned to 10 kHz, using the various BZT techniques, operating at a sampling frequency of 96 kHz. For the BZT (using no pre-warping) produces a noticeable increase in slope roll-off and is

approaching 3 dB error at 20 kHz. The error at the tuned frequency (10 kHz) is approximately 0.3 dB. The BZT (prewarping Fc) produces a slope distortion of approximately 2 dB at 20 kHz. The BZT (prewarping Fc/Q) produces a maximum error at the tuned frequency of approximately 0.3 dB.

Figure 3-25 shows the response distortion errors for a Butterworth response LPF tuned to 10 kHz, using the various BZT techniques, operating at a sampling frequency of 192 kHz. All three schemes produce considerably less slope roll-off distortion, approximately 0.5 dB error at 20 kHz. The BZT (prewarping Fc) and BZT (prewarping Fc/Q) produce a maximum error at the tuned frequency of less than 0.1 dB.

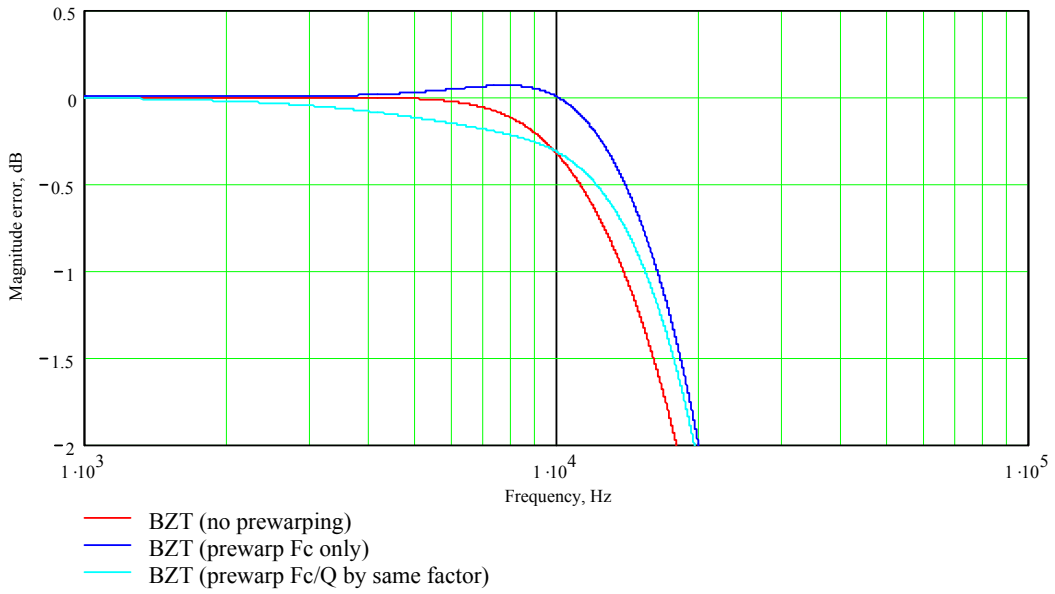

Figure 3-24 Magnitude response error comparison, low pass filter, Fc=10 kHz, Fs=96 kHz.

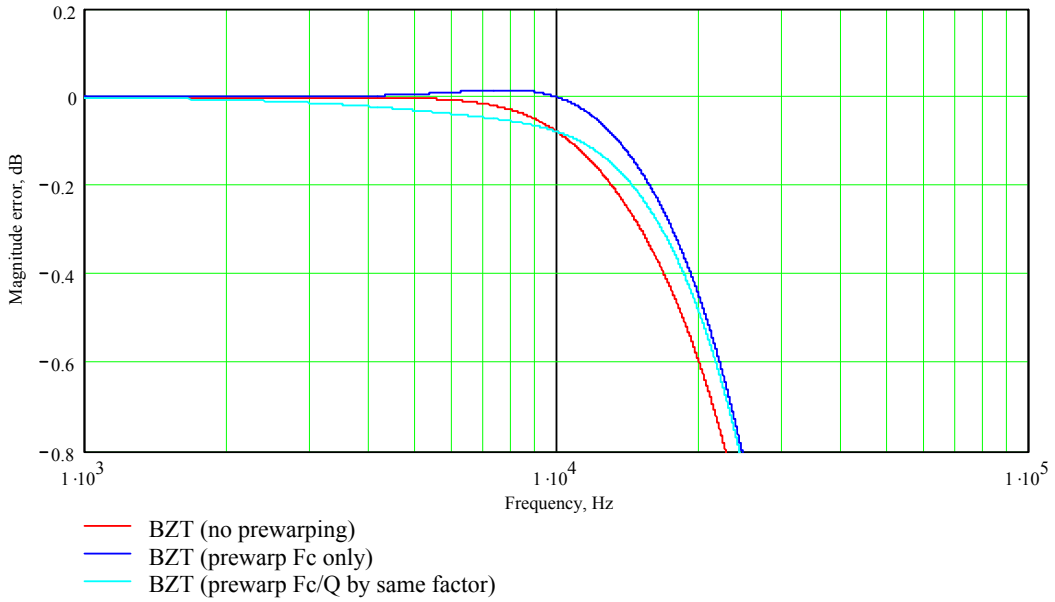

Figure 3-25 Magnitude response error comparison, low pass filter, Fc=10 kHz, Fs=192 kHz.

3.8.3 LF shelving filters response distortion at 96 and 192 kHz

It is suggested in Section 3.4.1 that the BZT (pre-warping Fc) and the MZT (with offset scaling), produce acceptable response distortion for worst case settings, at a sampling frequency of 48 kHz. It has also been found that the BZT, at a sampling frequency of 48 kHz, does not need pre-warping for typical settings (Fc is less than 1 kHz). Figure 3-26 shows the magnitude response error for a LF shelving filter (Fc of 2 kHz) using the BZT (with no pre-warping) and the MZT (with offset scaling) at a sampling frequency of 96 kHz. The MZT magnitude error at 20 kHz is less than 0.02 dB. The BZT magnitude error is less than 0.4 dB at 20 kHz. Figure 3-27 shows the magnitude response error for a LF shelving filter (Fc of 2 kHz) using the BZT (with no pre-warping) and the MZT (with offset scaling), at a sampling frequency of 192 kHz. The MZT magnitude error at 20 kHz is approximately 0.002 dB. The BZT magnitude error is less than 0.1dB at 20 kHz.

It is clear that, at the higher sampling frequencies, the MZT (with offset scaling) still produces much less response distortion than the BZT (with no pre-warping). However the

magnitude response errors for the BZT (using no pre-warping) at 96 kHz and 192 kHz are almost negligible.

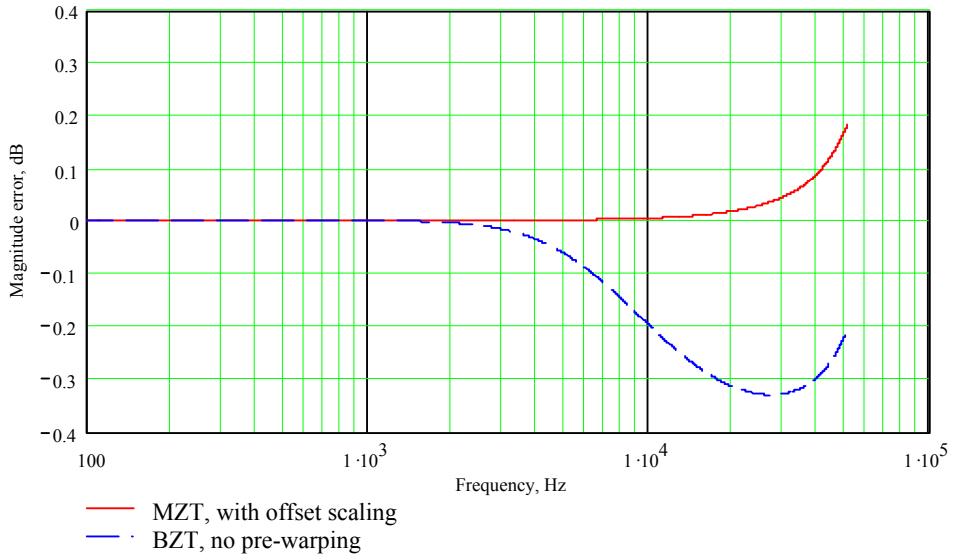

Figure 3-26 Magnitude response error comparison, LF shelving filter, Fc=2 kHz, boost gain=15 dB, Fs=96 kHz

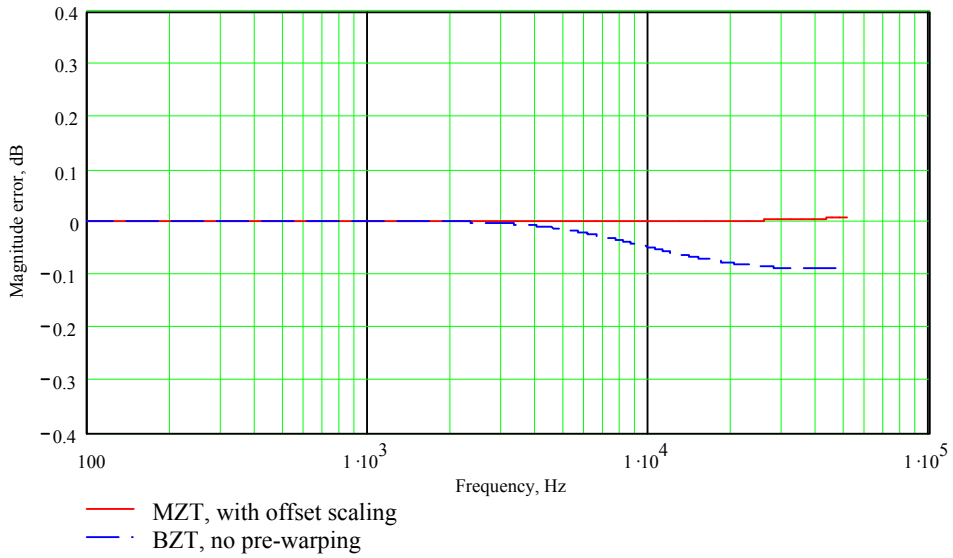

Figure 3-27 Magnitude response error comparison, LF shelving filter, Fc=2 kHz, boost gain=15 dB, Fs=192 kHz

3.8.4 HF shelving filters response distortion at 96 and 192 kHz

MZT HF shelving filter response distortion is minimal at a sampling rate of 48 kHz. The BZT, at a sampling frequency of 48 kHz, produces considerable response distortion,

drastically increasing the slope in the magnitude response. BZT (with no pre-warping) at a sampling frequency of 48 kHz incurs further slope distortion. Figure 3-28 shows the magnitude response error for a HF shelving filter (Fc of 12 kHz) using the BZT (with no pre-warping) and the MZT (with offset scaling) at a sampling frequency of 96 kHz. The BZT (no pre-warping) produces less than 0.2 dB error. Figure 3-29 shows the magnitude response error for the BZT (no pre-warping) and the MZT at a sampling rate of 192 kHz. The BZT (no pre-warping) produces a maximum error of less than 0.05 dB (at 20 kHz). It is also shown that the MZT magnitude response errors, up to 20 kHz, at the sampling frequencies of 96 kHz and 192 kHz are extremely low (less than 0.01 dB).

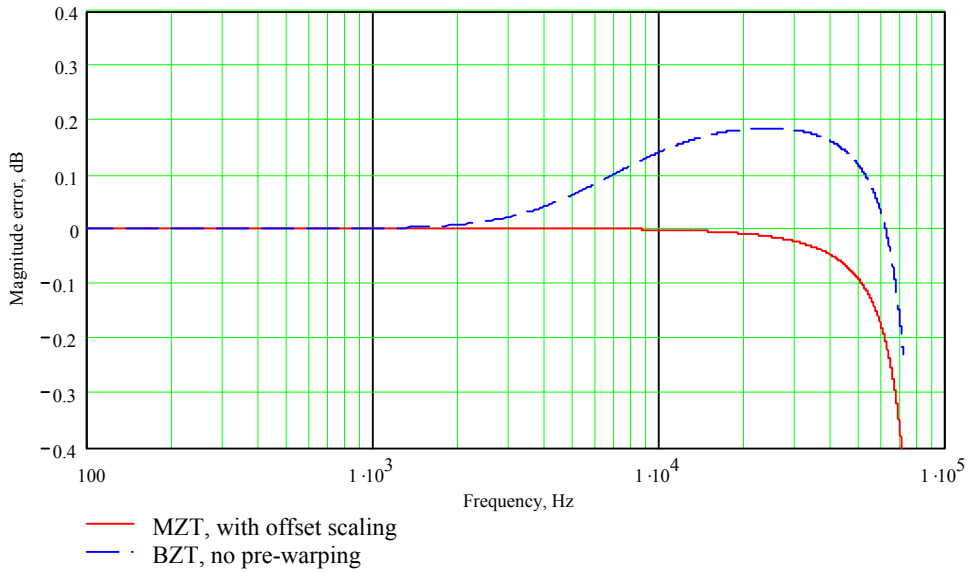

Figure 3-28 Magnitude response error comparison, HF shelving filter, Fc=12 kHz, boost gain=15 dB, Fs=96 kHz

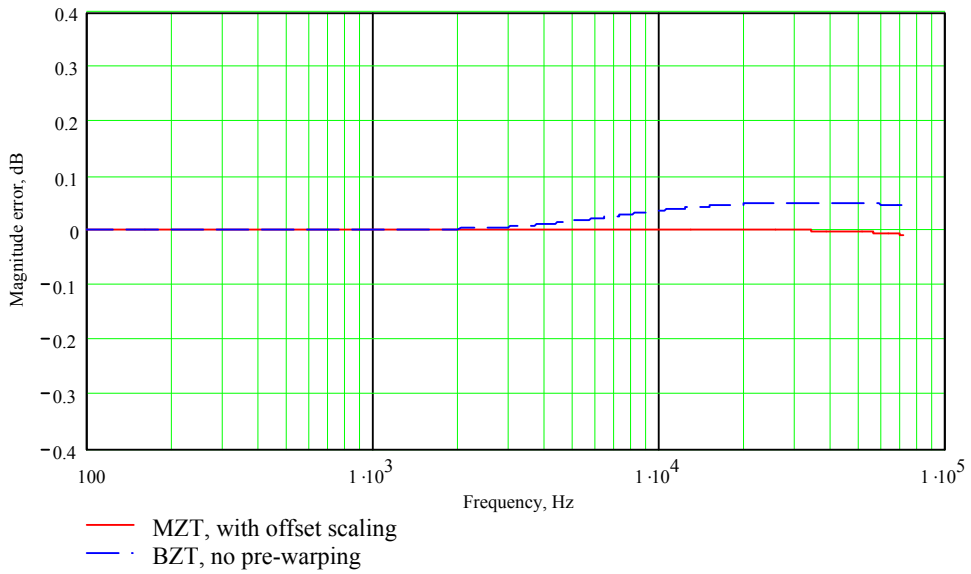

Figure 3-29 Magnitude response error comparison, HF shelving filter, Fc=12 kHz, boost gain=15 dB, Fs=192 kHz

3.8.5 Bell filter response distortion at 96 and 192 kHz

Figure 3-30 shows the magnitude response errors, for various schemes, for a bell filter (Fc=15 kHz) at a sampling frequency of 96 kHz. The MZT (with offset scaling), without peak gain pre-warping, produces 0.15dB overshoot at the peak gain frequency. The respective magnitude error curve, at a sampling frequency of 48 kHz is shown in Figure 3-17. This suggests that the MZT peak gain pre-warping scheme is not required to combat peak gain overshoot problems in the bell filter magnitude response, at a sampling frequency of 96 kHz.

Further inspection of Figure 3-30 shows that the BZT (pre-warping Fc/Q) technique produces 0.7dB error at 20 kHz. By omitting Equations (2-8) and (3-1) from the BZT mapping, the potential of no pre-warping can be explored. It can be shown that a system sampling frequency of 96 kHz does not alleviate the need for pre-warping Fc/Q in the BZT mapping. The implementation of the 15 kHz bell filter using the BZT (without pre-warping) results in a peak magnitude frequency of 14 kHz. Such an error in the peak

magnitude frequency is not acceptable. Furthermore the response also suffers from noticeable slope distortion and the magnitude response error at 20 kHz is 2.5 dB.

Figure 3-31 shows the magnitude response errors for the BZT, BZT (with no pre-warping) MZT (with offset scaling), MZT (with peak gain pre-warping) at a sampling frequency of 192 kHz. The MZT (with peak gain pre-warping) scheme does not produce a significant difference in magnitude error than that of the standard MZT. The standard MZT magnitude error at the peak frequency (15 kHz) is less than 0.01 dB, rendering the error negligible. The BZT (without pre-warping) can also produce acceptable response errors at a sampling frequency of 192 kHz. Implementing the 15 kHz bell filter with the BZT (without pre-warping) results in a peak magnitude frequency of 14.75 kHz. This is thought not acceptable.

The MZT group delay error responses at the higher sampling rates are also fairly flat and small in magnitude. At a sampling frequency of 96 kHz the group delay error for the 6 kHz bell filter is approximately 50μs. At 192 kHz the group delay error for the same bell filter is approximately 10μs (compared to 180μs at the sampling frequency of 48 kHz). At sampling frequencies of 96 and 192 kHz the BZT (pre-warping Fc/Q) produces a more non-linear group delay error, similar in response to the 48 kHz sampling rate results, Figure 3-19. However the magnitude of this group delay error is again small, at a sampling frequency of 96 kHz the BZT produces a peak group delay error, at 6.5 kHz, of 70μs. At 192 kHz the peak group delay error is 15 μs.

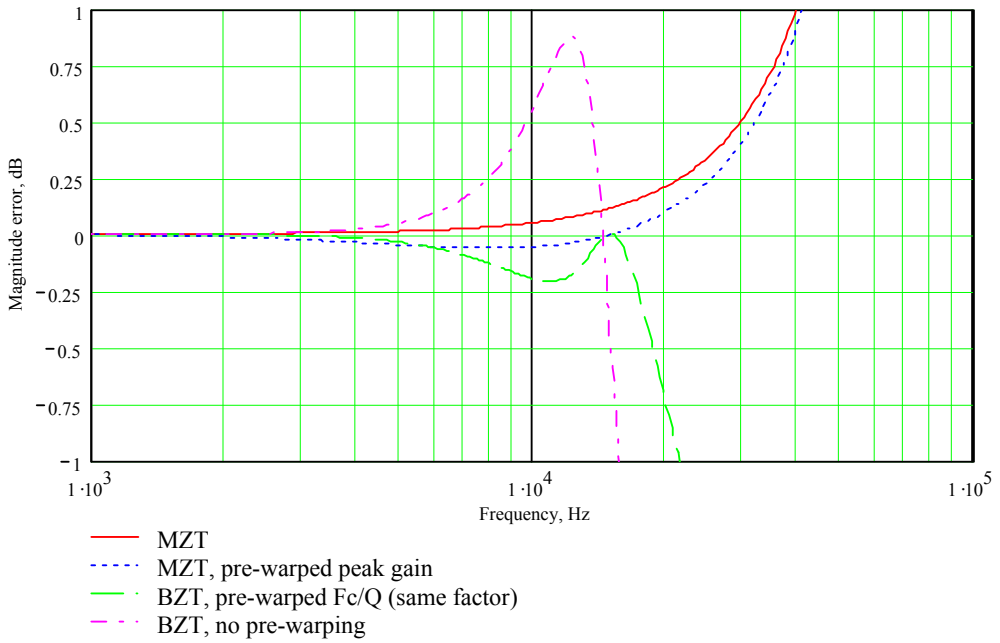

Figure 3-30 Magnitude response error comparison, bell filter, Fc=15 kHz, Q=2, boost gain=15 dB, Fs=96 kHz

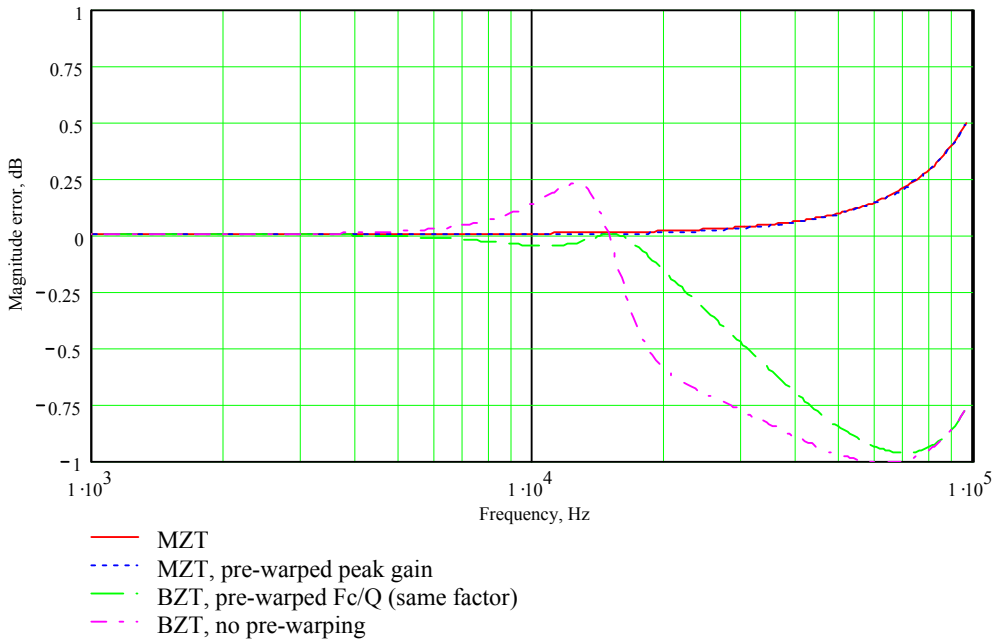

Figure 3-31 Magnitude response error comparison, bell filter, Fc=15 kHz, Q=2, boost gain=15 dB, Fs=192 kHz

3.9 Implementation of techniques in DSP

3.9.1 Comparisons of computational loads

Both the BZT and MZT techniques require a large operational numerical range and are naturally more suited to implementation in floating point arithmetic. Implementation in fixed point arithmetic is feasible through arithmetic scaling or through the use of look-up tables to circumvent arithmetic overflow. However, arithmetic scaling reduces the computational accuracy of the resulting coefficients and the use of look-up tables reduces parameter resolution (since the number of parameter settings depends on look-up table capacity). There are two implementation platform scenarios, implementation on DSP hardware or on a standard microprocessor platform. Implementation on DSP hardware can exploit high order polynomials to approximate functions since DSP processors are efficient at multiplication. Furthermore a ratio of polynomials can also be used for approximations, since efficient divide iterations (Cavanagh, 1984) are typically achievable on DSP platforms. Many standard microprocessors are not efficient at multiplication. The CORDIC algorithm (Volder, 1959) is an efficient and widely adopted technique used to implement function approximation on multiplier-less processors. Cosine/Sine, Cosh/Sinh functions are simple to implement. The Cosh/Sinh CORDIC processor can also be used to calculate an exponential. Tangent functions can be derived from the Cosine/Sine by the use of a divide function. Newton-Raphson's iteration (Cavanagh, 1984) is an efficient method to calculate the square-root function.

Table 3-1 gives a break-down of the computational load of the various key transforms, once optimised. The MZT (with peak gain pre-warping) and the BZT (bandwidth and Nyquist gain preserved) clearly require the highest computational load. It is estimated that the computational load of these two techniques is fairly similar. It is also clear from Table 3-1 that the BZT (pre-warping Fc/Q) has considerably less computational load than the

standard MZT (with offset scaling). It is clear that the BZT (with no pre-warping) is the method requiring least computational load.

Arithmetic Function	BZT No pre-warping	BZT Pre-warping Fc & Q	MZT Gain shift corrected	MZT Peak gain pre-warping	BZT BW & Nyquist gain preservation
Multiplication	12	15	17	32	31
Addition	6	6	6	17	36
Division	2	2	2	4	6
Square Root	0	0	2	3	7
Exponential	0	0	2	4	0
Cosine/Cosh	0	0	2	3	0
10^X	1	1	1	1	2
Tangent	0	1	0	0	2

Table 3-1 Computational load comparison of the various coefficient calculation techniques

3.10 Discussion

Magnitude and phase response distortions for BZT and MZT–based filters were examined. Offset response distortions in the MZT-based filters have been identified. Offset correction schemes are developed for the LF, HF shelving and bell filters. Image response distortions in the form of excessive peak gain are found in the MZT-based bell filter. Correction schemes were developed to improve the MZT bell filter image response distortion. It is shown that pre-warping the peak gain in the MZT results in a relatively affordable computational increase and improves the response of the resulting filters. It is also shown that for high frequency tuned filters the stability and noise characteristics of the MZT-based filter is superior to that of a BZT-based filter. This is even true for a flat (unity gain) filter.

Filter response distortions using existing BZT pre-warping techniques operating at a sampling frequency of 48 kHz were reviewed. BZT-based LF shelving filters produce negligible response distortion and are computationally efficient. BZT pre-warping techniques produce large amounts of response distortion in HF shelving filters. Prewarping Fc and Q is found to be the best BZT technique for the bell filter. However, prewarping Fc is found to be the best BZT technique for LPF and HPF.

MZT-based bell, LF and HF shelving filters have been shown to produce less response distortion than any of the BZT techniques at the sampling frequencies of 48 kHz, 96 kHz and 192 kHz. However, the response distortions are considerably smaller at the higher sampling frequencies. At sampling frequencies of 96 and 192 kHz, it was found that the MZT pre-warped peak gain technique is redundant and the MZT (with offset scaling) is sufficient since there is no significant overshoot in the peak gain. At a sampling frequency of 96 kHz the BZT (pre-warped Fc/Q) response distortions are extremely small. At a sampling frequency of 192 kHz the BZT (pre-warping Fc/Q) and MZT (with offset scaling) response distortions are negligible furthermore the BZT (without pre-warping) could be used, reducing the computational load considerably. However, the bell filter response error at a sampling frequency of 96 kHz for the BZT (no pre-warping) is considerable.

4 Emulation of finite wordlength arithmetic

4.1 Introduction

The aim of the work described in this chapter was to develop finite wordlength arithmetic functions for use in a filter topology emulation environment. The emulation environment is developed in Mathcad (Mathsoft, 1998). The work examines the effects of quantisation in sign magnitude and twos-complement binary coded data. Mathcad functions are developed to emulate the effects of quantisation in fixed and floating point arithmetic. The emulated finite wordlength arithmetic is compared at bit level to IEEE 32 bit (single precision) floating point arithmetic and 32 bit fixed point twos-complement arithmetic.

4.2 Quantisation models

As discussed in Chapter 2 Section 2.4, finite wordlength arithmetic produces data quantisation errors in filter topologies. The two forms of quantisation are truncation (round to zero) and rounding (round to nearest). Furthermore there are two binary data coding schemes considered in this work, sign magnitude and twos-complement. Quantisation error, Qe, can be defined as the difference between the quantised data, Q(x) and the unquantised data, x, Equation (4-1).

$$Q_e = Q(x) - x \tag{4-1}$$

The truncation of sign magnitude coded data produces a zero biased, bipolar quantisation error. The quantisation error is bipolar because the truncation of sign magnitude data always produces a smaller magnitude, irrespective of the sign of the data. Therefore truncating positive data produces a negative quantisation error and truncating negative data produces a positive quantisation error. Sign magnitude truncation can be modelled using the *floor* and *ceil* functions [1, 2]. If the data is positive then the *floor* function is used, for example,

$$\text{floor}(10.1) = 10 \quad .$$

If the data is negative then the *ceil* function is used, for example,

$$\text{ceil}(-10.1) = -10 \quad .$$

In order to perform quantisation at any specified fractional binary point a shifting factor has to be applied to the data before and after the *floor* and *ceil* functions. The data is shifted such that the target fractional wordlength is to the left of the decimal point. Then application of the *floor* or *ceil* function truncates the unwanted data to the right of the decimal point. After the *floor* and *ceil* operation, the truncated data is realigned with the inverse of the shifting factor. A Mathcad sign magnitude truncation function is shown in Equation (4-2). Data x, is truncated assuming the data is represented by n, sign magnitude coded, fractional binary bits.

$$\text{smtrc}(x) := \begin{cases} \left(\text{floor}\left(x \cdot 2^{n}\right) \cdot 2^{-n}\right) & \text{if } x \geq 0 \\ \left(\text{ceil}\left(x \cdot 2^{n}\right) \cdot 2^{-n}\right) & \text{otherwise} \end{cases} \tag{4-2}$$

[1] *floor*(x) returns the greatest integer that is less than or equal to x.
[2] *ceil* (x) returns the smallest integer that is greater than or equal to x.

Twos-complement coding is intrinsically different for positive and negative data representations. Truncated twos-complement coded data becomes smaller in magnitude for positive data and larger in magnitude for negative data. This produces a negative bias, unipolar quantisation error. Twos-complement truncation can simply be modelled through the use of the *floor* function, Equation (4-3).

$$\text{twosCtrc}(x) := \text{floor}\left(x \cdot 2^{n}\right) \cdot 2^{-n} \tag{4-3}$$

The basic principle of rounding (*to the nearest*) follows the rule that if the residual magnitude below the quantisation binary point is less than or equal to one half of the LSB of the quantised wordlength then the data is truncated and the LSB remains unchanged. If the residual magnitude is greater than one half of the LSB of the quantised wordlength then one LSB is added to the magnitude of the data. Rounding can easily be implemented by adding a rounding quanta (one half of an LSB at the fractional bit boundary) to the pre-quantised data and then truncating the result. The Mathcad functions developed to emulate rounding (*to the nearest*) for twos-complement and sign magnitude coded data are given in Equations (4-4) and (4-5) respectively.

$$\text{twosCrnd}(x) := \text{floor}\left[\left[x + 2^{-(n+1)}\right] \cdot 2^{n}\right] \cdot 2^{-n} \tag{4-4}$$

$$\text{smrnd}(x) := \begin{cases} \left[\text{floor}\left[\left[x + 2^{(-(n+1))}\right] \cdot 2^{n}\right] \cdot 2^{-n}\right] & \text{if } x \geq 0 \\ \left[\text{ceil}\left[\left[x - 2^{(-(n+1))}\right] \cdot 2^{n}\right] \cdot 2^{-n}\right] & \text{otherwise} \end{cases} \tag{4-5}$$

Rounding sign magnitude coded data produces a symmetrical quantisation error for positive and negative data. Therefore no statistical bias (dc offset) is introduced into the

quantisation error. However for twos-complement rounding there is one data instance (exactly one half the LSB residual magnitude) that generates a different rounding result for positive and negative data, and thus introduces a bias in the quantisation error. This bias, caused by one data instance in a population of 2^r (where r is the number of residual bits) is extremely small and is not considered important in this work. Twos-complement convergent rounding (Motorola, 1999) is a technique which eliminates this bias, and should be used where any small bias is problematic.

Equations (4-2), (4-3), (4-4) and (4-5) form the basis of the Mathcad truncation and rounding quantiser functions for sign magnitude and twos-complement coded data. The resulting quantisation errors using these rounding and truncation functions, quantising to 23 fractional bits, for twos-complement and sign magnitude coding are shown in Figure 4-1 and Figure 4-2. Note the sign magnitude coded truncation produces the largest noise variance and the bipolar error is correlated with the input, Figure 4-1. The twos-complement coded data truncation produces a bias (dc offset) in the quantisation error, shown in Figure 4-2. Figures 4-3 and 4-4 show the spectral energy of the quantisation error for truncated sign magnitude and twos-complement data coding, respectively. Figure 4-3 shows the large odd order harmonic distortion components and the distortion at the fundamental frequency (999.023 Hz), attributable to the correlation with the sinusoidal input. The dc energy in the twos-complement truncation error is visible in Figure 4-4.

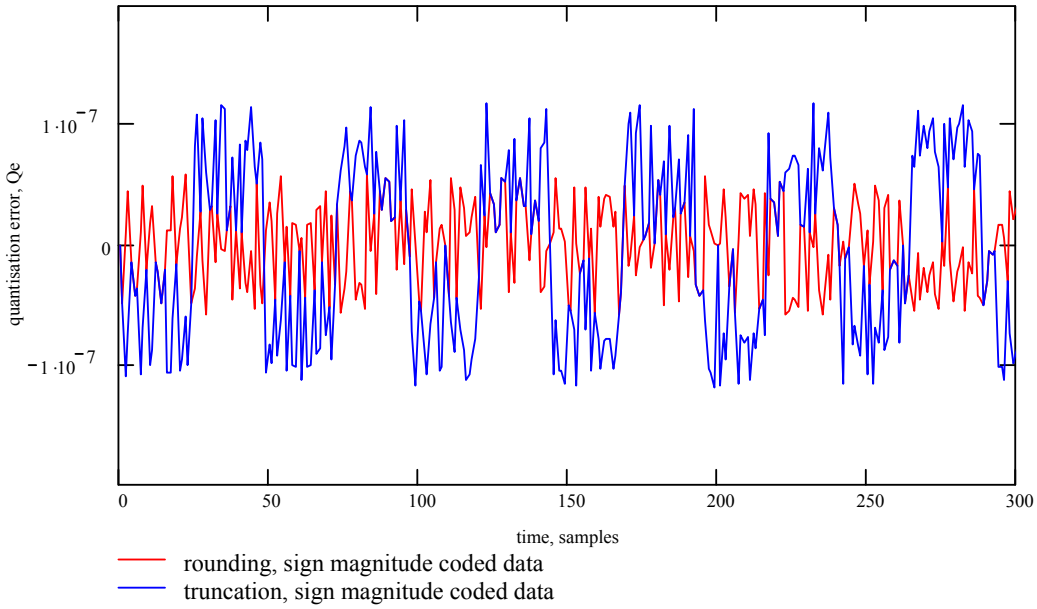

Figure 4-1 Quantisation error from truncation and rounding of sign magnitude fixed point data to 23 fractional bits, (sinusoidal input, 999.023 Hz, 0dBFS)

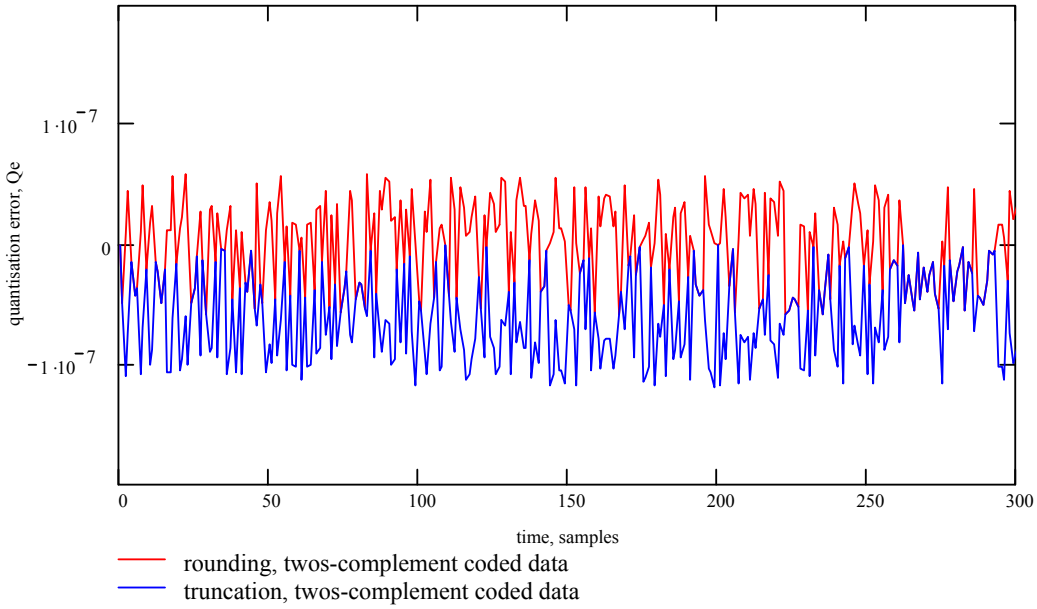

Figure 4-2 Quantisation error from truncation and rounding of twos-complement fixed point data to 23 fractional bits, (sinusoidal input, 999.023 Hz, 0dBFS)

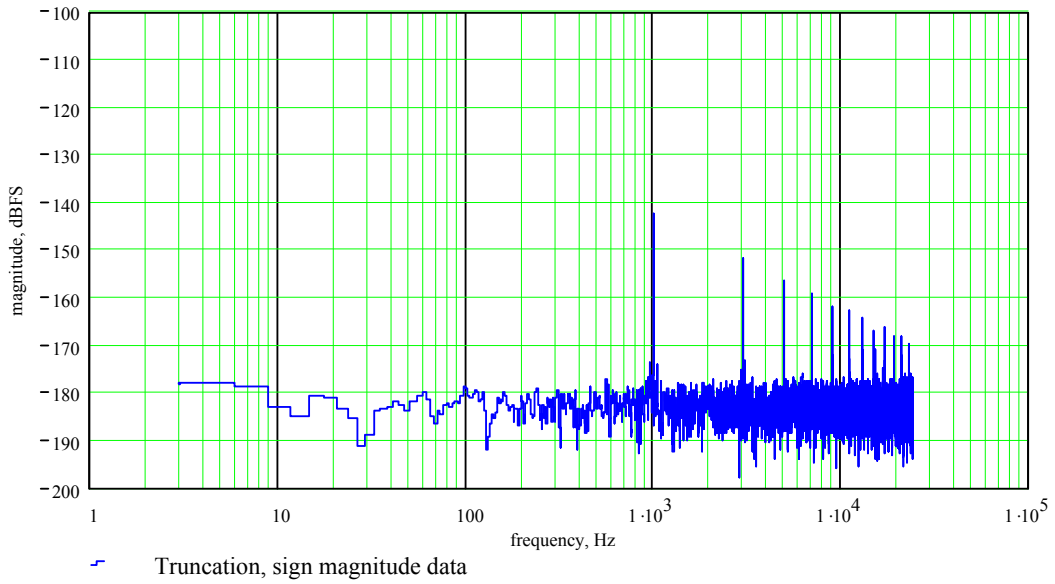

Figure 4-3 Spectral energy of quantisation error from truncation of sign magnitude fixed point data to 23 fractional bits, (sinusoidal input, 999.023 Hz, 0dBFS)

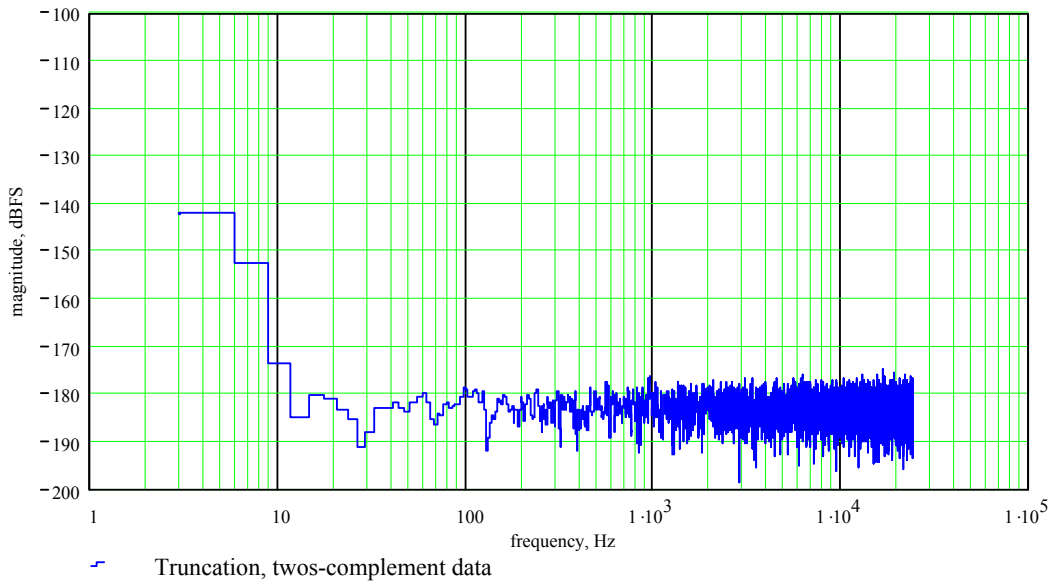

Figure 4-4 Spectral energy of quantisation error from truncation of twos-complement fixed point data to 23 fractional bits, (sinusoidal input, 999.023 Hz, 0dBFS)

4.3 Fixed point model

Figure 4-5 shows a finite wordlength fixed point multiply-accumulate model. The multiplier and multiplicand are fractional n bit words. These produce a fractional product of 2n-1 bits. If the wordlength used to represent the product is less than (2n-1) bits then a

quantisation error is introduced. If the product is stored as a single precision number or used as a source for a further single precision multiplication then the product is quantised to n bits. If the product is used in an accumulation operation it remains at 2n-1 fractional bits. The entire accumulation process remains at 2n-1 fractional bits (ignoring accumulator headroom bits). The accumulator output is quantised to n fractional bits for single precision storage or for single precision multiplication. Equation (4-6) performs the multiplication of two 'n' bit fractional numbers, x and y, assuming fixed point arithmetic. The product is stored as n bits, shown as point p in Figure 4-5. The multiplication input variables x and y are quantised to n bits by a quantiser function - denoted Q_m in Equation (4-6). The product is then quantised back to n bits by the same quantiser function, Q_m. The finite wordlength fixed point model ignores headroom bits, which are typically supplied in the accumulator. Accumulator integer bits are ignored, since all fixed point implementations considered in this work rely on modulo wrap-round or do not exceed the fractional boundaries of the arithmetic.

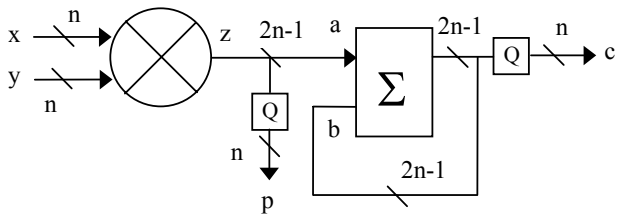

Figure 4-5 Finite wordlength fixed point multiply-accumulate model

$$Q_m [Q_m(x) . Q_m(y)] \qquad (4\text{-}6)$$

4.4 Floating point model

Equation (4-7) describes the floating point representation of a number, A. The signed mantissa, A(man), is scaled by two raised to the power of the signed exponent of A,

A(exp). The product, C, of two floating point numbers A and B is shown in Equation (4-8). Floating point accumulation is described by Expressions (4-9) and (4-10).

$$A = A(man) \cdot 2^{A(exp)} \tag{4-7}$$

$$C = A \cdot B = A(man) \cdot B(man) \cdot 2^{A(exp) + B(exp)} \tag{4-8}$$

$$\text{if } A(exp) \geq B(exp)$$
$$\text{then } C = A \pm B = [A(man) \pm (B(man) \cdot 2^{-(A(exp) - B(exp))})] \cdot 2^{A(exp)} \tag{4-9}$$

$$\text{if } A(exp) < B(exp)$$
$$\text{then } C = A \pm B = [B(man) \pm (A(man) \cdot 2^{-(B(exp) - A(exp))})] \cdot 2^{B(exp)} \tag{4-10}$$

A floating point number is typically said to be normalised if the mantissa most significant fractional bit is a one. This results in an effective magnitude of between one and two, since floating point implementations use a hidden bit, which is conceptually placed to the right of the binary point (Analog Devices, 1997; Texas Instruments, 1992). Normalisation involves shifting the mantissa left until the leading bit is the MSB (the mantissa magnitude is between one and two). The number of mantissa shifts is subtracted from the exponent to compensate for the mantissa shifting. Numerical data in Mathcad is manipulated in decimal representation, not with a separate mantissa and exponent. In order to perform floating point finite wordlength arithmetic, functions were developed to extract a normalised mantissa and exponent from the nominal Mathcad decimal representation. Therefore the normalisation principle is used to extract the exponent from a decimal

number. Equation (4-11) is a Mathcad function 'flnorm' that extracts the exponent from a decimal number.

```
flnorm(x) := | a ← |x|
             | return 0 if a = 0
             | i ← 0
             | while a < 1
             |     a ← a·2
             |     i ← i − 1
             |     return i if a ≥ 1
             | while a ≥ 2
             |     a ← a·0.5
             |     i ← i + 1
             |     return i if a < 2
             | return i
```

(4-11)

Once the exponent, exp, is known the normalised mantissa can be realised by simply scaling the decimal representation by 2^{-exp}. It is more efficient in Mathcad to pass variables between arithmetic operations in decimal representation. Therefore mantissa and exponents are extracted from decimal data when required for arithmetic operations. Floating point quantisers can be completely emulated in decimal form, increasing efficiency. Fractional quantisation of floating point data is governed by the finite wordlength of a normalised mantissa. Numerically equivalent quantisation can be performed on a decimal number if an 'equivalent quantisation boundary' is determined. The 'equivalent quantisation boundary' of a decimal number is the number of fractional bits in the target mantissa wordlength summed with the extracted exponent, assuming a normalised mantissa. An example floating point quantiser function, performing twos-complement truncation on a decimal number, is given in Equation (4-12). The function extracts the exponent and adds this to the specified fractional boundary (target mantissa wordlength), 'n'. This is the equivalent quantisation boundary of the decimal number and

produces the same quantisation error in the data as for the specified floating point format. The *floor* function is then used to perform twos-complement truncation at the equivalent quantisation boundary. A sign magnitude rounding quantiser could be implemented by replacing the *floor* function with Expression (4-5). The complete set of floating point quantisers is given in Appendix A.

$$
\begin{aligned}
&\text{fl2ctrc}(x, n) := \\
&\quad a \leftarrow |x| \\
&\quad i \leftarrow 0 \\
&\quad \text{return } x \text{ if } x = 0 \\
&\quad \text{while } a < 1 \\
&\quad\quad a \leftarrow a \cdot 2 \\
&\quad\quad i \leftarrow i - 1 \\
&\quad \text{while } a \geq 2 \\
&\quad\quad a \leftarrow a \cdot 0.5 \\
&\quad\quad i \leftarrow i + 1 \\
&\quad \text{return } \text{floor}\left[x \cdot 2^{(n-i)}\right] \cdot 2^{-(n-i)}
\end{aligned}
\tag{4-12}
$$

Resulting quantisation errors operating on a sinusoid signal at 999.023 Hz, using the twos-complement and sign magnitude rounding and truncation quantiser functions are shown in Figure 4-6 and Figure 4-7. Figure 4-8 shows the spectral energy of the quantisation error for sign magnitude truncated data. The strong correlation with the input signal is evident as distortion at the fundamental frequency (999.023 Hz). Figure 4-9 shows the spectral energy of twos-complement truncated data. Even order distortion (second, forth and sixth) components are noticeable.

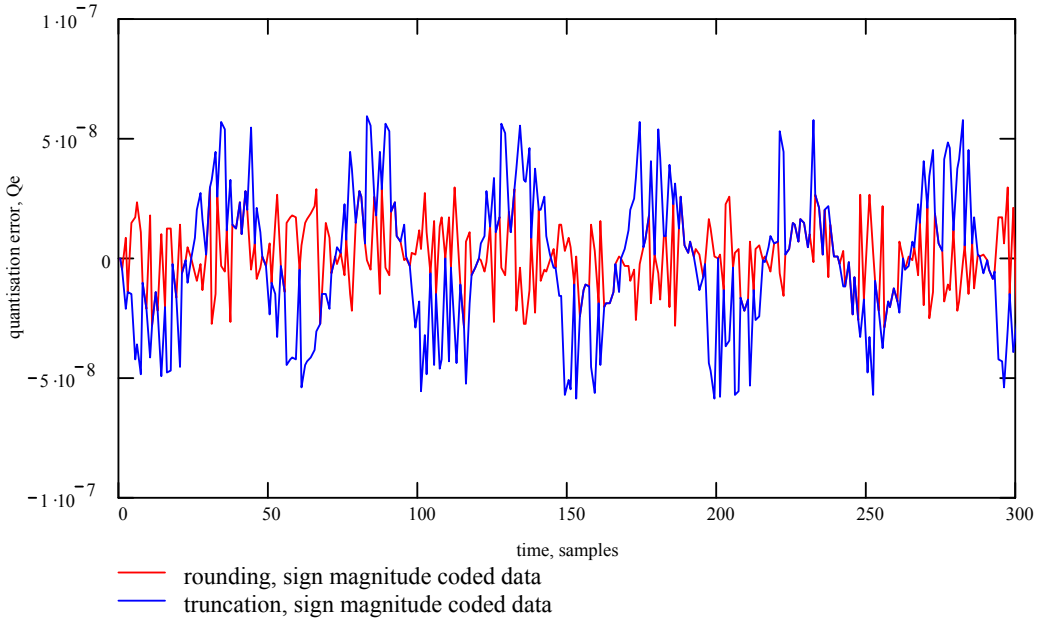

Figure 4-6 Quantisation error from truncation of sign magnitude floating point data to 23 fractional bits in the mantissa, (sinusoidal input, 999.023 Hz, 0dBFS)

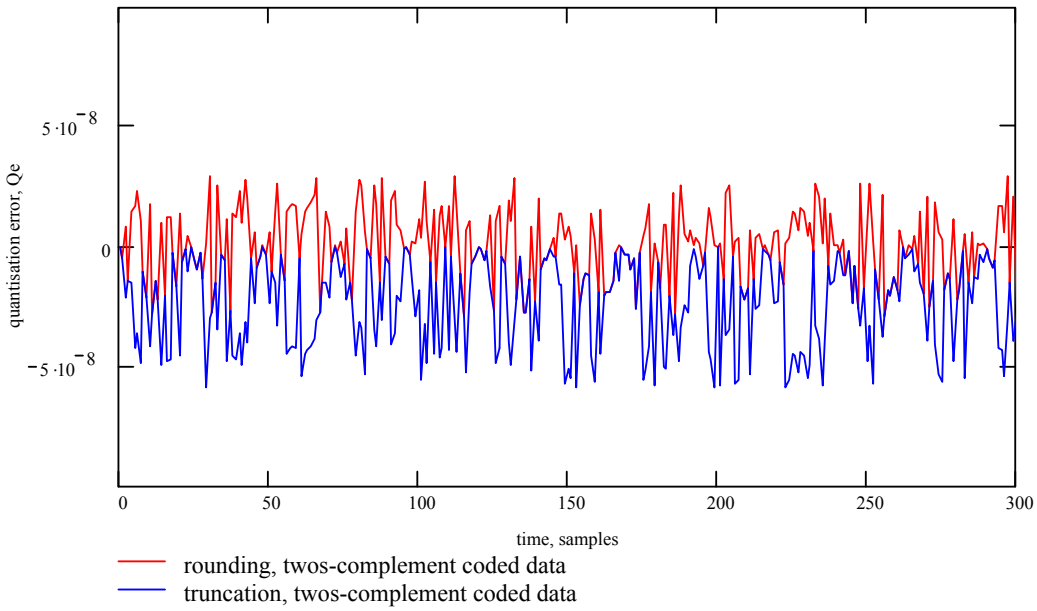

Figure 4-7 Quantisation error from truncation of twos-complement floating point data to 23 fractional bits, in the mantissa, (sinusoidal input, 999.023 Hz, 0dBFS)

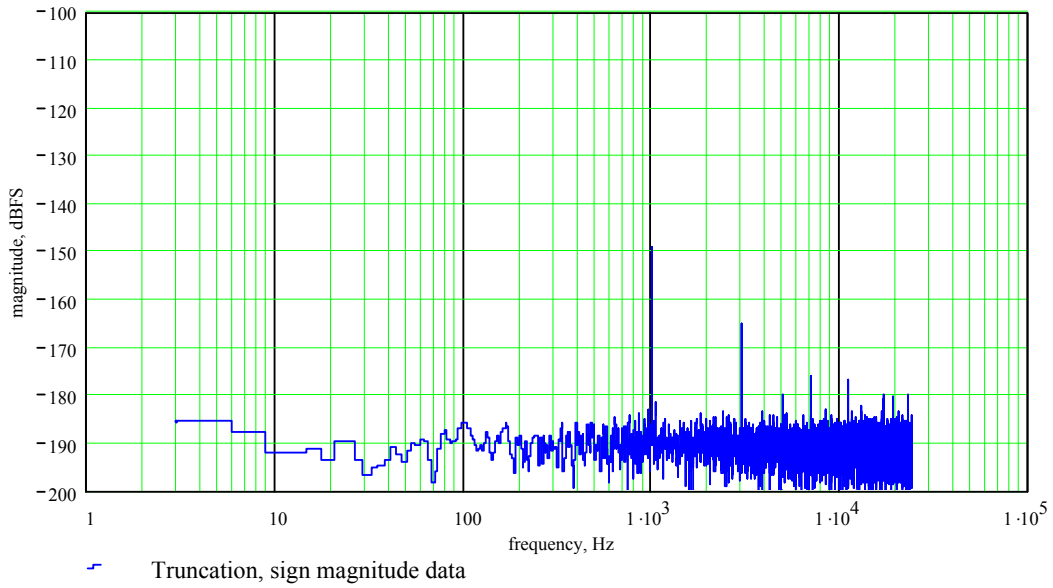

Figure 4-8 Spectral energy of quantisation error from truncation of sign magnitude floating point data to 23 fractional bits in the mantissa, (sinusoidal input, 999.023 Hz, 0dBFS)

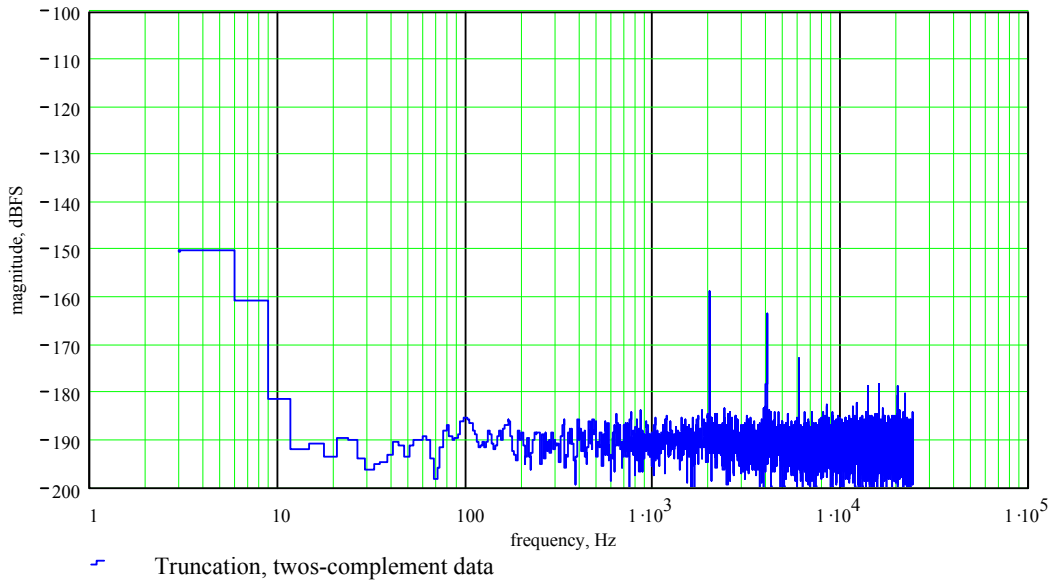

Figure 4-9 Spectral energy of quantisation error from truncation of twos-complement floating point data to 23 fractional bits in the mantissa, (sinusoidal input, 999.023 Hz, 0dBFS)

4.4.1 Floating point multiplication model

Floating point multiplication is defined in Expression (4-8). The exponents are summed and the mantissas are directly multiplied together. The mantissas do not strictly need to be

normalised, although it is very likely the data is already normalised. The product may need re-normalising and is as a matter of course. This may incur quantisation. An example floating point multiplier is given in Expression (4-13). The flnorm function, Expression (4-11), is used to extract the two exponents of the numbers, a and b. The normalised mantissas are derived from their exponents. The mantissas are then multiplied together and quantised at the specified product wordlength (floatextprec). This is equivalent to any potential re-normalisation quantisation. The product mantissa is then scaled by the exponent, forming a decimal number. Floating point multiplier functions for sign magnitude and twos-complement rounding and truncation are given in Appendix A.

$$\begin{aligned}
&\text{Multsmtrc}(a,b) := \\
&\quad aexp \leftarrow flnorm(a) \\
&\quad bexp \leftarrow flnorm(b) \\
&\quad cexp \leftarrow aexp + bexp \\
&\quad amant \leftarrow a \cdot 2^{-aexp} \\
&\quad bmant \leftarrow b \cdot 2^{-bexp} \\
&\quad cmant \leftarrow amant \cdot bmant \\
&\quad \text{return } flsmtrc(cmant, floatextprec) \cdot 2^{cexp}
\end{aligned} \tag{4-13}$$

4.4.2 Floating point addition model

Floating point accumulation is defined in Expressions (4-9) and (4-10). Mathcad Expression (4-14) is an example of a floating point addition function, using sign magnitude binary coding and using rounding as a quantiser. The exponents of the two source numbers to the accumulator are calculated. The largest exponent is used as the result exponent, termed 'cexp' in Expression (4-14). The mantissas for the two data sources are generated and quantised at the specified extended precision. The source data with the smaller exponent, has its mantissa aligned to compensate for the result exponent. Alignment of the mantissa (shift right) should potentially produce quantisation, however

all floating point implementations eliminate this quantisation source through the use of two extra precision bits, typically referred to as the guard and 'sticky' bits. The two mantissas are then summed and the accumulator result is re-normalised (quantised) by a relevant quantisation function at the specified extended precision. The result is then scaled by the result exponent producing a decimal number. The other floating point accumulator operations are given in Appendix A.

$$\begin{aligned}
\text{Addsmrnd}(a,b) := \;& \text{aexp} \leftarrow \text{flnorm}(a) \\
& \text{bexp} \leftarrow \text{flnorm}(b) \\
& \text{cexp} \leftarrow \text{aexp} \quad \text{if} \quad \text{aexp} > \text{bexp} \\
& \text{cexp} \leftarrow \text{bexp} \quad \text{otherwise} \\
& \text{amant} \leftarrow \text{flsmrnd}\left(a \cdot 2^{-\text{aexp}}, \text{floatextprec}\right) \\
& \text{bmant} \leftarrow \text{flsmrnd}\left(b \cdot 2^{-\text{bexp}}, \text{floatextprec}\right) \\
& \text{result} \leftarrow \left[\text{bmant} + \text{amant} \cdot 2^{-(\text{bexp}-\text{aexp})}\right] \quad \text{if} \quad \text{aexp} < \text{bexp} \\
& \text{result} \leftarrow \left[\text{amant} + \text{bmant} \cdot 2^{-(\text{aexp}-\text{bexp})}\right] \quad \text{otherwise} \\
& \text{out} \leftarrow \text{flsmrnd}(\text{result}, \text{floatextprec}) \\
& \text{return} \quad \text{out} \cdot 2^{\text{cexp}}
\end{aligned} \tag{4-14}$$

4.5 Testing finite wordlength arithmetic operations

The accuracy and operation of the finite wordlength arithmetic quantisation functions was compared with arithmetic operations implemented on an industry standard DSP platform (Analog Devices, 1997). The DSP platform implements IEEE 32 bit (40 bit extended precision) floating point and 32 bit twos-complement fixed point arithmetic. Mathcad arithmetic and quantisation functions for sign magnitude floating point and twos-complement fixed point were compared, at bit level, with arithmetic on the DSP platform. The operations implemented on the DSP platform were chosen to be 32 bit single precision (24 bit mantissa plus 8 bit exponent). The floating point arithmetic operations in the Mathcad environment were configured to 23 fractional bits. The test data chosen

therefore reflected this and exercised the wordlength boundary, 2^{-23} and 2^{-24}. The test data used, Mathcad test scripts, DSP program and resulting test data is shown in Appendix A.

4.6 Summary

Quantisation behaviour of sign magnitude and twos-complement binary coded data has been examined. This led to the specification of quantisation functions for truncation and rounding under twos-complement and sign magnitude coded data. Floating point arithmetic operations such as addition, multiplication and normalisation were analysed. Mathcad functions are developed that emulate the quantisation effects of finite wordlength fixed and floating point arithmetic operations. The emulated finite wordlength arithmetic is found to be 'bit exact' in comparison with a IEEE 32 bit floating point and 32 bit twos-complement fixed point DSP platform.

These functions will facilitate the implementation of various filter topologies under various finite wordlength constraints in the Mathcad environment. The emulation of finite wordlength filter topologies in the time domain provides a method of topology comparison using fixed and floating point arithmetic with different forms of input stimuli. This work is described in Chapter 5. These functions will also facilitate the implementation of various coefficient interpolators under finite wordlength constraints. Coefficient interpolator behaviour under finite wordlength constraints and the associated effects on the audio signal is investigated in Chapter 6.

5 Topology behaviour in finite wordlength arithmetic

5.1 Introduction

The behaviour of static frequency response filter topologies under various forms of finite wordlength arithmetic are investigated in this chapter. Using finite wordlength arithmetic functions, described in Chapter 4, a finite wordlength filter topology emulation environment is developed in Mathcad. Filter topologies are implemented in the discrete time domain, under fixed and floating point arithmetic with varying wordlengths. Various filter frequency responses, input excitations and sampling rates are used to explore the behaviour of filter topologies under finite wordlength constraints.

5.2 Filter test environment

The finite wordlength filtering system, Figure 5-1, is implemented in the Mathcad environment (Mathsoft, 1998). Various time domain filter topologies are implemented with finite wordlength arithmetic functions. The tests conducted in this environment use a range of different filter input stimuli. Each filter test uses a static filter frequency response with a static set of pre-calculated coefficients. Furthermore, various signal processing sampling frequencies are used in the investigation (48, 96 and 192 kHz). The time domain filters operate for a determined time period or number of samples (typically two seconds). This provides sufficient filter output samples to conduct comparative analysis through FFT spectrum analysis, time domain plots and RMS noise benchmarking.

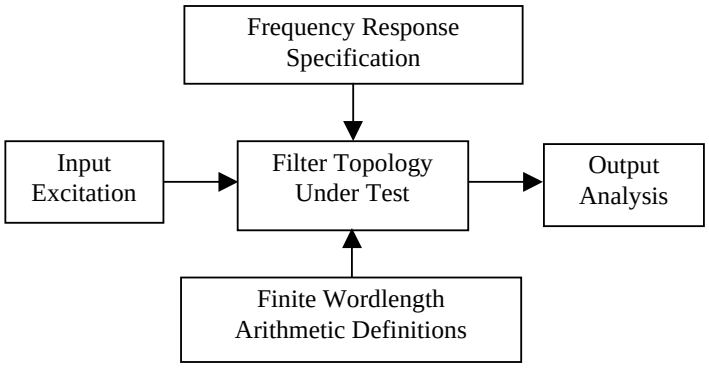

Figure 5-1 Time domain filter topology analysis system

5.2.1 Finite wordlength filter topologies

The topologies considered in this work are introduced in Chapter 2, Section 2.5. All topologies were implemented with floating point arithmetic. Topologies that are immune from accumulator overflow, through the use of twos-complement modulo wrap-round, were also implemented in fixed point arithmetic. As discussed in Chapter 2 Sections 2.4 and 2.5, topologies which rely on scaling to prevent overflow were not implemented in fixed point arithmetic.

For fixed point implementation the basic arithmetic operators for multiply and addition (accumulation) can be used. The fixed point format is assumed to be fractional twos-complement, the result of each arithmetic operation can simply be quantised to the resolution specified by a suitable rounding or truncation function, as discussed in Chapter 4, Sections 4.2 and 4.3 . Furthermore a quantiser has to be applied at every state variable storage point in the topology. Double precision is simply implemented by increasing the resolution of the state variable quantiser to the relevant resolution. Each of the topologies implemented are shown in Chapter 2, Section 2.5. The quantisation points in each topology are shown, assuming single precision implementation.

Floating point arithmetic cannot rely on the basic operators, since finite wordlength quantisation is dependent on the mantissa and exponent of the data. Therefore specific

arithmetic functions for multiply and addition are used in replacement for the basic operators, as discussed in Chapter 4 (Section 4.4). For example, consider the following arithmetic, a.b + c.d + e.f , would be implemented as,

Add(Add(Mlt(a,b) , Mlt(c,d)) , Mlt(e,f)).

The functions 'Add' and 'Mlt' perform addition and multiplication floating point arithmetic operations and apply quantisation for a specified wordlength and binary coding technique. These functions are developed in Chapter 4 (Section 4.4). Mathcad scripts of the discrete-time implementation of all of the filter topologies used in the tests are given in Appendix B.

5.2.2 Frequency Response Specification

In Chapter 2, Section 2.2 an introduction to audio equaliser filter types and associated filter frequency responses relevant to this work is given. Filters with high gain in the pole transfer function and minimal attenuation in the zero transfer function, typically produce high noise gain. These filters are generally low frequency filters with high Q factors and are critical in the examination of finite wordlength effects in filter topologies.

Unfortunately no single frequency response illustrates all filter quantisation noise issues. Thus, in this work specific frequency responses are chosen to highlight particular noise issues. However, the chosen filter parameter settings and associated frequency response are practical and could be used in mixing system equalisation.

The High Pass Filter (HPF) is the most likely filter type to be used with an extremely low tuned frequency, for example 5 to 10 Hz. However, HPF Q factor settings are typically low - in the region of 0.7071 (Butterworth). A HPF tuned to 5 Hz, operating at a sampling frequency of 48 kHz, produces a high gain in the pole transfer function (a maximum of 115 dB). However, low frequency tuned HPF produce high attenuation in the zero

transfer function, especially at dc (zero Hz). A notch filter is tuned to a minimum frequency of 20 Hz. Using a –3 dB bandwidth of 7.5 Hz, at a sampling frequency of 48 kHz, produces a maximum pole transfer function gain of 112 dB, Figure 5-2. The notch filter zero transfer function does not produce as much broad band attenuation as the HPF and its overall response is zero dB at dc. The bell filter is tuned to a minimum of 30 Hz. Bell filter Q settings are rarely as high as notch filter Q settings, especially for boost gain settings. A Q setting of three is moderately high for boost scenarios. A bell filter providing a gain of 18 dB at the tuned frequency of 30 Hz, with a Q factor setting of three, produces a maximum gain of 106 dB in its pole transfer function.

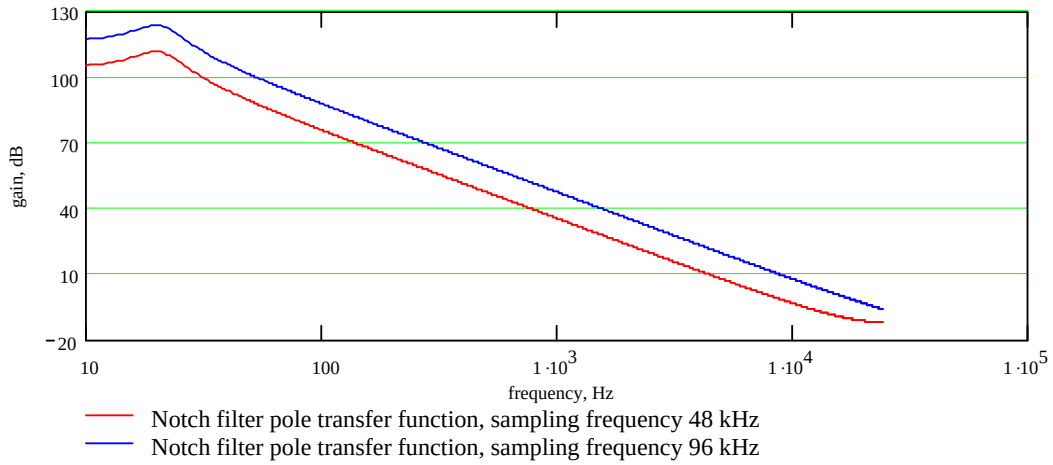

Figure 5-2 Pole transfer functions for 20 Hz notch filter (-3 dB bandwidth of 7.5 Hz).

Filters can also produce high amounts of gain in the pole transfer function at high frequencies (close to the Nyquist frequency). A notch filter tuned to 19 kHz, with a bandwidth of 7.5 Hz (which is not a practical scenario) produces 64 dB maximum gain in its pole transfer function. Shelving filters do not produce high gain in their pole transfer functions and are not considered for noise behaviour investigations.

Four filter types are used to obtain test frequency responses, in the work described in this chapter. Three bell filter responses are used, one boosting at low frequency (30 Hz), one providing low frequency attenuation (20 Hz) and a high frequency bell filter boosting at

19 kHz. Narrow bandwidth notch filters (-3 dB bandwidth of 7.5 Hz) tuned to various frequencies are used. For example a low frequency 20 Hz notch and various high frequency notch filters are used in the zero input tests.

- '20 Hz bell filter cut case', tuned to 20 Hz, 18 dB attenuation, Q factor of 8.65.
- '30 Hz bell filter boost case', tuned to 30 Hz, 18 dB gain, Q factor of 3.
- '19 kHz bell filter boost case', tuned to 16 kHz, 18 dB gain, Q factor of 3
- '20 Hz notch filter', -3dB bandwidth of 7.5 Hz.
- '15, 16, 19 kHz notch filters', -3dB bandwidth of 7.5 Hz.

5.2.3 Input Excitation

Single sinusoids, twin-tone signals and bursts of white noise are used as input stimuli for the various time domain filter tests. Filters can be stimulated with high level white noise, approximating audio programme. Noise analysis relies on an ideal filter output, under the same stimulus, being subtracted from the output of the filter under test. However, white noise stimulus will not provide an efficient way of examining harmonic distortion. A sinusoid, at a carefully selected frequency, is a good input stimulus for noise and harmonic distortion analysis. Sinusoidal frequencies that are sub-multiples of the sampling frequency produce highly correlated quantisation noise. This can be undesirable if the test is attempting to examine broad band quantisation noise effects. For example, highly correlated quantisation errors are generated from one and four kHz tones at a sampling frequency of 48 kHz (Clark et al, 1995). Fast fourier transforms (FFT's) are used to provide signal spectrum analysis. Sinusoidal input frequencies were bin centred, reducing FFT spectral leakage, (Harris, 1978). Single sinusoidal input stimulus used the bin centred frequency of 999.0234 Hz. This is the centre frequency for bin 341, using a 16384 point FFT at a sampling frequency of 48 kHz. At the sampling frequency of 96 kHz, a 32768 point FFT is used. The same bin centred frequency of 999.0234 Hz is used as an

input stimulus (bin 341). Despite this bin centred frequency being a sub-multiple of the sampling frequencies, it does not produce a harmonically rich quantisation noise. Sinusoids were chosen around the 1 kHz region, since it is not in the same spectral region as the tuned frequencies of the example filters, chosen in this work. This reduces the possibility of the input sinusoid masking any interesting noise products. Zero input tests are also useful in the examination of limit cycle behaviour in filter structures. However the filter must be excited prior to the zero input period. This is performed using a burst of white noise followed by samples of zero amplitude. Low level signals provide a method of examining the noise characteristics of topologies under low level signal operation. This is performed using a sinusoid at –90 dBFS. Inter-modulation distortion (IMD) tests are used to evaluate high level non-linearity in systems (Metzler, 1993). IMD tests, using a twin-tone stimulus, are used to evaluate high signal level non-linearity of the filter topologies under test.

5.2.4 Output Analysis

Spectral analysis is performed through FFT's providing the facility to examine distortion and noise products in the filter topology output. Windowing functions can be applied in the time domain, prior to the FFT, to reduce spectral leakage, (Harris, 1978). The Blackman Harris 4 (BH4) windowing function is widely used in audio applications (Harris, 1978). The FFT frequency bins and resulting bin width, is largely responsible for the overall spectral resolution. At a sampling frequency of 48 kHz, 16384 sample FFT's are used, producing a frequency bin width of 2.92 Hz. At the higher sampling frequencies of 96 and 192 kHz, 32768 sample FFT's are used. This maintains spectral bin width to 2.92 Hz, for a 96 kHz sampling frequency. At 192 kHz 32768 point FFT were used due to extremely long emulation periods. Four 16384 (or 32768) sample FFT measurements

were averaged to reduce the effects of noise variation, which was especially noticeable at low frequencies.

Root Mean Square (RMS) total harmonic distortion plus noise (THD+N) figures are used for broad band (0 to Fs/2 Hz) residual noise/distortion benchmarking. A simple method of deriving an RMS residual noise/distortion figure from the output of the filter under test is to subtract an 'ideal' filter output from the filter under test, revealing the residual noise generated by the filter under test. Since the 'ideal' filter has no finite wordlength limitations, the residual noise components are negligible. It is essential that the ideal filter and filter under test produce the same frequency response (use the same coefficients, quantised or unquantised). Equation 5-1 is a time domain function calculating the broad band RMS THD+N figure of a filter under test (FUT) within the time period defined by 'FirstSample' and 'LastSample'. Note 'IdealFilter' is the output of the filter topology using no finite wordlength functions. However the 'ideal filter' is subject to the finite wordlength limitations of Mathcad.

$$\mathrm{TimeThD} := 20\cdot\log\left[\left|\sqrt{\frac{\sum_{k=\mathrm{FirstSample}}^{\mathrm{LastSample}}\left(\mathrm{FUT}_k - \mathrm{IdealFilter}_k\right)^2}{\mathrm{LastSample} - \mathrm{FirstSample}}}\right|\right] \tag{5-1}$$

5.2.5 Finite wordlength performance of the emulator

This section of work aims to investigate the effects of controlled finite wordlength functions used to implement filter topologies. Therefore the intrinsic finite wordlength limitations of the emulation environment, Mathcad, must be examined. Mathcad is implemented using the 'intel double precision' data format, (Mathsoft, 1998). This provides a 64 bit floating point data word comprising of a 53 bit significand (unsigned

fractional bits). This finite wordlength determines the residual noise floor and dynamic range of the filter emulation environment. Figure 5-3 shows the residual noise of the Mathcad system using a sinusoidal input, 993.023 Hz at 0 dBFS (dB full-scale), processed by the 16384 point FFT (using a BH4 window), averaging four FFT measurements, sampling at 48 kHz. The largest distortion component is less than -241 dBFS at 11 kHz. This suggests a dynamic range, in the region of 240 dB.

The DF1 topology was implemented using no finite wordlength arithmetic functions, operating at a sampling frequency of 48 kHz. Figure 5-3 shows the residual distortion products for the DF1, for the 20 Hz notch filter, using a single sinusoidal stimulus of 999.023 Hz, at 0 dBFS. This measurement highlights a problem in noise measurement of emulated recursive filters. At the start of the emulation, the filter's output contains excessive 'ringing' energy in the pole frequency region. This decays as the filter approaches steady-state operation. It is important that this ringing energy is not confused or allowed to mask the actual noise and distortion products of interest. It is therefore important to use a filter relaxation period before any filter output analysis is conducted. This period should be of a length in time to ensure that any ringing has decayed to a level below the residual noise of the emulation environment. Figure 5-3 shows the output spectrum of the DF1 using the 20 Hz notch filter for various relaxation periods (26000, 41000 and 90000 samples). The FFT measurement made after a 26000 sample relaxation period shows considerable ringing energy. It was found that for the notch and bell (cut) filter examples FFT measurements taken after a 41000 sample relaxation period produced negligible ringing above the residual noise floor of the ideal filter. Figure 5-3 shows the output spectrum after a relaxation period of 90000 samples. The residual noise products are not noticeably different of that from the 41000 sample starting point. Therefore a relaxation period of 41000 samples, prior to any output analysis was adopted for sampling frequencies of 48 kHz. For higher sampling rates, the pole ringing is considerably worse.

At sampling frequencies of 96 kHz a relaxation period of 82000 samples was required to reduce the pole ringing to below the residual noise floor.

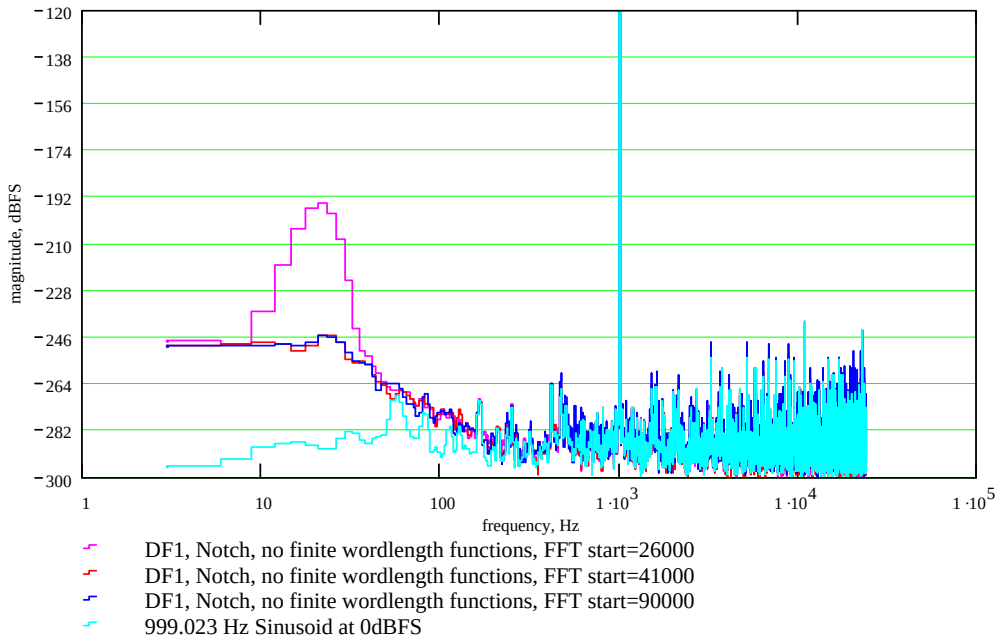

Figure 5-3 Ideal filter noise measurements from Mathcad emulation environment at a sampling frequency of 48 kHz.

5.3 Comparison of real and emulated systems

The DF1 implemented in the finite wordlength emulation environment was compared to real measurements made from DF1 implementations on a DSP platform. The DSP platform is based on the Analog Devices SHARC processor ADSP-21061 (Analog Devices, 1997) and performs IEEE 32 bit floating point arithmetic and twos-complement 32 bit fixed point arithmetic. An '*AES 3*' digital audio interface (Audio Engineering Society, 1992), (Cirrus Logic, 1999) provided an interface between the DSP platform and the audio measurement system (Audio Precision System Two Cascade). These tests were conducted at a sampling frequency of 48 kHz. The entire audio signal path remained in the digital domain. The AES/EBU digital input and output interface is a fixed point 24 bit word format. For this reason the input and output data in the emulation used a 24 bit (sign

bit and 23 fractional bits) truncation quantiser. Triangular dither was applied prior to the 24 bit quantiser, as the digital signal generator in the Audio Precision measurement system also dithered its source prior to quantising the data to the 24 bit AES/EBU format. The filter response used for the emulator DSP platform comparisons was the 20 Hz bell cut example (20 Hz, Gain of –18dB, Q factor of 8.65). The SHARC assembler codes and respective Mathcad scripts for the emulated filters for the DF1 floating and fixed point implementations are given in Appendix C.

5.3.1 Fixed point Direct Form 1

The fixed point measurements made with the DSP platform and the Mathcad emulation environment are shown in Figure 5-4 and Figure 5-5, respectively. Two DF1 structures were implemented, one using rounding and one using truncation quantisation. Both the real and emulation measurements used a single sinusoidal input stimulus of 999.023 Hz at –10dBFS. The fixed point implementation also uses a coefficient scaling factor of two, requiring an arithmetic shift left on the output of the accumulator, as shown in Figure 2.5 . FFT measurements of the DSP platform filters were made with the Audio Precision system using a 16384 point FFT, BH4 window, taking sixteen averages. FFT measurements of the emulated filters use four averaged 16384 point FFT's due to the limitations in emulation sample lengths. Comparison of Figure 5-4 and Figure 5-5 shows that the measurements taken for the 32 bit fixed point DF1 emulation and that from the real 32 bit fixed point DSP platform are similar.

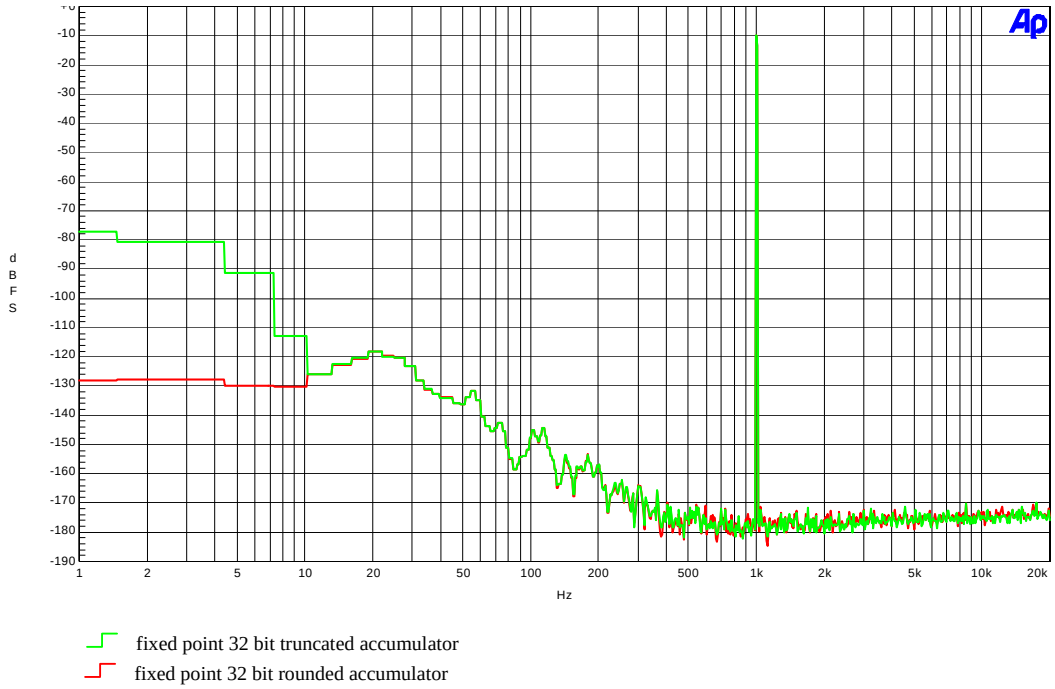

Figure 5-4 DF1output spectrum from the SHARC DSP platform, 32 bit fixed point. Bell filter (20 Hz, –18dB, Q=8.65), sine input, 999.023 Hz, –10 dBFS, 16 FFT averages 16384 bins, BH4 window.

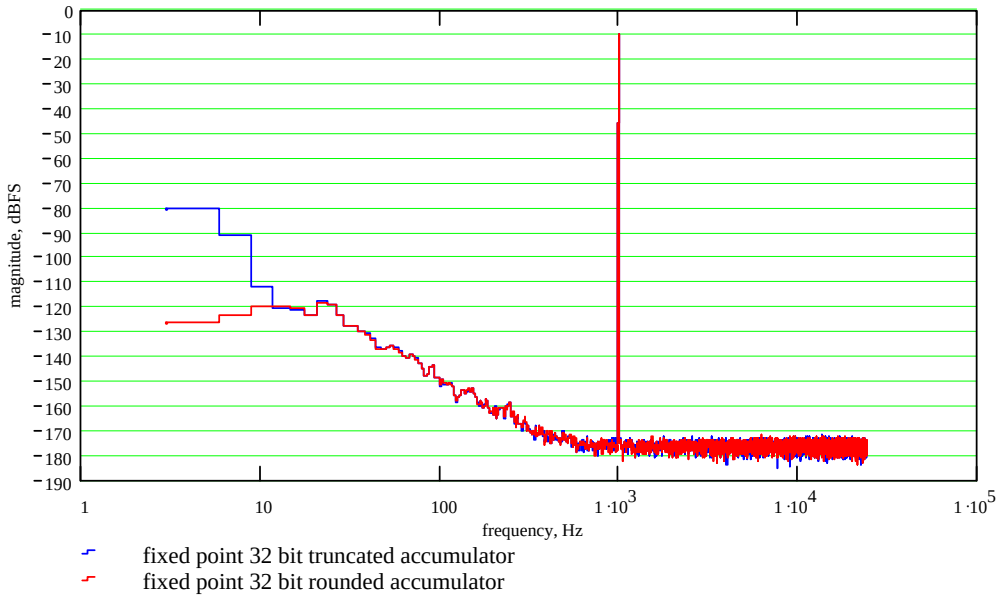

Figure 5-5 DF1output spectrum from Mathcad emulation , 32 bit fixed point. Bell filter (20 Hz, –18dB, Q=8.65), sine input, 999.023 Hz, –10 dBFS, 4 FFT averages 16384 bins, BH4 window.

5.3.2 Floating point Direct Form 1

The DSP platform implements IEEE floating point arithmetic (Analog Devices, 1997). The IEEE floating point arithmetic emulation uses 32 bit sign magnitude arithmetic functions (23 fractional bits) with a 40 bit extended format (31 fractional bits). Two DF1 implementations were developed one using rounding and one using truncation quantisation. Three low level sinusoids were used as test stimuli, -80, -90 and -100 dBFS, all at a frequency of 999.023 Hz.

Figure 5-6 shows the output spectrum of the DF1 DSP platform, 32 bit (single precision storage elements), using truncation quantisation. Figure 5-7 shows the equivalent output spectrum for the emulated DF1. Figure 5-8 shows the output spectrum of the DF1 DSP platform, 32 bit (single precision storage elements), using rounding quantisation. This can be compared to the emulated floating point DF1 using rounding in Figure 5-9. The emulated filter output spectra for truncation and rounding are both similar to the DSP platform output spectra. However it is evident that there is some variation in the low frequency noise products. This was examined further, by comparing single FFT measurements across time, made with the Audio Precision measurement system. Noticeable differences were found in the LF noise products for each FFT measurement. For the real DSP platform measurements sixteen FFT averages could simply be employed to reduce noise variance. However increasing the FFT number of averages in the emulation, led to unfeasible emulation time periods

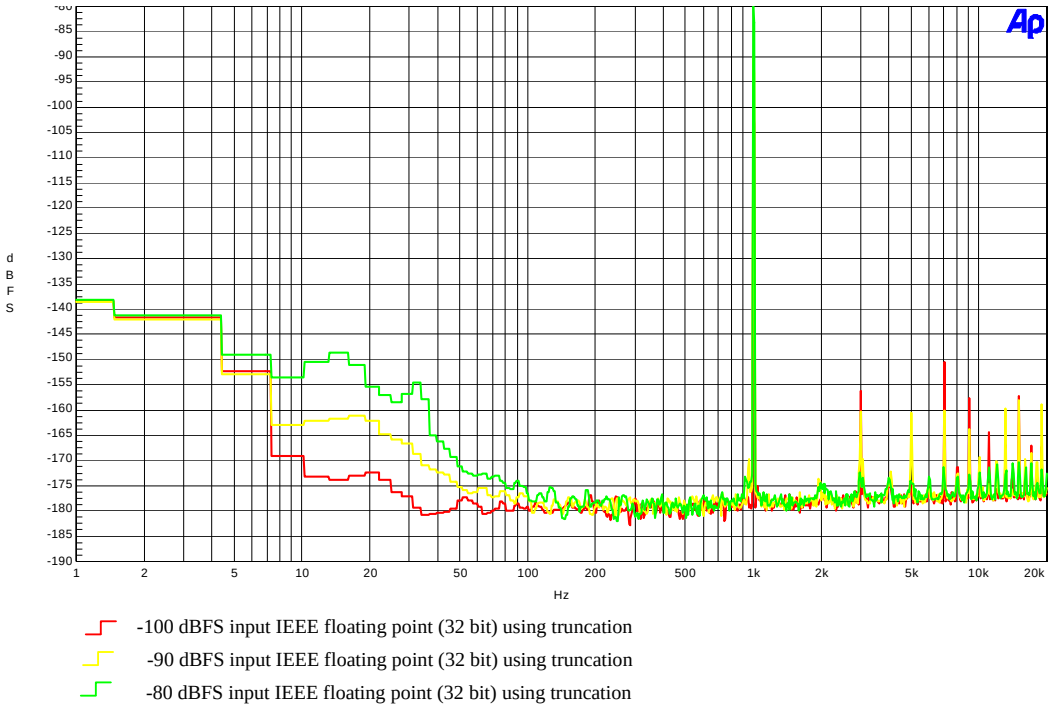

Figure 5-6 DF1output spectrum from DSP platform, 32 bit IEEE floating point, using truncation. Bell filter (20 Hz, –18dB, Q=8.65), sine input, 999.023 Hz, 16 FFT averages 16384 bins, BH4 window.

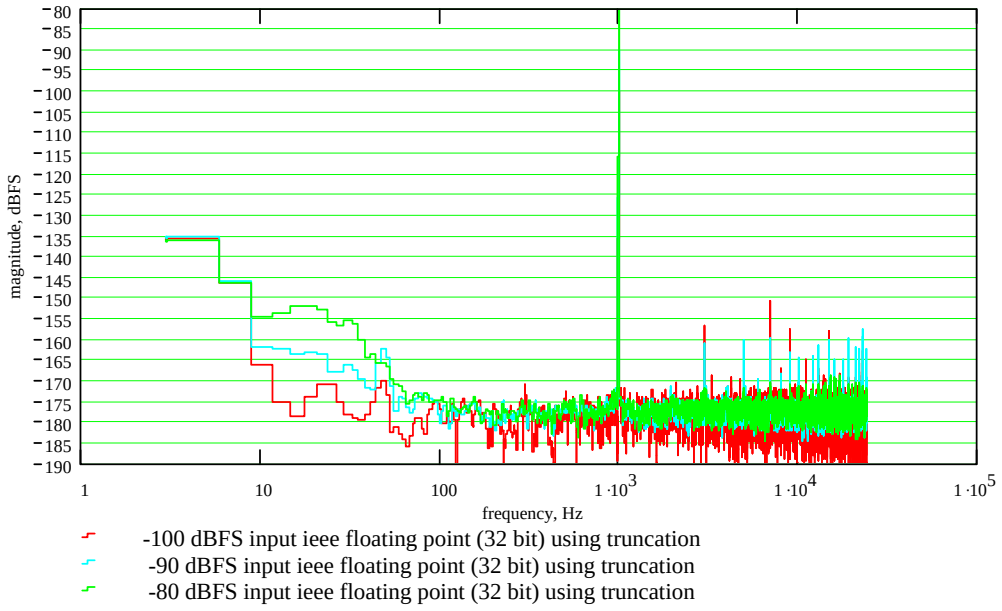

Figure 5-7 DF1output spectrum from Mathcad emulation , 32 bit IEEE floating point, using truncation. Bell filter (20 Hz, –18dB, Q=8.65), sine input, 999.023 Hz, 4 FFT averages 16384 bins, BH4 window.

Figure 5-8 DF1output spectrum from DSP platform, 32 bit IEEE floating point, using rounding. Bell filter (20 Hz, –18dB, Q=8.65), sine input, 999.023 Hz, 16 FFT averages 16384 bins, BH4 window.

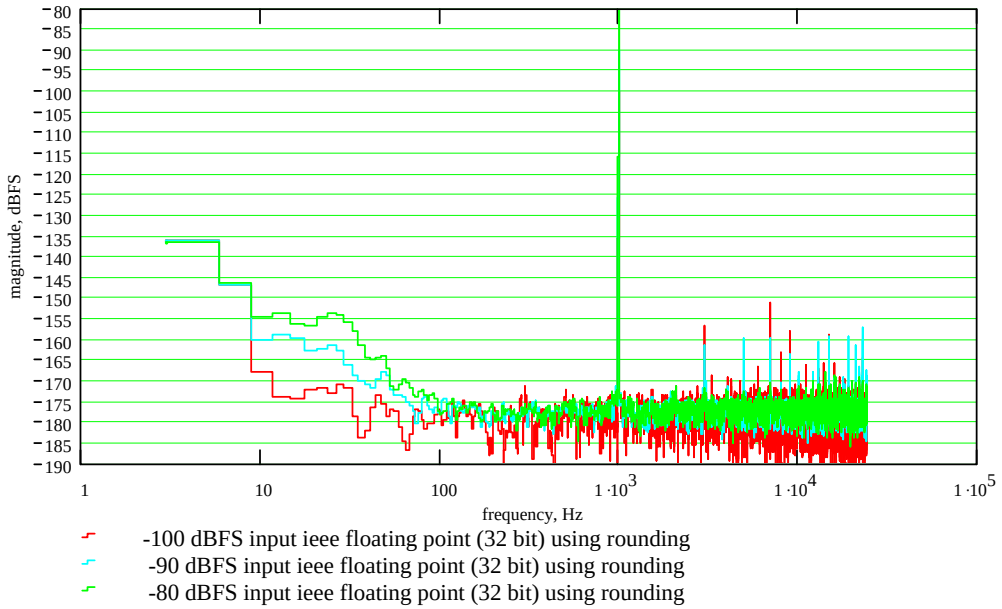

Figure 5-9 DF1output spectrum from Mathcad emulation , 32 bit floating point, sign magnitude using rounding. Bell filter (20 Hz, –18dB, Q=8.65), sine input, 999.023 Hz, 4 FFT averages 16384 bins, BH4 window.

5.4 Filter topology noise analysis, sampling at 48 kHz

In this section finite wordlength filter topology noise analysis is performed using near maximum level single sinusoidal input (-1dBFS). As discussed in Section 5.2.3 the sinusoidal input frequency is bin centred, 999.023 Hz, and does not produce a highly correlated quantisation noise. This section studies the topologies operating at a sampling frequency of 48 kHz. FFT's (four averaged, 16384 point, using the BH4 window) are used to provide spectrum analysis of noise and distortion components of the filter's output. Broad band RMS THD+N figures are used for comparative purposes. These figures suggest the potential dynamic range of the filter topologies.

5.4.1 Direct Form 1

Direct Form 1 (DF1) can be implemented in fixed point arithmetic, without scaling, through the use of twos-complement modulo wrap-round. Therefore the DF1 was implemented in both fixed and floating point arithmetic. Figure 5-10 shows the DF1 output spectrum for a single precision, 24 bit fixed point implementation using truncation. The filter response used was the 20 Hz notch filter case. It is clear that the dc offset (negative bias) in the truncation noise is greatly amplified by the pole transfer function. The rounding accumulator does not suffer from this since the rounding quantiser produces no dc offset. Using first order error feedback (addition of a zero at dc in the error transfer function) greatly reduces the pole transfer function noise, inherent in low frequency high Q filters. The same implementation was made for the DF1, 24 bit fixed point using the bell boost case filter. The output spectrum is shown in Figure 5-11. The bell boost case filter produces less LF noise compared to the notch filter. This is due to less gain in the bell filter pole transfer function.

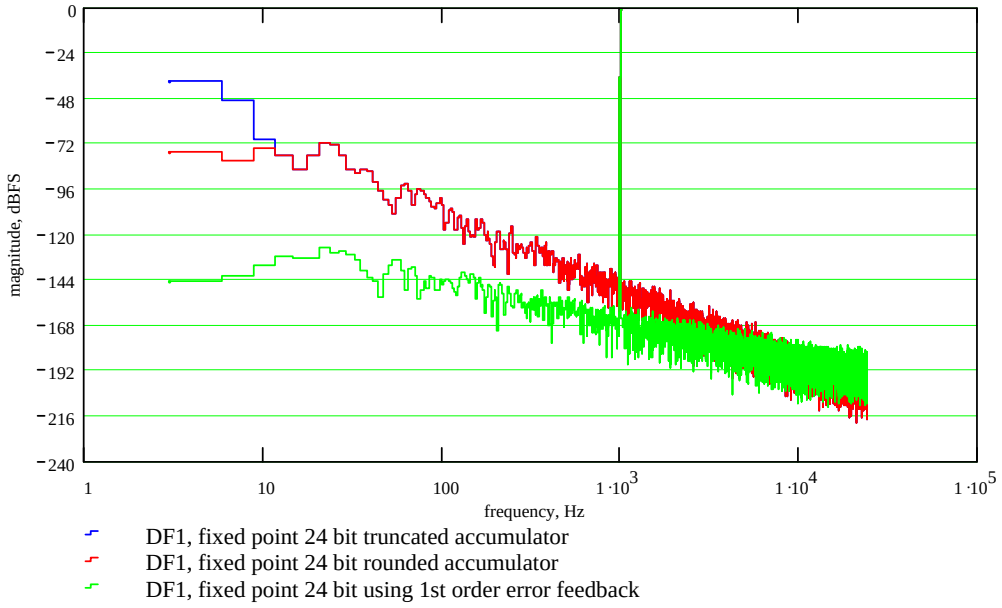

Figure 5-10 DF1 output spectrum for 24 bit fixed point implementations, 20 Hz notch filter, 7.5 Hz bandwidth.

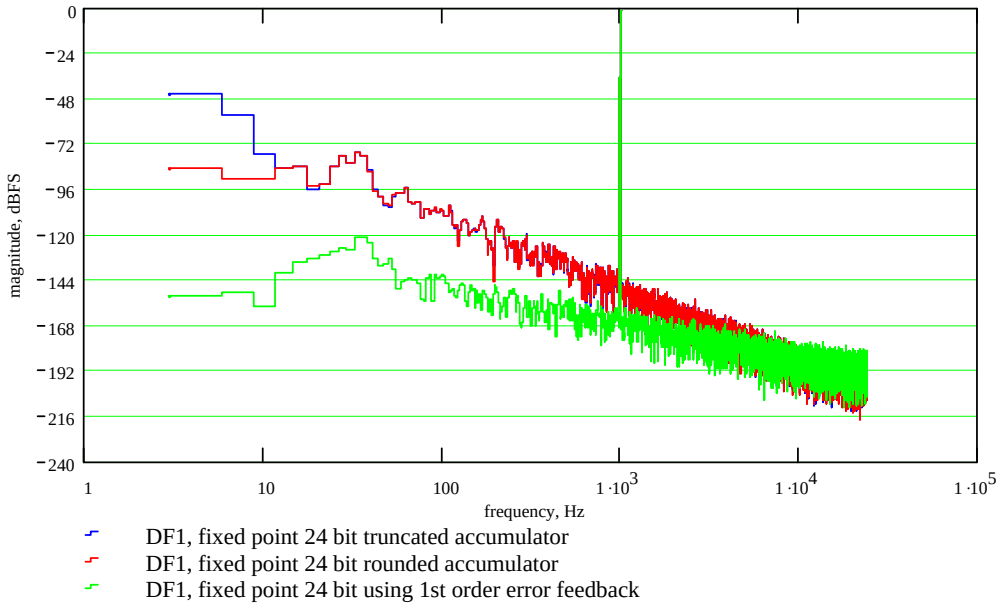

Figure 5-11 DF1 output spectrum for 24 bit fixed point implementations, Bell filter, 30 Hz, Q of 3, Gain 18dB.

Figure 5-12 shows the DF1 output spectrum implemented in 24 bit double precision and 32 bit fixed point. State variables stored at 24 bit double precision is equivalent to 46 fractional bits. The filter response used was the 20 Hz notch filter. Comparison of Figure

5-10 and Figure 5-12 shows that a 24 bit wordlength, using first order error feedback (zero at dc) is similar in noise performance to a 32 bit wordlength, using rounding. The truncated 32 bit DF1 produces noticeable dc offset, approximately –88 dBFS, Figure 5-12.

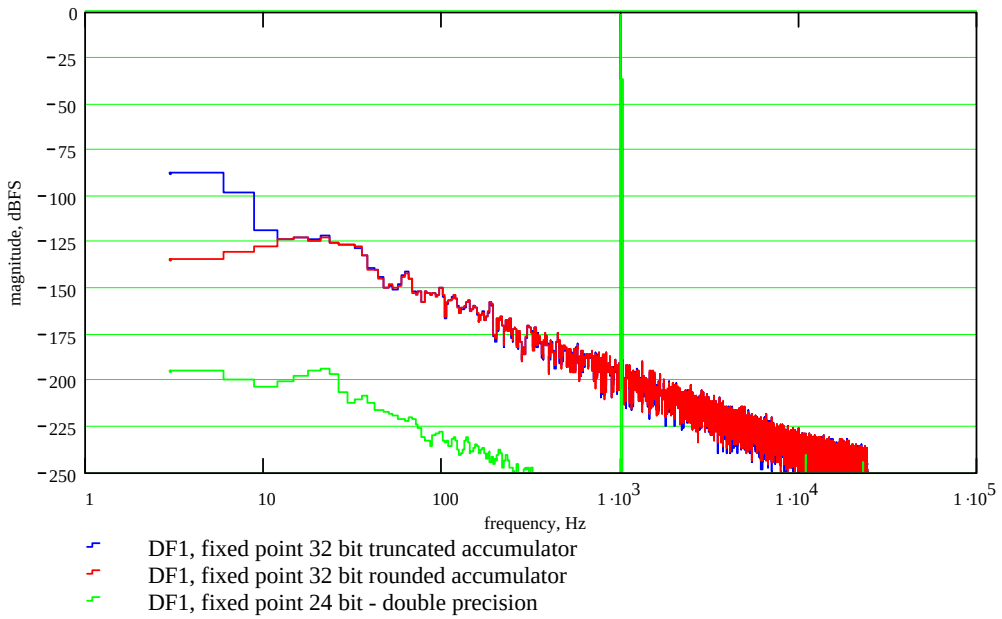

Figure 5-12 DF1 output spectrum for single precision 32 bit and double precision 24 bit, fixed point implementations, 20 Hz notch filter, 7.5 Hz bandwidth.

Figure 5-13 shows the DF1 output spectrum using sign magnitude and twos-complement 32 bit floating point implementations, where rounding is used at each quantisation point. Figure 5-14 shows the same measurement for twos-complement and sign magnitude truncation quantisation. It is clear that the truncation quantiser under floating point produces considerably more harmonic distortion than for rounding quantisation. Distortion products from floating point extended precision implementation (40 bits, using 31 fractional bits) are shown in Figure 5-15 and Figure 5-16. Figure 5-15 shows the DF1 output spectrum, rounding to 31 fractional bits (sign magnitude and twos-complement). Figure 5-16 shows the same measurement for truncation quantisation. The noise performance is similar to that produced by 32 bit fixed point arithmetic and is inferior to fixed point 24 bit double precision (46 fractional bits), Figure 5-12.

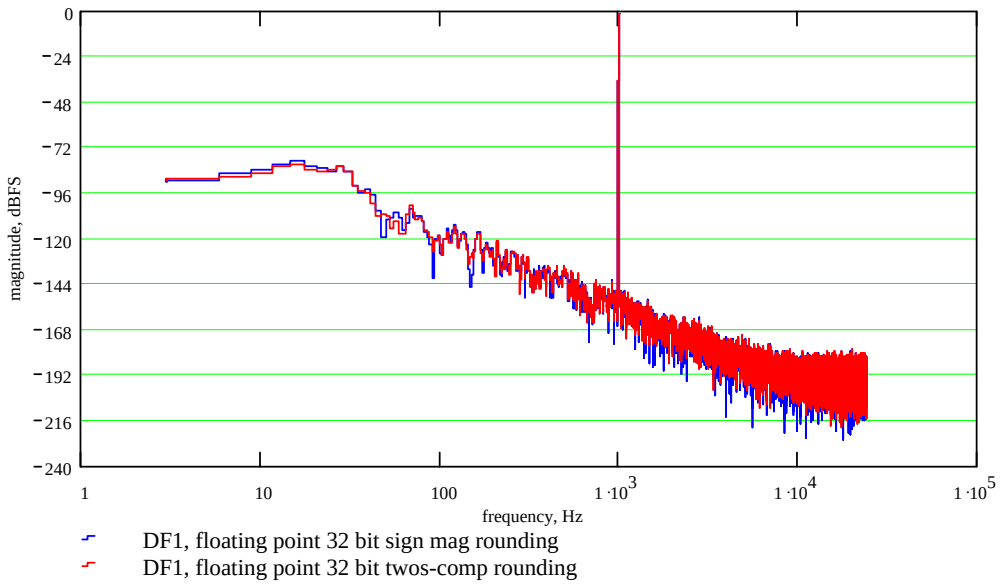

Figure 5-13 DF1 output spectrum for single precision, 32 bit floating point implementations using rounding, 20 Hz notch filter, 7.5 Hz bandwidth.

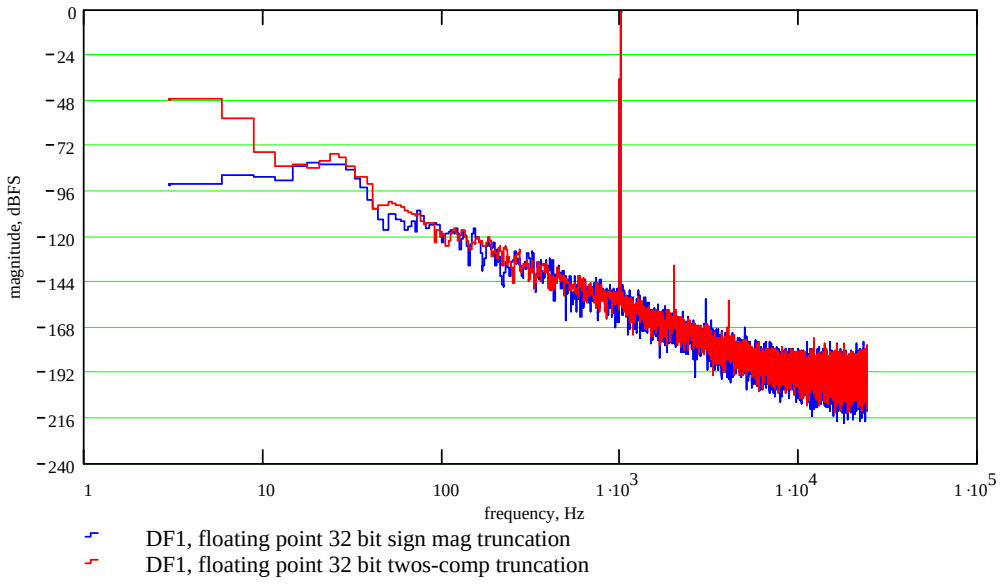

Figure 5-14 DF1 output spectrum for single precision, 32 bit floating point implementations using truncation, 20 Hz notch filter, 7.5 Hz bandwidth.

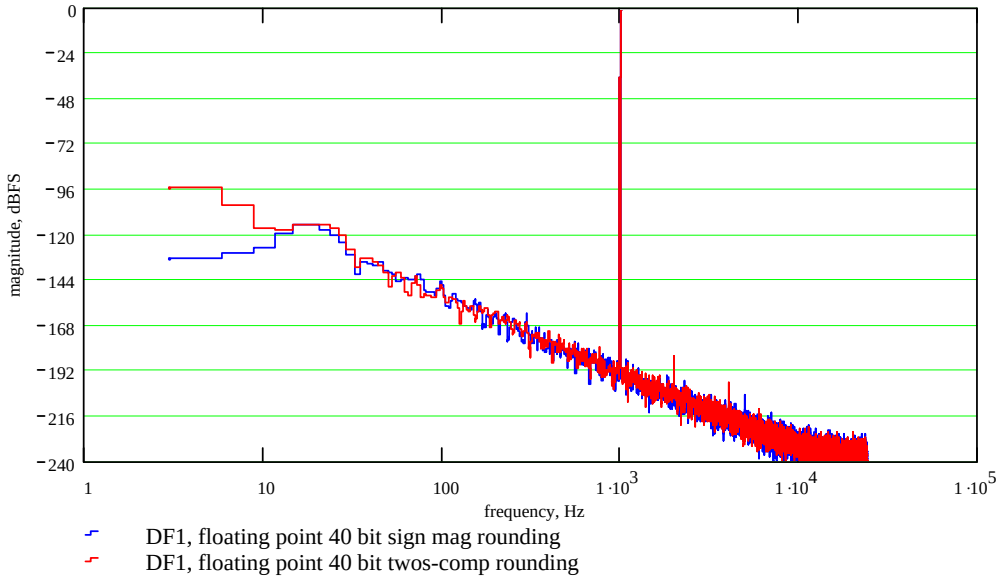

Figure 5-15 DF1 output spectrum for extended precision 40 bit floating point implementations using rounding, 20 Hz notch filter, 7.5 Hz bandwidth.

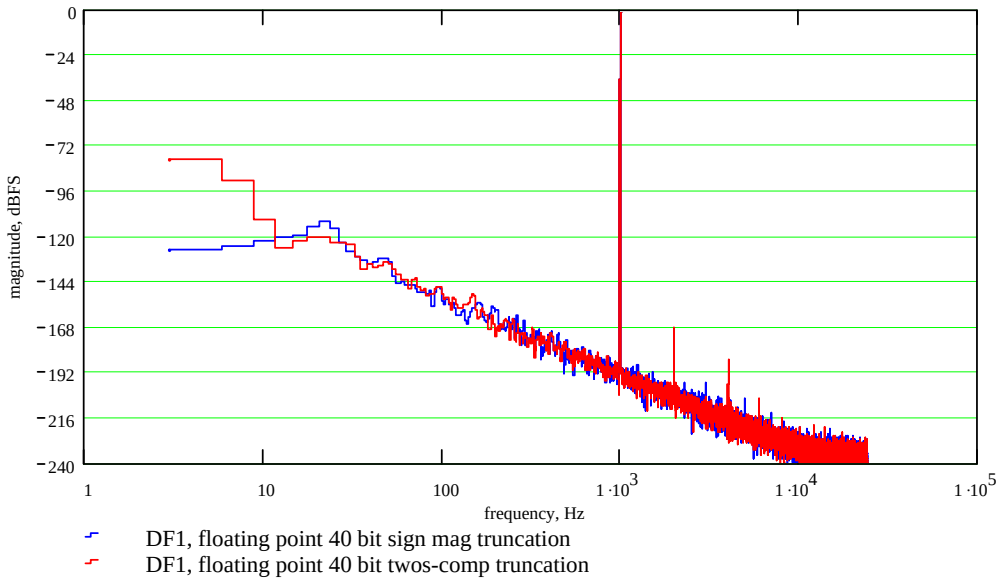

Figure 5-16 DF1 output spectrum for extended precision 40 bit floating point implementations using truncation, 20 Hz notch filter, 7.5 Hz bandwidth.

5.4.2 Direct Form 2

The Direct Form 2 (DF2) is a pole before zero topology. DF2 cannot use modulo wrap-round and relies on scaling to avoid overflow in fixed point implementations. Therefore, in this work, DF2 is only considered for floating point implementation. The DF2 error

transfer function contains the pole and zero transfer functions and thus the error transfer function resembles that of the overall filter frequency response. Therefore filters with gain in the magnitude frequency response typically produce worse case noise performance. The bell boost filter is favoured as a worse case test and is a filter response with high gain in its pole transfer function.

Figure 5-17 shows the noise behaviour for DF2 realising the 20 Hz notch filter. The overall noise floor response is flat in the 20 Hz pole region, despite the high pole gain.

Figure 5-17 also shows the intermediate pole only output spectrum, which suffers from high noise gain in the 20 Hz pole region.

Figure 5-17 also shows the 20 Hz notch filter zero transfer function, that attenuates this pole noise energy at the final output of DF2. Figure 5-18 shows the DF2 realising the 30 Hz bell boost filter. Its intermediate (pole only) output produces approximately 10dB less gain than the 20 Hz notch filter pole only output. This results in the intermediate pole only noise product being 10dB less than that of the notch filter. However the 30 Hz bell filter zero transfer function produces less attenuation and consequently its overall noise product is larger than the 20 Hz notch filter, as shown in

Figure 5-17 and Figure 5-18.

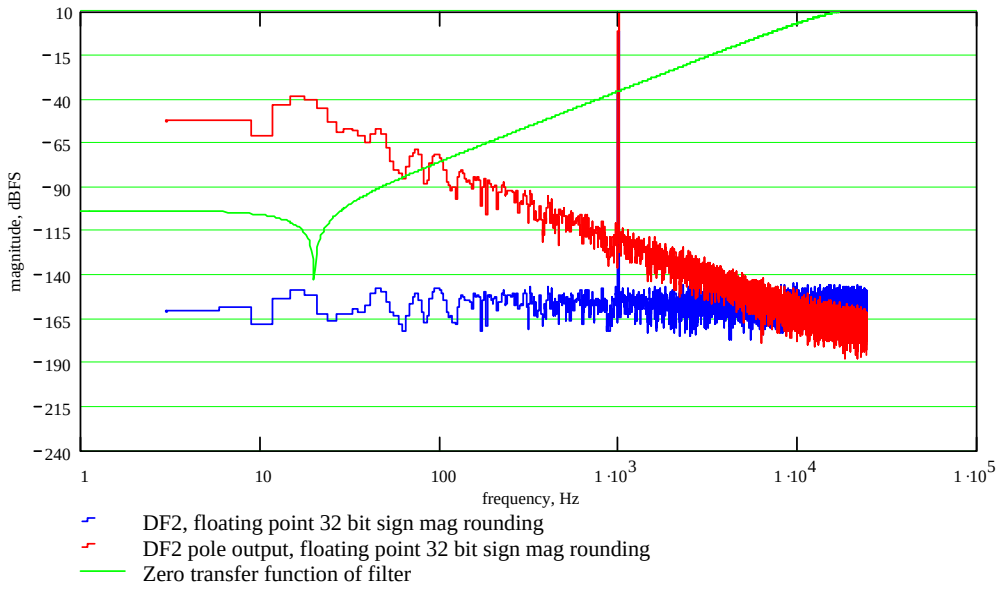

Figure 5-17 DF2 output spectrum for single precision 32 bit floating point implementation using rounding, 20 Hz notch filter, 7.5 Hz bandwidth.

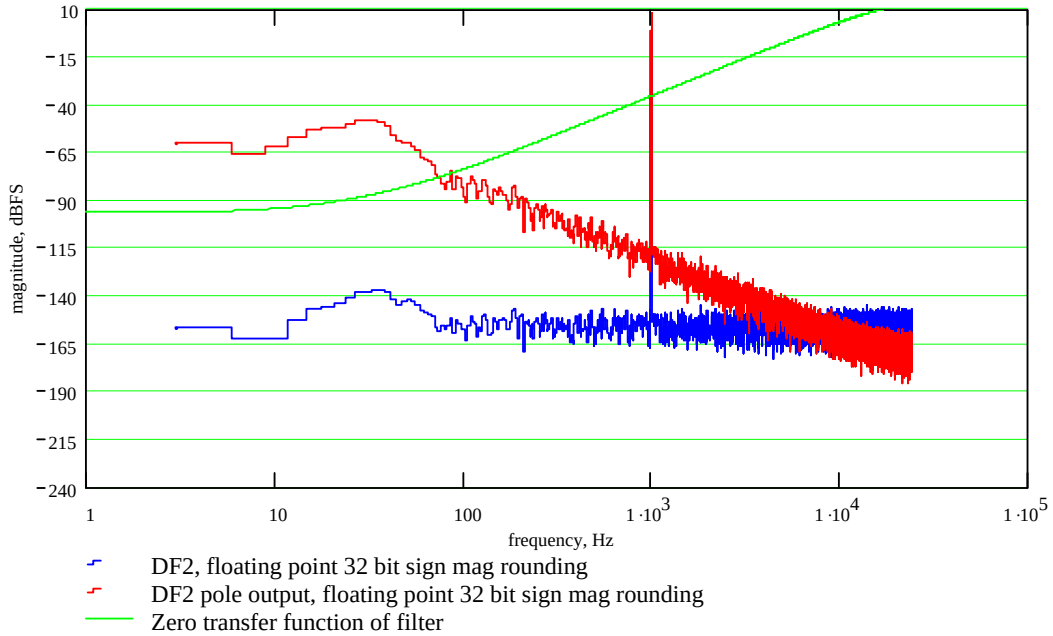

Figure 5-18 DF2 output spectrum for single precision 32 bit floating point implementation using rounding, Bell Filter 30Hz, Q of 3, gain 18dB.

Figure 5-19 shows the DF2 output spectra, using 32 bit floating point twos-complement and sign magnitude truncation. Harmonic distortion caused by the truncation of twos-complement and sign magnitude coded data is visible. Output spectra for DF2 using

floating point 40 bit extended precision (31 fractional bits), are shown in Figure 5-20 and Figure 5-21. Even order harmonic distortion components from truncating twos-complement data are most noticeable, Figure 5-21.

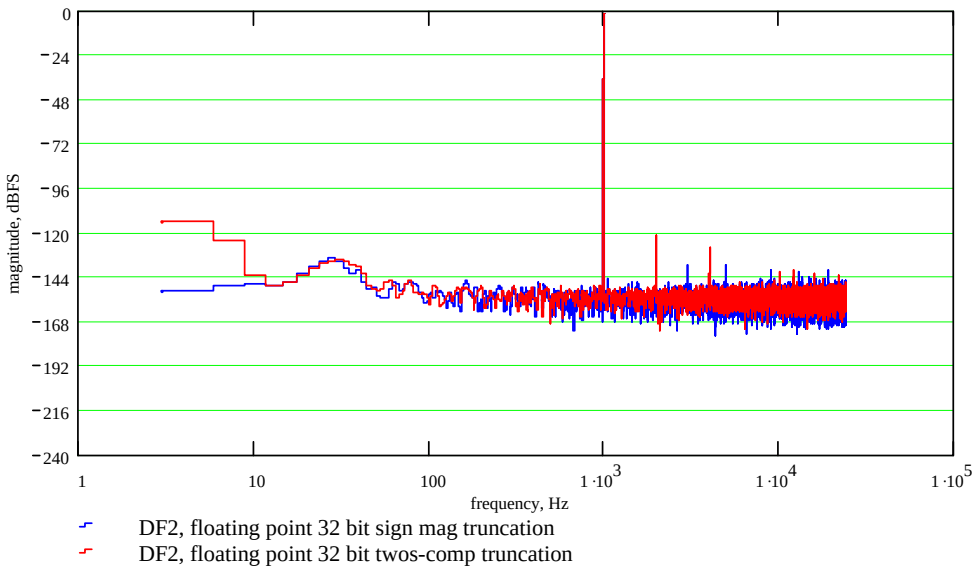

Figure 5-19 DF2 output spectrum for single precision 32 bit floating point implementation using truncation, Bell Filter 30Hz, Q of 3, gain 18dB.

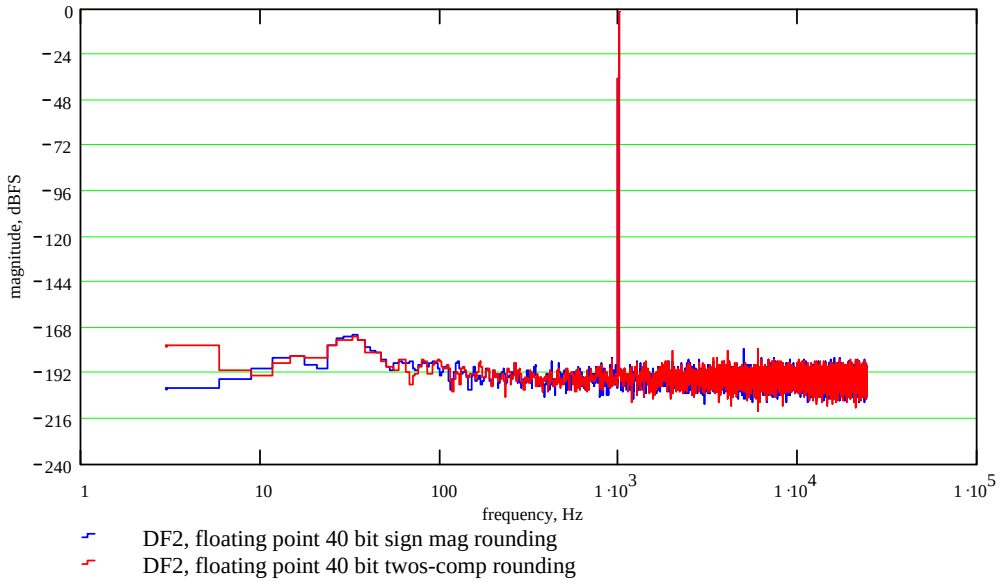

Figure 5-20 DF2 output spectrum for extended precision 40 bit floating point implementation using rounding, Bell Filter 30Hz, Q of 3, gain 18dB.

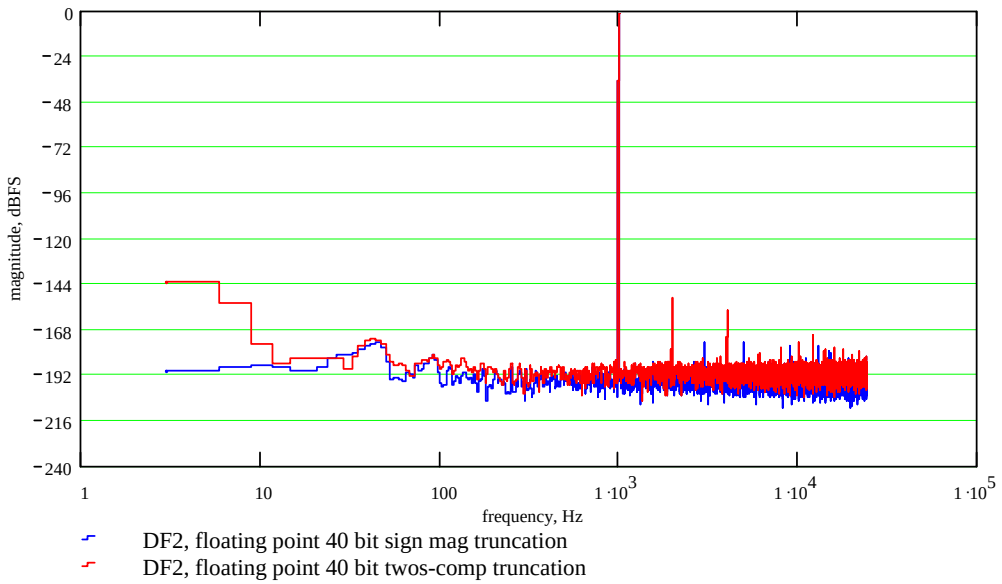

Figure 5-21 DF2 output spectrum for extended precision 40 bit floating point implementation using truncation, Bell Filter 30Hz, Q of 3, gain 18dB.

5.4.3 Transposed Direct Form 1

The Transposed Direct Form 1 (DF1T) is a pole before zero topology. DF1T relies on scaling for fixed point implementations. Therefore DF1T is only examined in floating point implementations. Using the 30 Hz bell filter, DF1T is compared to DF2 in Figure 5-22, using single precision 32 bit floating point sign magnitude coding. DF1T produces slightly more noise than DF2. This is due to the additional quantisation points (noise source) in the topology, Figure 2-9. The RMS THD+N figures for the DF1T in the example shown in Figure 5-22 is 2 dB greater than that of the DF2 (DF1T produces -115 dBFS, DF1T produces -113 dBFS).

Figure 5-23 shows the noise products using extended precision 40 bit (31 fractional bits) floating point arithmetic. The corresponding RMS THD+N figure for the DF1T is –153.2 dBFS and for the DF2 is -150.9 dBFS. These results are explained by the nature of quantisation noise in extended precision implementation. Single precision implementation generates relatively large quantisation noise source at the points in the topology prior to

state variable storage. In floating point extended precision implementations every arithmetic operation is potentially a quantisation noise source, although much smaller noise source, as discussed in Chapter 2. The DF1T topology, Figure 2-9, has separate addition nodes for the pole paths prior to their addition to the relatively small input (magnitude less than unity). This forces the pole path state variables to be summed prior to the addition with the input. Since the pole path state variables are large in magnitude, summing these together prior to summation with the filter input generates a smaller overall quantisation error, under finite wordlength floating point addition.

The DF2 implementation uses one accumulator at this point in the topology, Figure 2-8. Therefore the state variable and input summation ordering has implications in the resulting quantisation error and ultimate noise performance in extended precision floating point implementations. Figure 5-23 shows extended precision implementations of DF1T, DF2 (summing input with the first pole path, then the second pole path) and DF2 summing the pole paths first, then the input). The latter DF2 is similar to the DF1T implementation and produces a slightly superior noise characteristic, as shown in Figure 5-23.

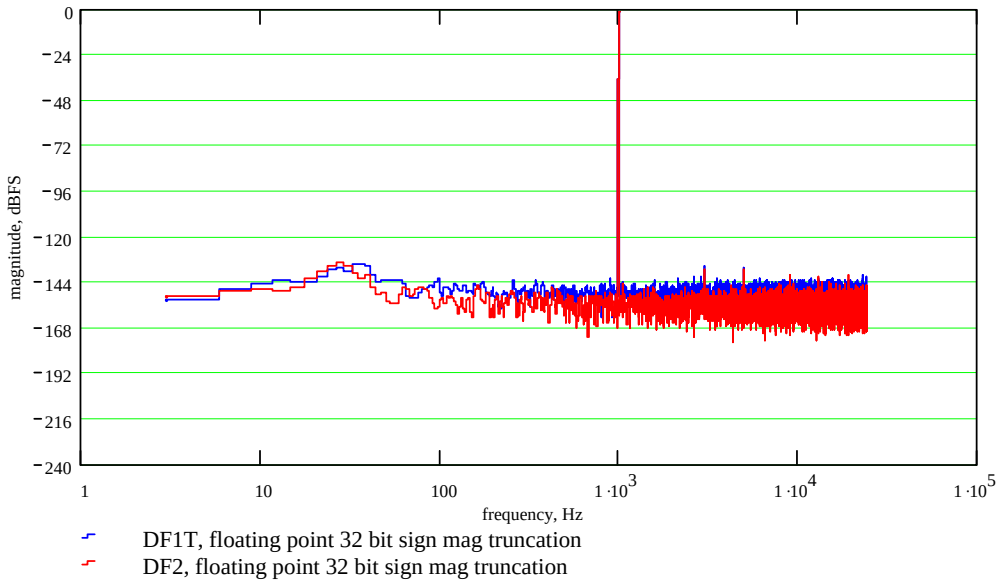

Figure 5-22 DF1T and DF2 output spectrum for single precision 32 bit floating point implementation using truncation, Bell Filter 30Hz, Q of 3, gain 18dB.

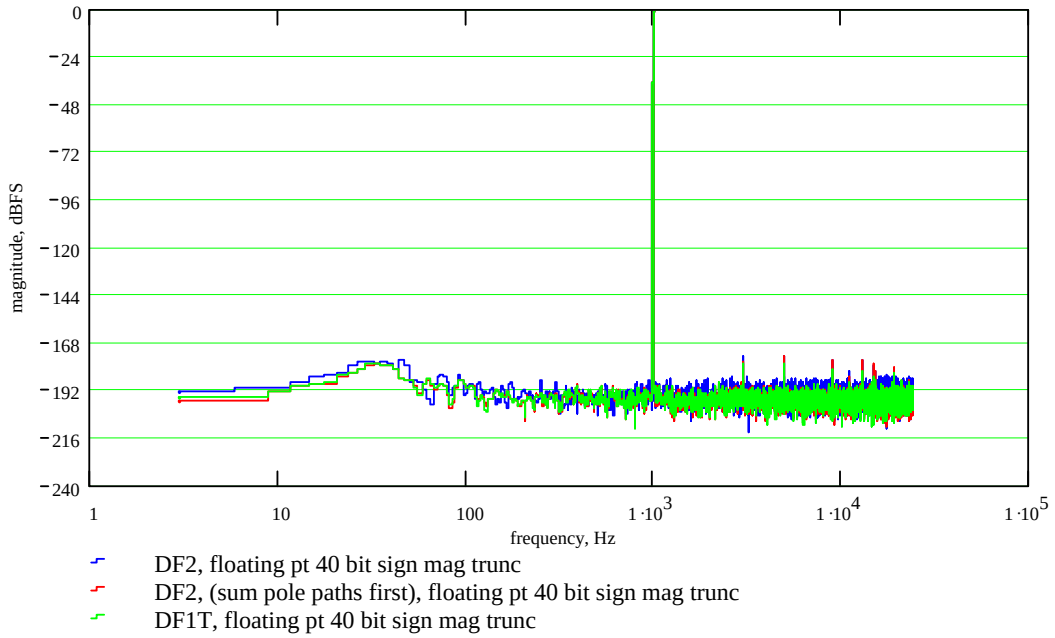

Figure 5-23 DF1T and DF2 output spectrum for extended precision 40 bit floating point implementation using truncation, Bell Filter 30Hz, Q of 3, gain 18dB.

5.4.4 Transposed Direct Form 2

The Transposed Direct Form 2 (DF2T) is a zero before pole topology. Despite its two accumulators, it can utilise modulo wrap-round to avoid accumulator overflow. Therefore the DF2T is examined under fixed and floating point implementations. The DF2T noise characteristics are similar to DF1 and are tested using the 20 Hz notch filter. Figure 5-24 and Figure 5-25 show the output spectra of DF2T and DF1 using 24 bit fixed point arithmetic for truncation and rounding. The DF2T produces approximately 12 dB more noise across the spectrum compared to DF1. This can be attributed to the additional quantisation points in the topology, Figure 2-10. Figure 5-26 compares output spectra of DF2T to DF1 under 32 bit single precision floating point implementation. Figure 5-27 compares the output spectra of DF2T to DF1 under 40 bit extended precision floating point implementation. Noise performance differences between DF1 and DF2T for both floating point implementations are considerably less noticeable.

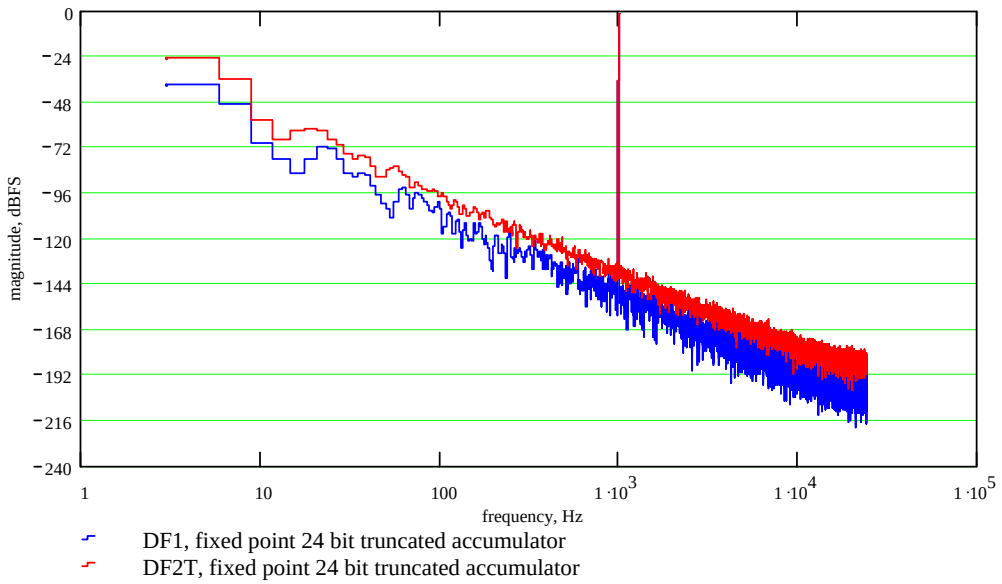

Figure 5-24 DF2T and DF1 output spectrum for 24 bit fixed point implementations, using truncation, 20 Hz notch filter, 7.5 Hz bandwidth.

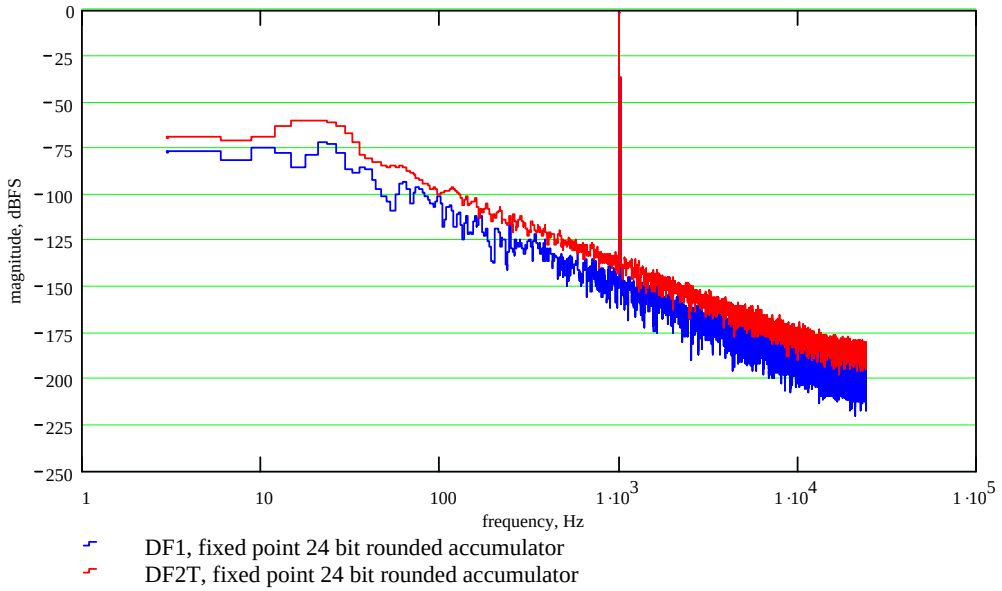

Figure 5-25 DF2T and DF1 output spectrum for 24 bit fixed point implementations, using rounding, 20 Hz notch filter, 7.5 Hz bandwidth.

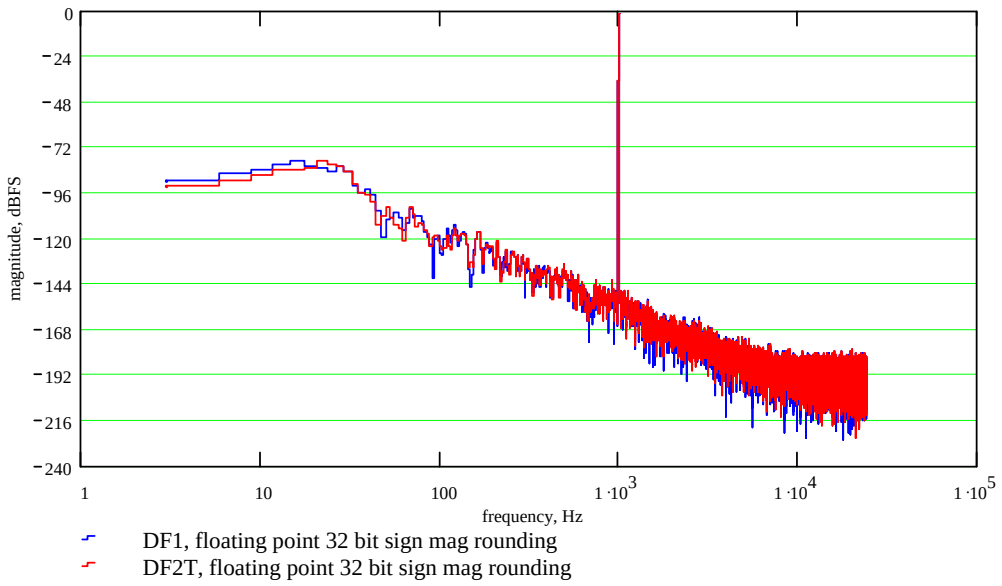

Figure 5-26 DF2T and DF1 output spectrum for single precision 32 bit floating point implementations, using rounding, 20 Hz notch filter, 7.5 Hz bandwidth.

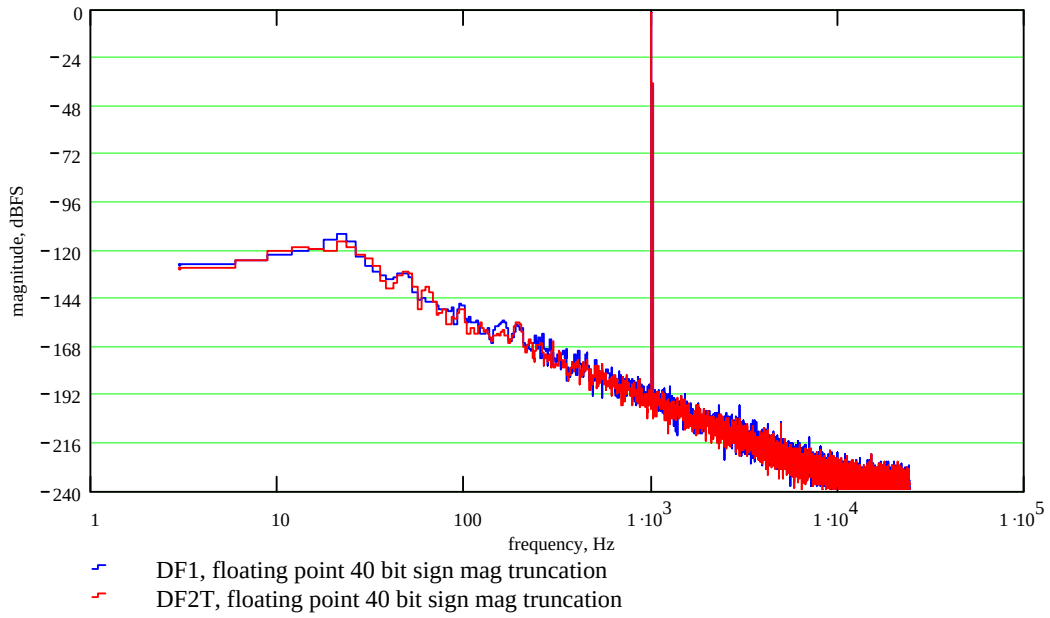

Figure 5-27 DF2T and DF1 output spectrum for extended precision 40 bit floating point implementations, using truncation, 20 Hz notch filter, 7.5 Hz bandwidth.

5.4.5 Coupled Forms (Gold-Rader, Zölzer, Kingsbury)

The coupled forms are all zero before pole topologies. They also require scaling to prevent overflow and are therefore only considered for floating point implementation in

this work. Figure 5-28 shows the output spectra for the couple forms, using 32 bit single precision floating point implementation, sign magnitude coding, truncating at quantisation points. The filter response used was the 20 Hz notch filter. All three coupled forms have high pole gain in their error transfer functions, however for LF tuned filters the coupled forms provide some additional noise rejection compared to DF1. RMS THD+N figures were derived from the time responses,

DF1	- 80.5 dBFS,
Gold-Rader	- 98.5 dBFS,
Kingsbury	- 105.6 dBFS,
Zölzer	- 106.9 dBFS.

However the coupled forms noise performance does not compare as well for high frequency tuned filters. Figure 5-29 shows the resultant noise products using the 19 kHz notch filter. The corresponding RMS THD+N figures are also given,

DF1	-124.7 dBFS,
Gold-Rader	-118.3 dBFS,
Kingsbury	-118.8 dBFS,
Zölzer	-118.3 dBFS.

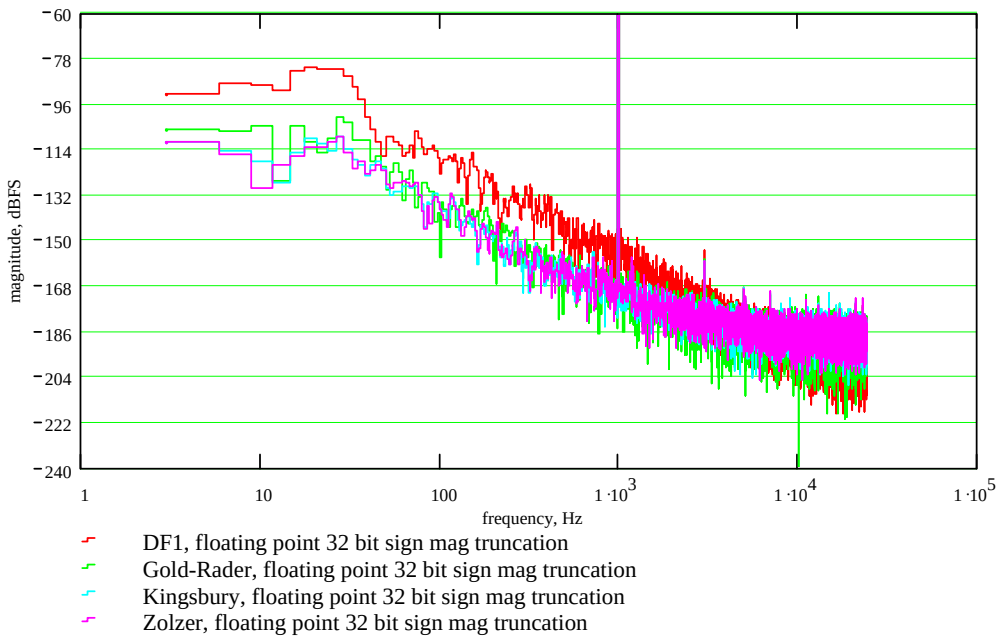

Figure 5-28 Coupled forms output spectra for single precision 32 bit floating point, sign magnitude using truncation, 20 Hz notch filter, 7.5 Hz bandwidth.

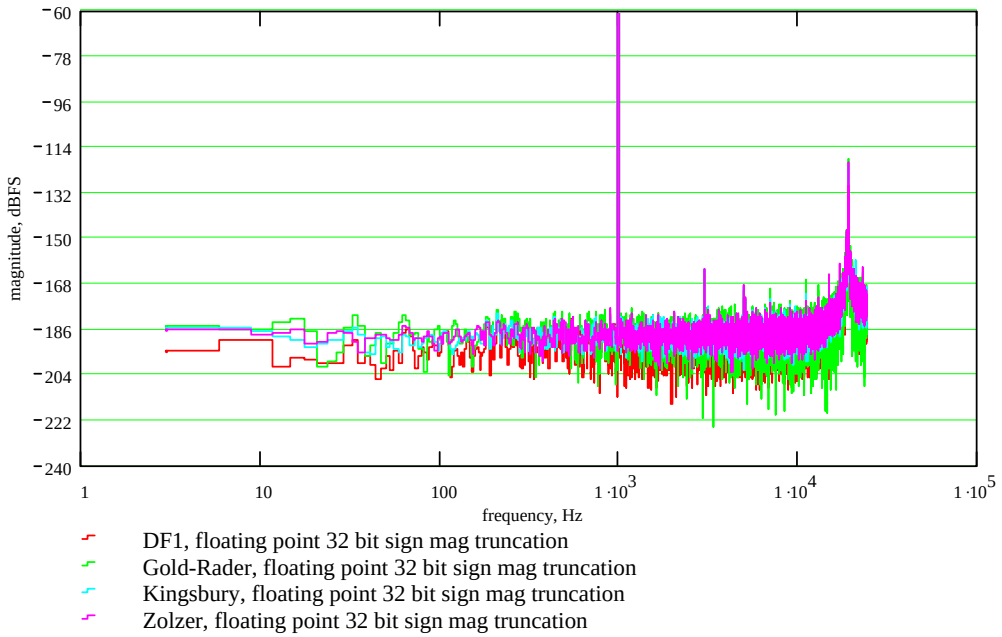

Figure 5-29 Coupled forms output spectra for single precision 32 bit floating point, sign magnitude using truncation, 19 kHz notch filter, 7.5 Hz bandwidth.

5.4.6 State-space structure

The state-space structure (or normal form) utilises L_2 scaling at internal nodes to minimise overflow. Therefore the structure is prone to overflow under high level input conditions whose frequency content is in the tuned pole frequency region. Therefore the structure is implemented under floating point for the purposes of this work. Furthermore the topology coefficients are not typically fractional. Coefficient scaling is complex, making fixed point implementation computationally exhaustive. The topology is not simply zero before pole, there are scaling coefficients after the pole transfer function and part of the zero transfer function is in parallel with the pole transfer function implementation.

Figure 5-30 shows the output spectrum of the state-space structure realising the 20 Hz notch filter example. The measurements shown are for 32 and 40 bit floating point wordlengths using sign magnitude coding. The measurement for 32 bit truncated wordlengths produce a THD+N figure of –150.6 dB. The measurement for 32 bit

rounding produces a THD+N figure of -156.9 dB. The 40 bit (extended precision) measurement produces a THD+N figure of -193.2 dB.

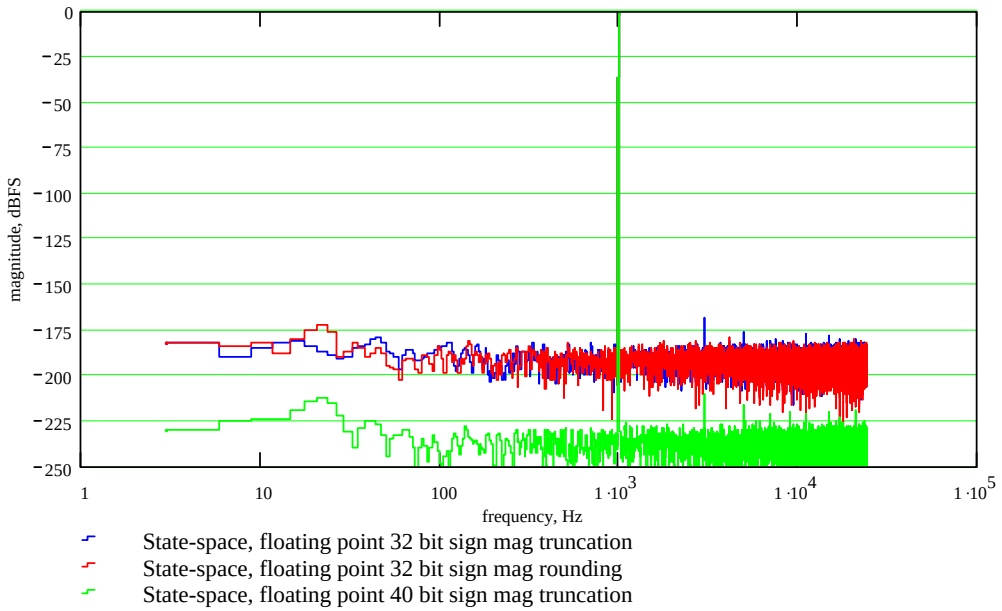

Figure 5-30 State-space output spectra for 32 and 40 bit floating point implementations, sign magnitude coding, 20 Hz notch filter, 7.5 Hz bandwidth.

5.4.7 State-space hybrid (Cabot) structure

The state-space hybrid, Cabot, structure, uses the pole transfer function implementation used in the state-space structure and couples that with a direct form zero transfer function implementation. The resulting zero before pole topology is shown in Figure 2-15. Through observation of numerical behaviour at the accumulator nodes it was found that for the 20 Hz notch filter, using a full scale sinusoidal input at various frequencies the topology did not generate accumulator overflow. Although this does not prove that the topology is immune from overflow, the topology was examined under fixed and floating point implementation. The resulting output spectra for 24 bit fixed and 32 bit floating point implementations are shown in Figure 5-31. For fixed point operation the topology in was found to produce similar noise characteristics to that of the DF1. In floating point

implementation the topology noise performance is superior to DF1, compare Figure 5-13 and Figure 5-31.

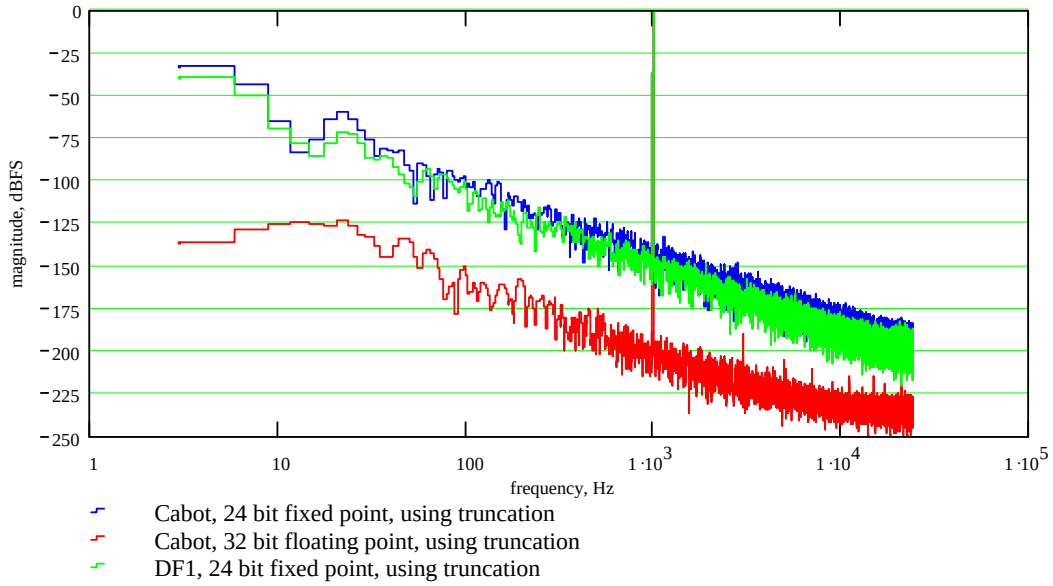

Figure 5-31 Cabot structure output spectra for 24 bit fixed point and 32 bit floating point implementations, 20 Hz notch filter, 7.5 Hz bandwidth.

5.4.8 Lattice and Ladder topologies

The ladder allpass topology is scaled using the L_2 norm. Despite being resilient to overflow, the topology can overflow under high level input excitation in the tuned frequency region. The lattice structure is prone to overflow, since it has unbounded accumulation nodes. Therefore, in this work, all ladder and lattice structures are implemented in floating point arithmetic. Since the structures are pole before zero, the 30 Hz bell boost filter example is used. The two ladder filter implementations, referred to as 'Moorer' and 'Massie' in this work are discussed in Chapter 2, Section 2.5.5 .

Figure 5-32 compares the noise products produced by the Massie and Moorer ladder structures in 32 bit floating point arithmetic, using truncation at quantisation points. The resulting noise floors for the ladder structures resemble the overall filter frequency response. The Moorer implementation produces less high frequency noise than the Massie structure. This is reflected in the respective THD+N figures, -148.8 dBFS for the Moorer

structure and -139.4 dBFS for the Massie structure. The noise products of the ladder allpass are the same as the Massie structure attenuated by 18 dB (the tuned gain factor). Figure 5-33 shows the residual noise produced by the lattice allpass and Massie structures, implemented in 32 bit floating point truncated arithmetic. The THD+N figure for the lattice Massie structure is -132.3 dBFS (7 dB higher than the ladder Massie).

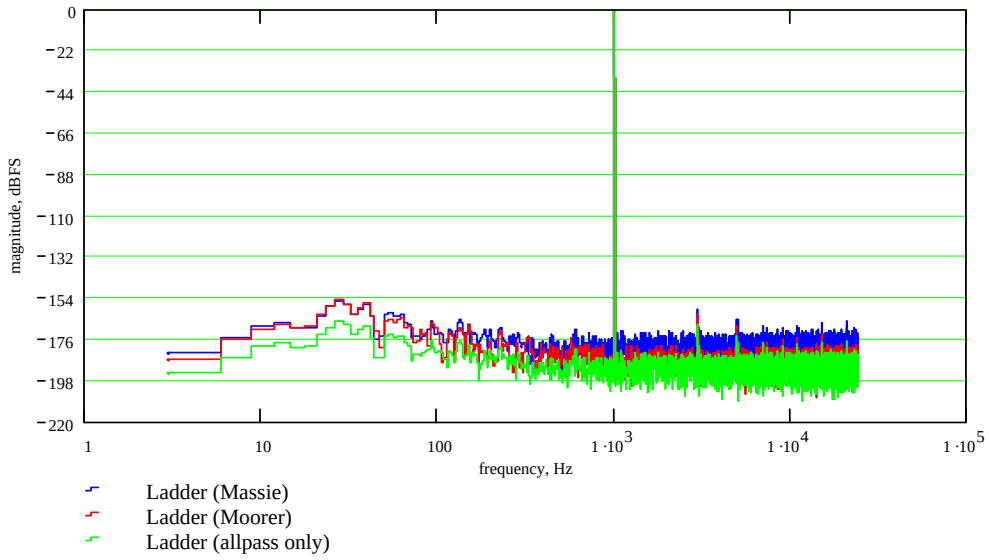

Figure 5-32 Ladder structures output spectra for single precision 32 bit floating point, sign magnitude implementation, 30 Hz bell filter, Q of 3, gain 18dB.

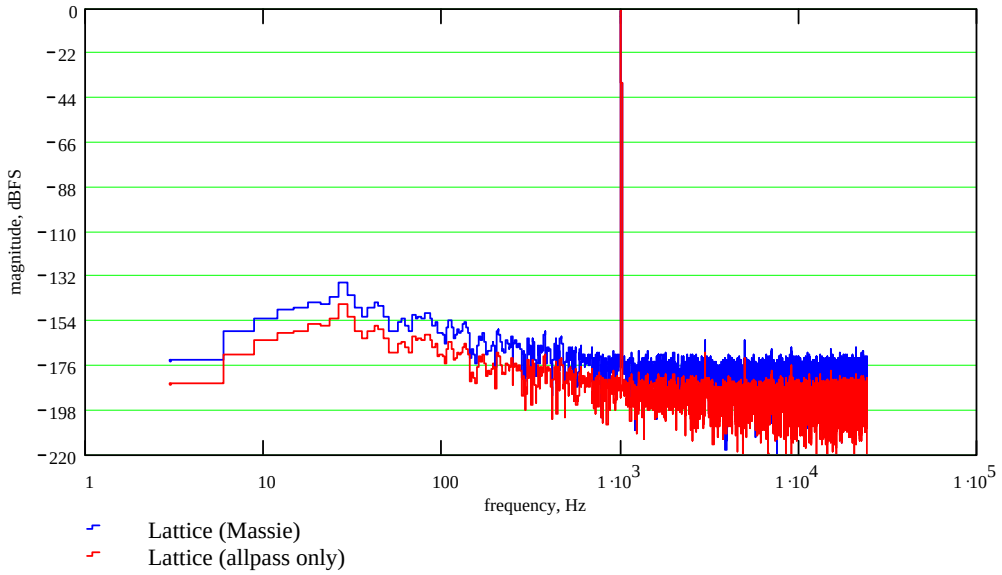

Figure 5-33 Lattice structures output spectra for single precision 32 bit floating point, sign magnitude implementation, 30 Hz bell filter, Q of 3, gain 18dB.

Figure 5-34 shows the ladder structures output spectra for the 19 kHz bell filter with 18 dB of gain. The filter is implemented using 32 bit floating point, sign magnitude coding. The residual noise produced by the Massie structure is 18 dB higher than that of the allpass. The Moorer structure produces lower noise components at low frequency than the Massie structure. The THD+N figure for the Moorer structure is -142.5 dBFS. The THD+N figure for the Massie structure is -138.2 dBFS. Figure 5-35 shows the residual noise produced by the Lattice allpass and Massie structures for the same 19 kHz bell boost filter example. The THD+N figure for the lattice Massie structure is -138.6 dBFS.

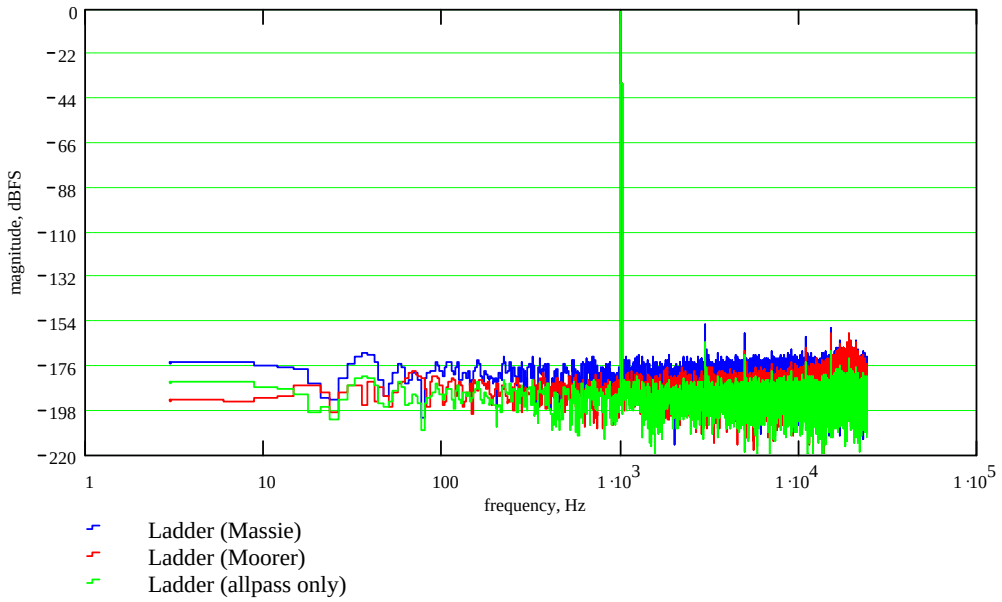

Figure 5-34 Ladder structures output spectra for single precision 32 bit floating point, sign magnitude implementation, 19 kHz bell filter, Q of 3, gain 18dB.

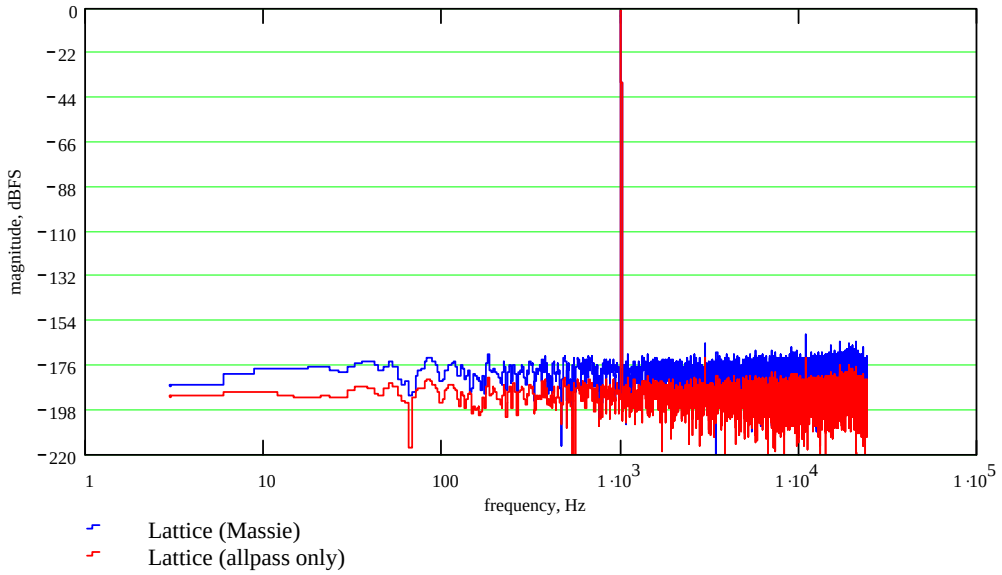

Figure 5-35 Lattice structures output spectra for single precision 32 bit floating point, sign magnitude implementation, 19 kHz bell filter, Q of 3, gain 18dB.

5.5 Low signal level non-linearity

Low level sinusoids at -90 dBFS were used to examine the behaviour of topologies operating at low levels. The 20 Hz notch filter was used due to its high gain pole transfer function, see Section 5.2.2 . Figure 5-36 shows the DF1 output spectrum implemented in 24 bit fixed point arithmetic, using a 999.023 Hz sinusoidal input at –90 dBFS. The harmonic distortion components of the –90 dBFS input sinusoid for 24 bit fixed point representation are visible. Comparing the noise products to that of the same test using a high level sinusoid (-1 dBFS), Figure 5-10, indicates that the peak noise region (20 Hz) has dropped by less than 20 dB, whereas the difference in input level is 89dB. This suggests an unmodulated noise product for change in input level. Fixed point quantiser noise variance is insensitive to changes in signal level, (Barnes, 1985) explaining the unmodulated noise products in the fixed point DF1. Despite the benefits of unmodulated noise products with input level, the low frequency noise products, for fixed point 24 bit DF1, are higher in magnitude than the input signal.

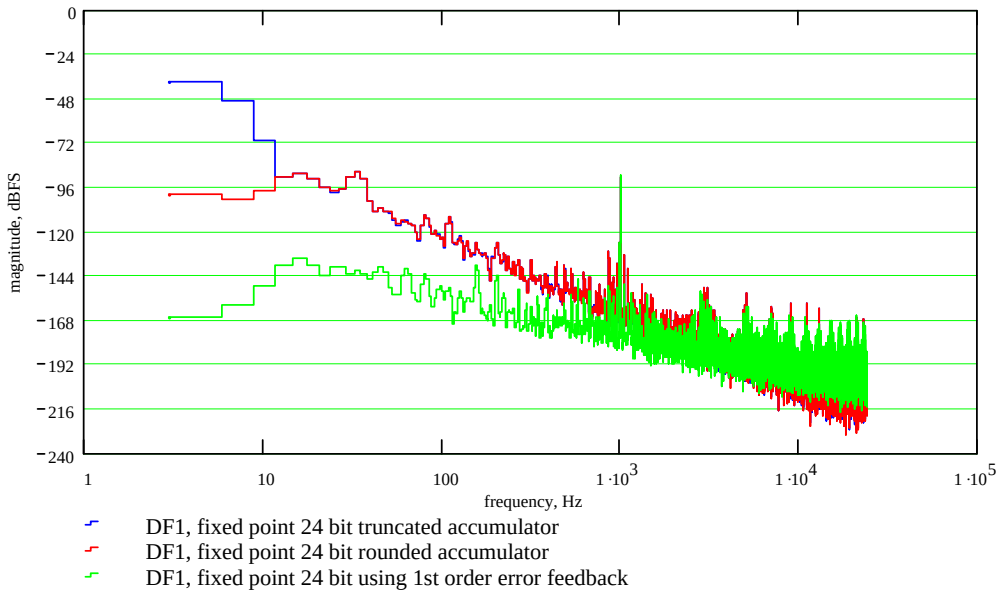

Figure 5-36 DF1 output spectrum using 24 bit fixed point arithmetic, under low level input –90 dBFS, 20 Hz notch filter.

DF1 floating point implementation produces opposing properties. Comparison of Figure 5-13 and Figure 5-37 shows that for floating point implementations the noise products drop by nearly 90 dB for a decrease of 89dB in signal level. This suggests high noise modulation against input operating level, naturally maintaining dynamic range of the filter. All other topologies implemented under floating point arithmetic were found to produce similar properties. The noise products were scaled by the input operating level, producing high noise modulation with change in signal level.

The application of triangular dither to a signal, prior to a quantiser, can eliminate non-linear quantisation distortion and provide an uncorrelated noise floor, with respect to the signal (Vanderkooy and Lipshitz, 1987). The application of dither to DF1 is simple, requiring a single dither source added to the accumulator, prior to quantisation point. The effects of dither in the DF1 topology, for fixed and floating point arithmetic, using low level inputs were examined. Figure 5-37 shows the DF1 output spectrum using floating point arithmetic, adding triangular dither at the mantissa LSB. The dither produces a

noise floor comparable to 24 bit fixed point. This noise floor does not modulate with the input signal level.

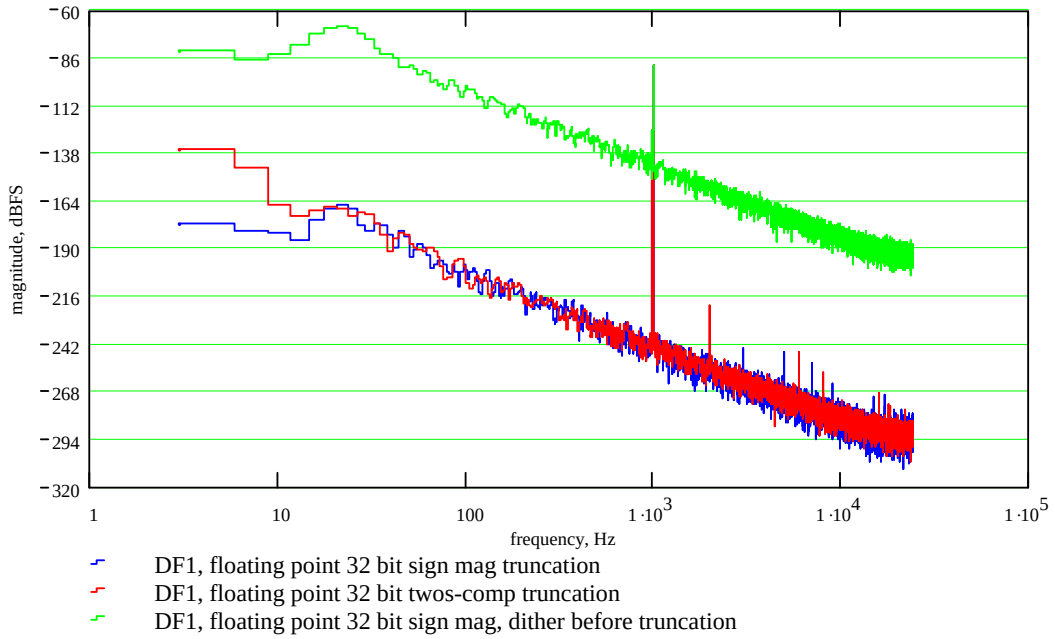

Figure 5-37 DF1 output spectrum using 32 bit floating point arithmetic, under low level input –90 dBFS, 20 Hz notch filter.

Figure 5-38 shows the DF2T noise products, in 24 bit fixed point, at low level operation. Comparing this with DF2T at high level operation, Figure 5-24, shows that noise products do not reduce with operating level proportionally, suggesting low noise modulation with input level. Figure 5-38 shows that the fixed point DF2T, at low operating level, produces less correlated noise products (harmonic distortion) than DF1. This de-correlation of the DF2T quantisation noise is attributable to its multiple quantisation noise source, compared to the single quantisation point in DF1. Figure 5-38 also shows the output spectrum of the 24 bit fixed point DF1 using triangular dither prior to truncation. The dither is shown to eliminate the harmonic distortion components shown in the undithered DF1 plot, at the expensive of a small increase in broad band noise. The undithered DF2T produces similar noise characteristics, namely an absence of harmonic distortion. However, the DF1 using dither produces a slightly lower noise floor than the undithered DF2T.

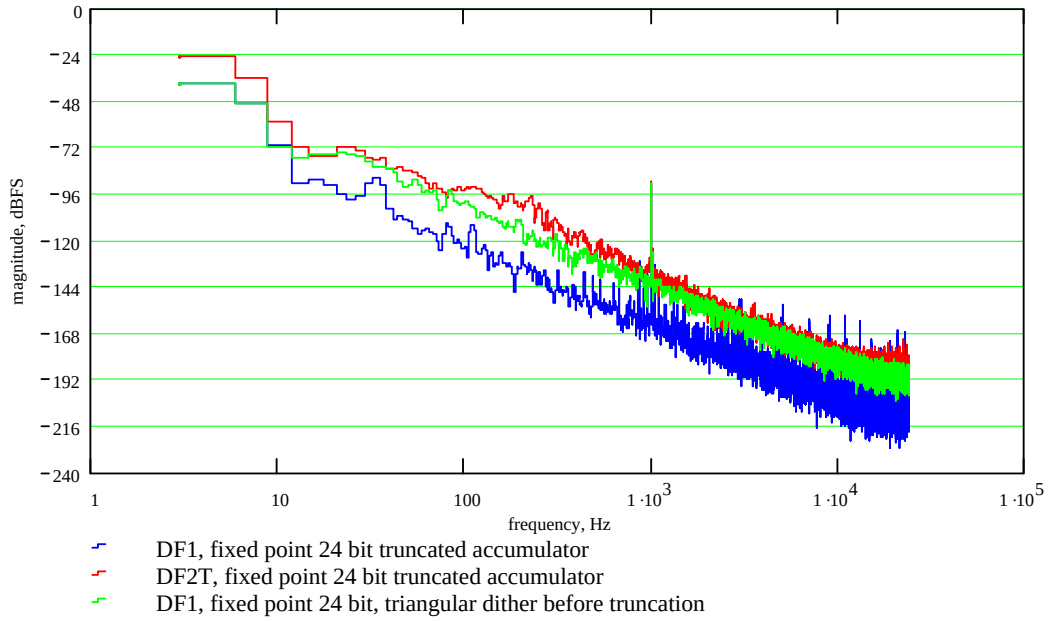

Figure 5-38 DF2T and DF1 output spectrum using 24 bit fixed point arithmetic, under low level input –90 dBFS, 20 Hz notch filter.

5.6 Zero input behaviour

The zero input test, uses a single initial white noise burst as an input stimulus. After an arbitrary period of one thousand samples of white noise, the input is switched to zero, Figure 5-39. At the point of zero input the filter is subjected to a step change and the pole paths produce a decaying oscillation as the filter approaches steady state. This pole ringing was discussed in Section 5.2.5 of this chapter. The limit cycle behaviour is caused by this decaying oscillation in the pole paths diminishing below the finite wordlength limitations of the topology implementation. As the state variables continuously attempt to numerically equal data values smaller than the finite wordlength capabilities, coarsely quantised data values re-circulate in the recursive pole paths. This causes zero input limit cycle behaviour.

During these tests, filter output analysis was conducted after the filter settling period. Limit cycle behaviour is easily distinguished as self-sustained, constant level energy. Non-steady state, filter ringing effects are a decaying energy against time.

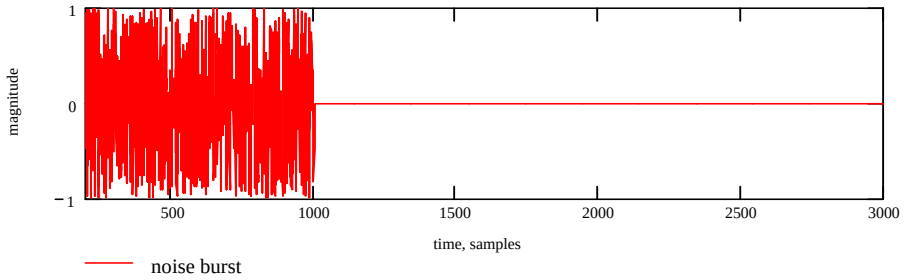

Figure 5-39 Noise burst input used in zero input tests

5.6.1 Zero input limit cycles in fixed point arithmetic

Zero input limit cycle effects are noticeable with fixed point arithmetic and have been widely documented. Jackson (1986) relates limit cycle behaviour to the position of the poles in the stability triangle, Figure 3-20. Numerous tests were conducted using notch filters of varying centre frequency for the DF1 and DF2T topologies, in fixed point implementation. High frequency tuned notch filters (for example, above 15 kHz, assuming a sampling frequency of 48 kHz) with extremely high Q (-3dB bandwidth 7.5 Hz) were found to produce worst case limit cycle behaviour.

However, (Jackson, 1986) reports limit cycle behaviour cannot always be determined by stability triangle rules. In this work, it was found that increasing the gain in the pole transfer function did not ensure an increase in limit cycle magnitude. However the following examples discussed in this section were typical and generalise the behaviour observed in these tests.

Despite the immunity of sign magnitude truncation to zero input limit cycles (Claasen et al, 1973), fixed point arithmetic utilises twos-complement truncation. Figure 5-40 shows zero input behaviour of DF1 using truncation and rounding for 24 bit fixed point arithmetic. The filter used is a 16 kHz notch filter (bandwidth of 7.5 Hz). In this case the truncated DF1 does produce an output of zero magnitude. The DF1 using rounding produces a limit cycle output at the pole frequency, with a peak magnitude of

approximately –82 dBFS. However scenarios were found where twos-complement truncation produces limit cycle behaviour. An example of this is shown in Figure 5-41, using a 19 kHz notch frequency filter (-3dB width equal to 7.5 Hz). The DF1 using rounding produces a limit cycle peak magnitude of approximately -102 dBFS. The DF1 using truncation produces a limit cycle peak magnitude of approximately -108 dBFS.

In 24 bit fixed point implementation using double precision pole paths limit cycle behaviour is eliminated. As the quantisation error noise source in the pole paths are completely removed. In (Higgins and Munson, 1984; Dattorro, 1988) first order error feedback is said to improve limit cycle suppression. In this work the use of first order error feedback is found to produce some limit cycle amplification, in some instances. Figure 5-40 shows the limit cycle behaviour for the 16 kHz notch filter. The DF1 using 24 bit fixed point truncation produces no limit cycle behaviour (converges to zero). Once error feedback is introduced limit cycle behaviour occurs. This is true for both first order error feedback schemes (dc and Nyquist frequency error feedback). For nominal input operating levels the noise performance of the 16 kHz notch filter would be improved through the use of Nyquist frequency error feedback. However the error feedback path promotes limit cycle behaviour for high frequency limit cycle scenarios. The dc zero error feedback scheme is worst (limit cycle peak magnitude of -78 dBFS) and produces a greater limit cycle magnitude than that caused by rounding. However the use of error feedback does not always amplify or promote limit cycle behaviour as shown in Figure 5-41.

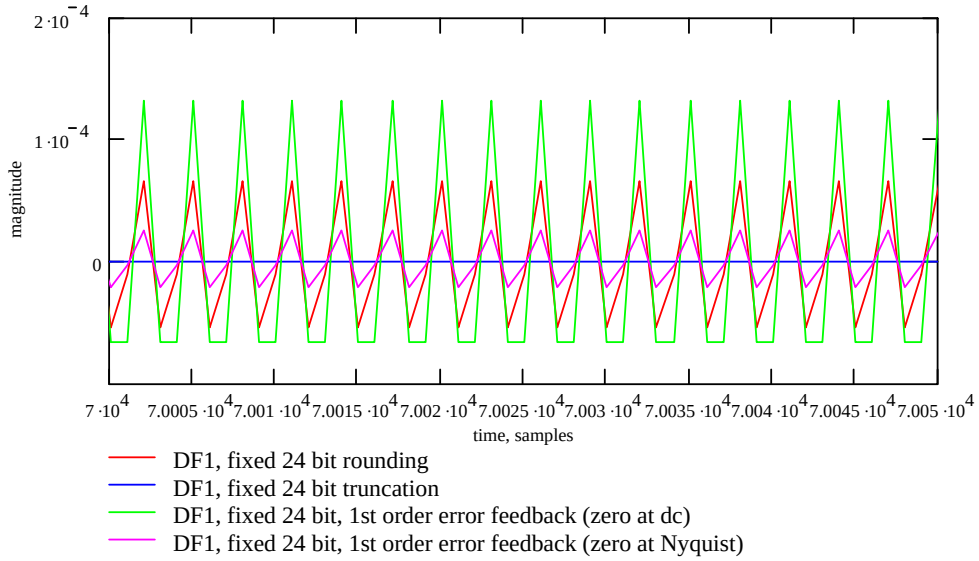

Figure 5-40 DF1 output, showing zero input limit cycle behaviour, using 24 bit fixed point arithmetic, using 16 kHz (7.5 Hz width) notch filter

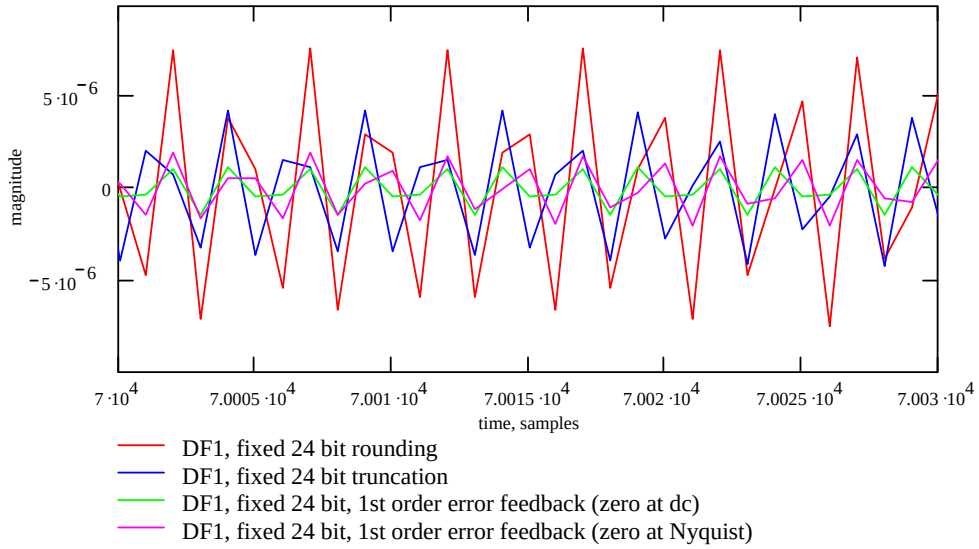

Figure 5-41 DF1 output, showing zero input limit cycle behaviour, using 24 bit fixed point arithmetic, using 19 kHz (7.5 Hz width) notch filter

Figure 5-42 shows an interesting example of limit cycle behaviour in DF2T in 24 bit fixed point implementation using the 19 kHz notch filter. For the tests conducted, DF2T produced less instances of zero input limit cycle behaviour for truncation and generally larger limit cycle magnitudes than DF1 for rounding. This could be explained by the extra quantisation points in the DF2T compared to the single quantisation point in the DF1

topology. Since rounding promotes limit cycle behaviour, it is conceivable that more rounding points in the topology, increases the probability of limit cycle behaviour. Since truncation can eliminate limit cycle behaviour, the more truncation points in the topology, may increase the probability of the state variables and output converging to zero.

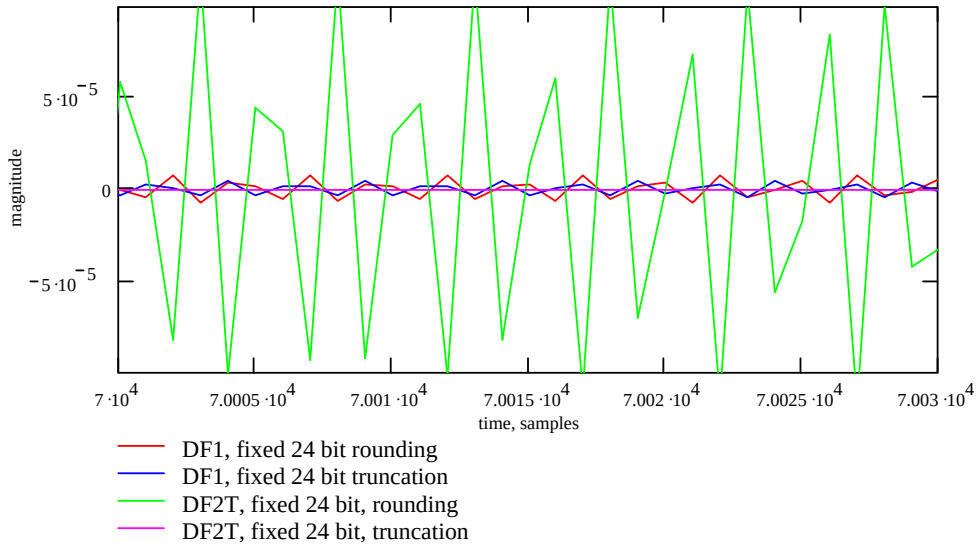

Figure 5-42 DF1 and DF2T output, showing zero input limit cycle behaviour, using 24 bit fixed point arithmetic, using 19 kHz (7.5 Hz width) notch filter

5.6.2 Zero input limit cycles in floating point arithmetic

Zero input limit cycle behaviour was examined using 32 bit sign magnitude floating point arithmetic. (Claasen et al, 1973) reports that zero input limit cycles are virtually non-existent using sign-magnitude truncation. Furthermore (Kaneko, 1973) shows that limit cycles occur in floating point arithmetic, using rounding. The DF1 was implemented using 32 bit sign magnitude floating point, for rounding and truncation. The DF1 using floating point sign magnitude truncation produced a continuously decaying frequency, in the pole frequency region. At the end of the test period the DF1 output had reached magnitudes in the region of –400 dBFS and was still decaying. This suggests the filter

output had not reached a state of steady cyclic behaviour at the end of the test period suggesting limit cycles were certainly below –400 dBFS.

However floating point sign magnitude rounding did produce limit cycles. Figure 5-43 shows a typical limit cycle using 32 bit floating point sign magnitude rounding, for a 16 kHz notch filter (bandwidth 7.5 Hz). The limit cycle is extremely small, approximately, -350 dBFS. Further tests using rounding did not produce limit cycles that were considerably larger than shown in this example.

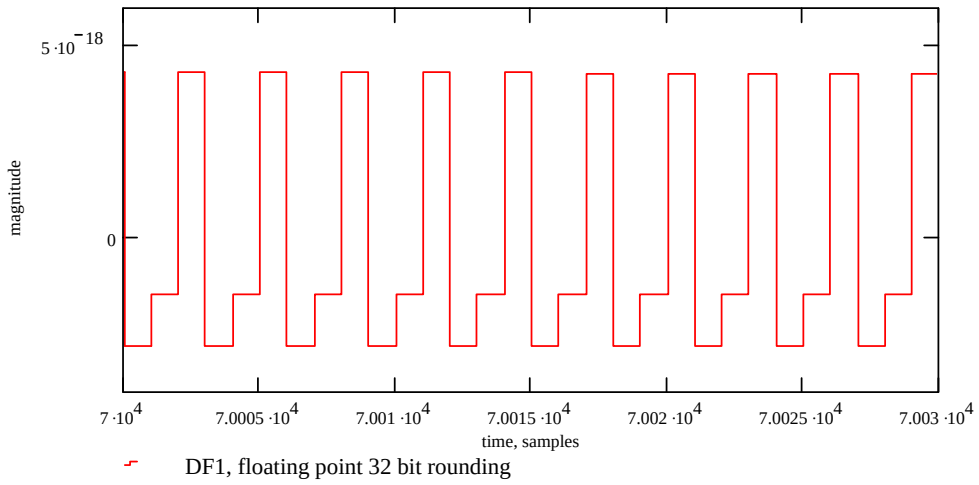

Figure 5-43 DF1 output, showing zero input limit cycle behaviour, for 32 bit floating point arithmetic using rounding, 16 kHz (7.5 Hz width) notch filter

5.7 High signal level non-linearity

The quantisation of low level signals produces non-linear correlated noise as examined in Section 5.5. High level non-linearity effects of finite wordlength are largely dependent on the correlation between the quantisation noise and input signal. Chapter 4, Section 4.2 examines fixed and floating point quantisation errors. It is shown that fixed point twos-complement truncation noise is less correlated with the source signal than for floating point truncation noise. Furthermore twos-complement coded floating point data produces odd and even order harmonic distortion products. Sign magnitude floating point

truncation produces odd order distortions with the largest distortion component at the input fundamental frequency, Figure 4-8.

Further consequences of the harmonic distortions caused by various quantisers can be performed using higher frequency sinusoids. Figure 5-44 shows the distortions components of basic fixed and floating point truncation quantisers using a 15000.3 Hz tone at –1dBFS, (15 kHz is a direct sub-multiple of the 48 kHz sampling frequency and produces heavily correlated quantisation noise). The sign magnitude floating point truncator produces harmonic distortion at the fundamental frequency and the fifth and seventh order harmonic, 75 and 105 kHz. At the sampling frequency of 48 kHz these harmonics are aliased to produce in band distortion components at 3 and 9 kHz. In addition to these harmonics, the twos-complement truncator produces noticeable even order harmonic distortion (second, forth and sixth). These even order components alias to produce in band distortion components at 6, 12 and 18 kHz. Despite the fixed point truncator producing a large dc offset and a higher noise floor, the harmonic distortion components generated by the floating point truncators are potentially more audible.

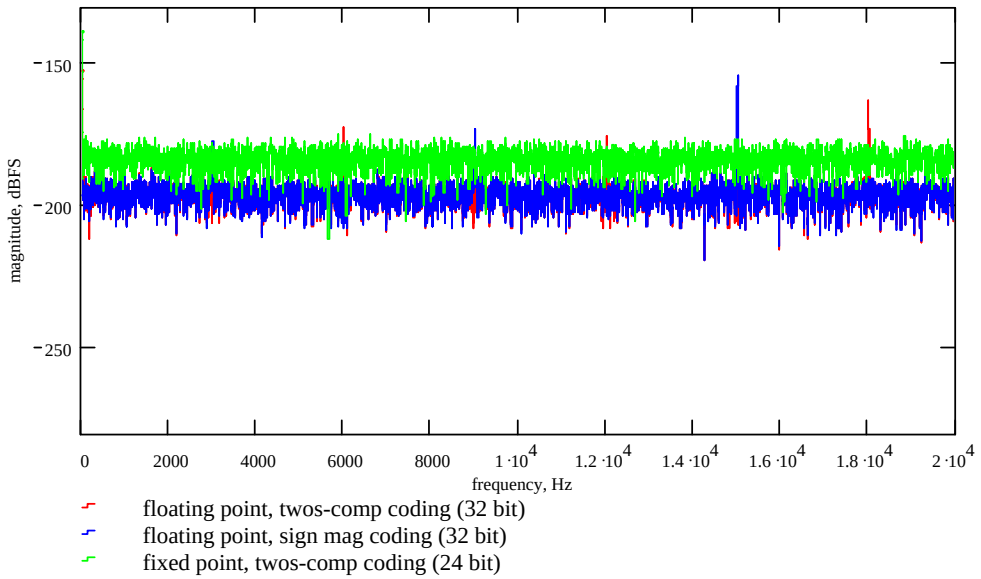

Figure 5-44 Distortion components generated by 24 bit fixed point and 32 bit floating point truncation. Operating on a 15000.3 Hz sinusoid at –1dBFS.

The evaluation of system non-linearity at high signal levels is conventionally measured through inter-modulation distortion (IMD) measurements, using multi-tone stimulus. Twin-tone stimulus of equal amplitude are commonly used to test bandwidth limited (digital) systems, (Metzler, 1993). Twin-tone digital systems with a limited bandwidth of 15 kHz have historically used 13 and 14 kHz tones. Modern standards (Audio Engineering Society, 1998) specify 18 and 20 kHz as suitable tones to test higher bandwidth systems, up to 24 kHz. The choice of twin tone frequencies is complex. The IMD test attempts to view IMD distortion products and not aliased harmonics or sampling frequency correlated distortion components. If a frequency is chosen that is a direct sub-multiple of the sampling frequency then the resulting quantisation noise will always be correlated with the input. Also the use of certain frequencies will produce high order harmonics aliased into the base band. Two sinusoidal frequencies, 14955 and 15952 Hz, were found to produce relatively flat (uncorrelated) truncation noise with no noticeable aliased harmonic distortions components which could be mistaken for IMD products. These two tones at an individual operating level of –6 dBFS were summed and used as input stimulus for fixed and floating point quantisers, Figure 5-45. Inter-modulation products, 1 and 2 kHz, are clearly visible for the twos-complement 32 bit floating point truncator. The 1 kHz distortion product is a second order IMD product. The 2 kHz distortion product is a fourth order IMD product.

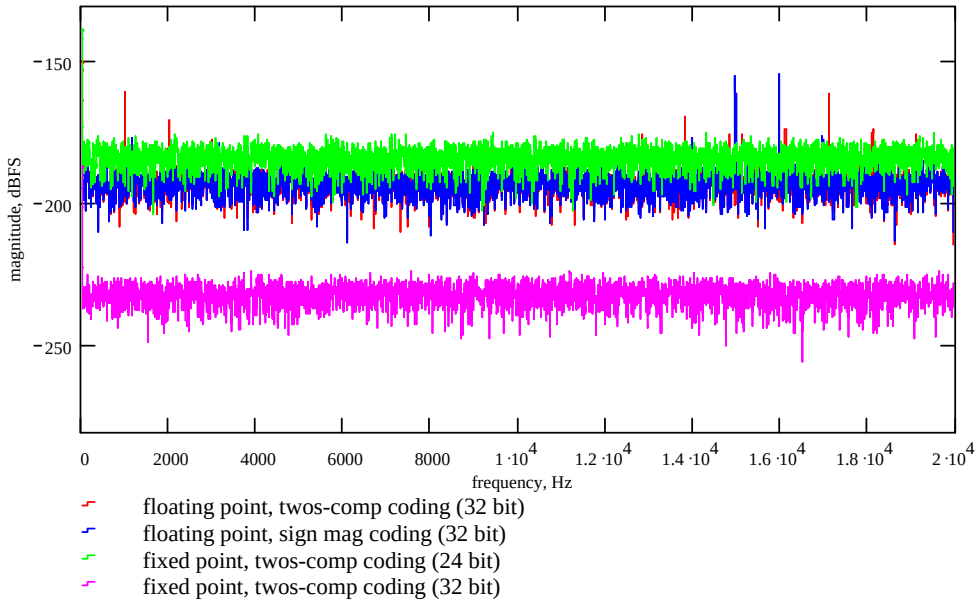

Figure 5-45 Inter-modulation distortion products for fixed and floating point truncation, using twin tones at 14955 Hz and 15952 Hz, each at –6 dBFS.

Figure 5-46 shows the output spectra for DF1 twin tone test, using fixed and floating point arithmetic, for the 20 Hz notch filter. Even order (second and forth) IMD products, from twos-complement floating point truncation, are visibly above the noise floors for each of the other implementations. Figure 5-47 shows the state-space topology output spectrum under twin tone excitation, using the 20 Hz notch filter. Despite its excellent noise floor, IMD products for twos-complement floating point truncation are visible.

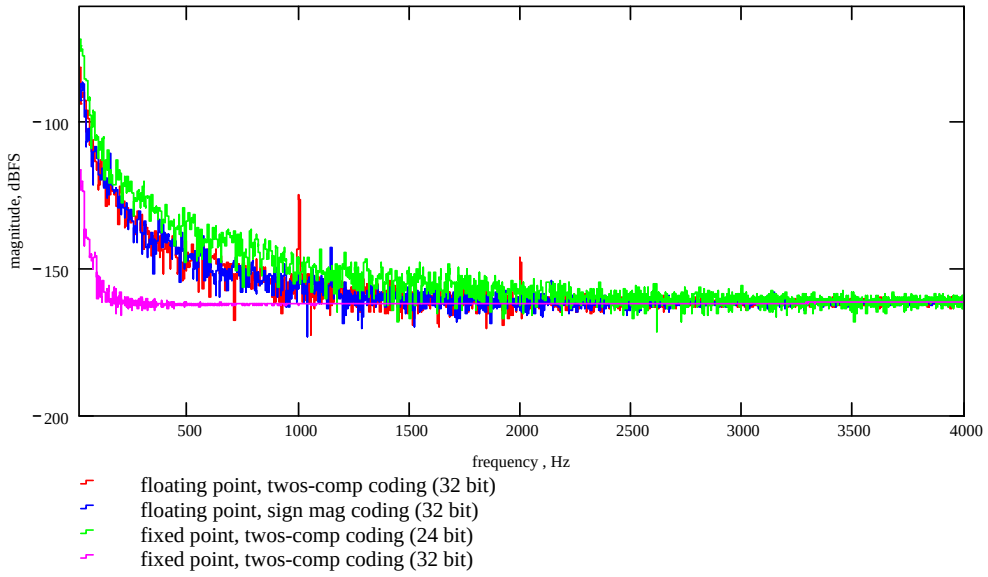

Figure 5-46 DF1 inter-modulation distortion products for fixed and floating point truncation, using twin tones at 14955 Hz and 15952 Hz, each at –6 dBFS, 20 Hz notch filter.

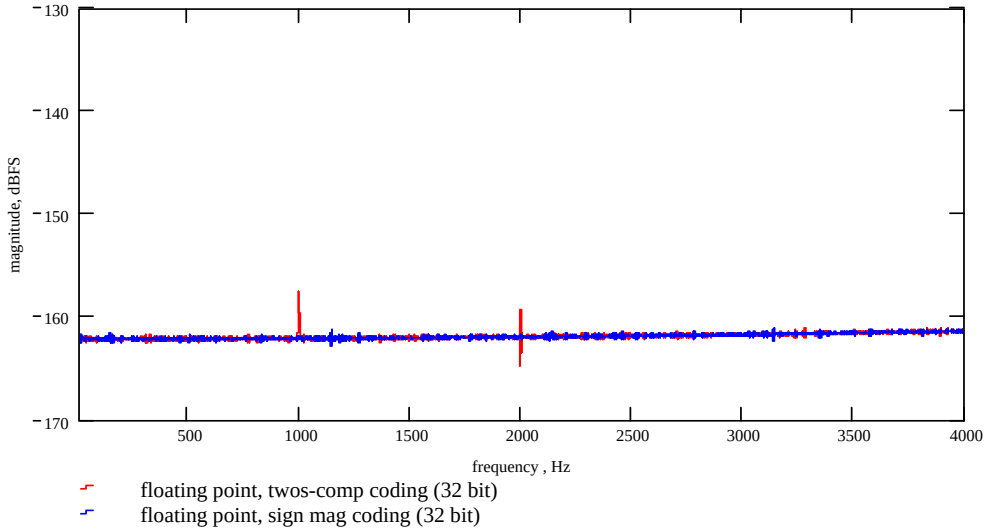

Figure 5-47 State space topology inter-modulation distortion products for floating point truncation, using twin tones at 14955 Hz and 15952 Hz, each at –6 dBFS, 20 Hz notch filter.

5.8 Effects of higher signal sampling frequencies

Finite wordlength filter topology emulation has been conducted at the higher signal sampling frequencies of 96 and 192 kHz. This section compares the resulting residual noise products at these sampling frequencies with the residual noise produced by topologies operating at 48 kHz (described in Section 5.4). As shown in Figure 5-2 there is

an increase of 12 dB in the pole transfer function gain for a doubling in sampling frequency (48 kHz to 96 kHz). Figure 5-48 shows the DF1 output spectra using 24 bit fixed point arithmetic, operating at a sampling frequency of 96 kHz. Figure 5-49 shows the DF1 output spectra using 24 bit double precision and 32 bit fixed point arithmetic, operating at a sampling frequency of 96 kHz. All output spectra, excluding DF1 using dc error feedback, show the expected 12 dB increase in noise products compared to those shown for the 48 kHz sampling frequency, Figure 5-10 and Figure 5-12. All DF1 floating point implementations at a sampling frequency of 96 kHz, produce increases of 12 dB in their noise products. Table 5-1 shows the THD+N figures for all DF1 implementations operating at 48 and 96 kHz, for the 20 Hz notch filter.

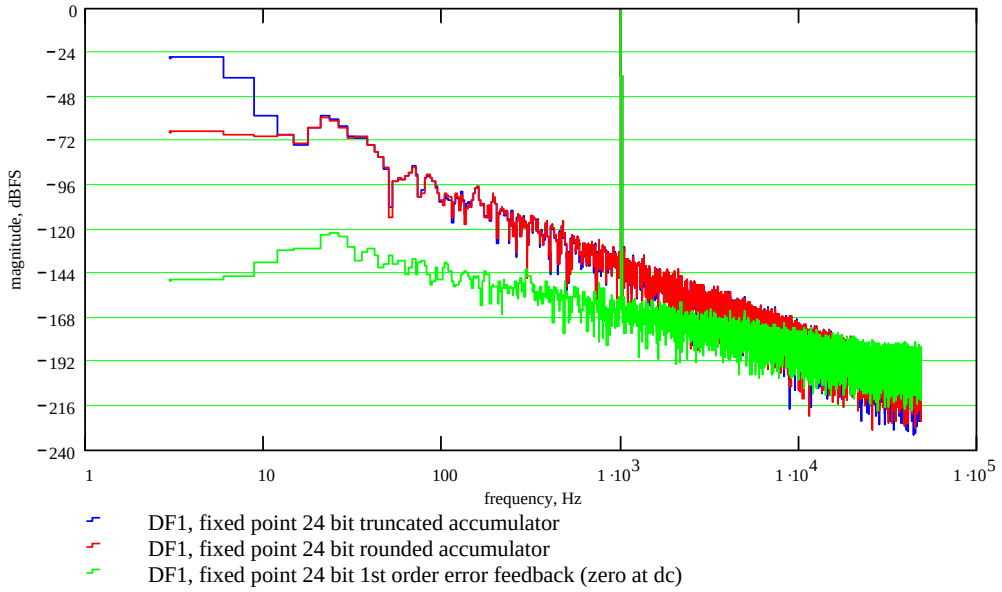

Figure 5-48 DF1 output spectrum for 24 bit fixed point implementations operating at a sampling frequency of 96 kHz, 20 Hz notch filter, 7.5 Hz bandwidth.

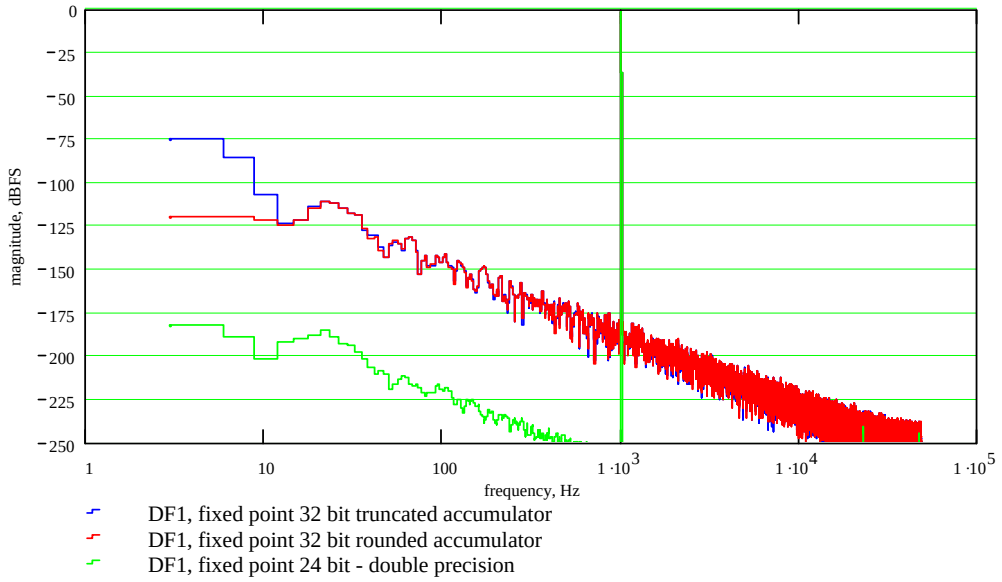

Figure 5-49 DF1 output spectrum for 24 bit double precision and 32 bit fixed point implementations operating at a sampling frequency of 96 kHz, 20 Hz notch filter, 7.5 Hz bandwidth.

DF1 implementation, 20 Hz notch filter (width 7.5 Hz)	48 kHz THD+N RMS figure (dBFS)	96 kHz THD+N RMS figure (dBFS)
Fixed point 24 bit truncation	-41.5	- 29.4
Fixed point 24 bit rounding	-70.3	- 59.5
Fixed point 32 bit truncation	-89.6	- 77.6
Fixed point 32 bit rounding	-120.9	- 112.6
Fixed point 24 bit double precision	-193.2	- 181.7
Fixed point 24 bit, dc error feedback	-123.8	- 121.9
Floating point 32 bit sign magnitude truncation	-80.5	- 67.5
Floating point 32 bit sign magnitude rounding	-79.0	- 66.6
Floating point 32 bit twos-complement truncation	-48.9	- 37.8
Floating point 32 bit twos-complement rounding	-79.4	- 66.7
Floating point 40 bit sign magnitude truncation	-114.4	-101.9
Floating point 40 bit sign magnitude rounding	-115.9	-103.2
Floating point 40 bit twos-complement truncation	-82.4	-71.2
Floating point 40 bit twos-complement rounding	-97.6	-85.2

Table 5-1 Total harmonic distortion plus noise figures (RMS) for DF1 implementations at 48 and 96 kHz.

Figure 5-50 shows the output spectra for DF2 and DF1T operating at a sampling frequency of 96 kHz, using a 32 bit floating point sign magnitude truncation implementation. The filter used was the 30 Hz bell boost example. Comparison with the equivalent 48 kHz measurements, Figure 5-22, shows approximately an increase of 12 dB in the noise components for the higher sampled filters. Figure 5-51 shows the output spectra for the coupled forms, implemented in floating point at a sampling frequency of 96

kHz. Comparison with the 48 kHz measurements, Figure 5-28, shows that the increase in noise components is less than 12 dB. Coupled form THD+N figures for these outputs, are given below. The relative THD+N increases, for a sampling frequency of 96 kHz compared to the 48 kHz, are shown in brackets.

Gold-Rader	- 94.9 dBFS (3.6 dBFS) ,
Kingsbury	- 99.9 dBFS (5.7 dBFS) ,
Zölzer	- 99.9 dBFS (7.0 dBFS).

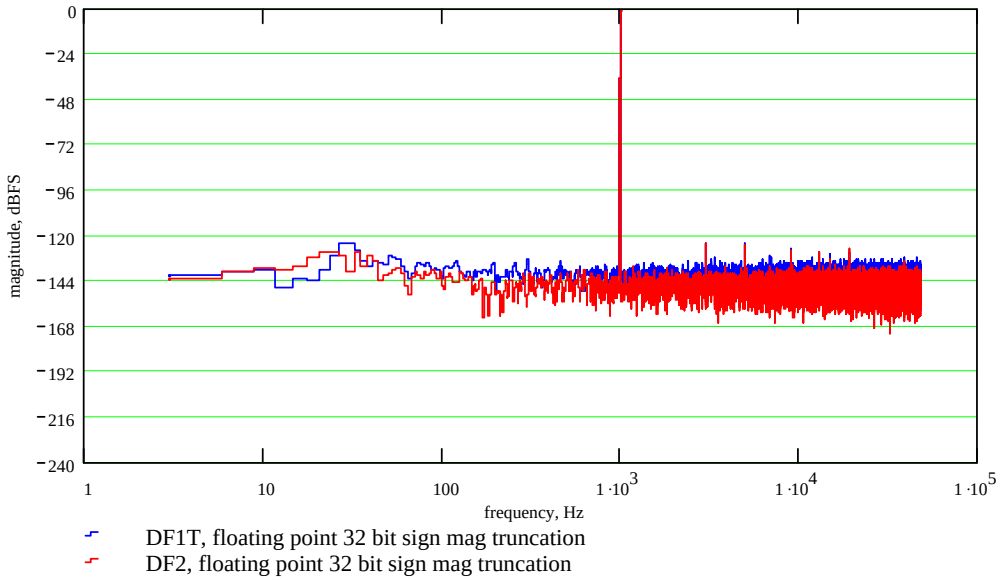

Figure 5-50 DF2 and DF1T output spectrum for 32 bit floating point implementations operating at a sampling frequency of 96 kHz, 30 Hz bell filter, Q of 3, gain 18 dB.

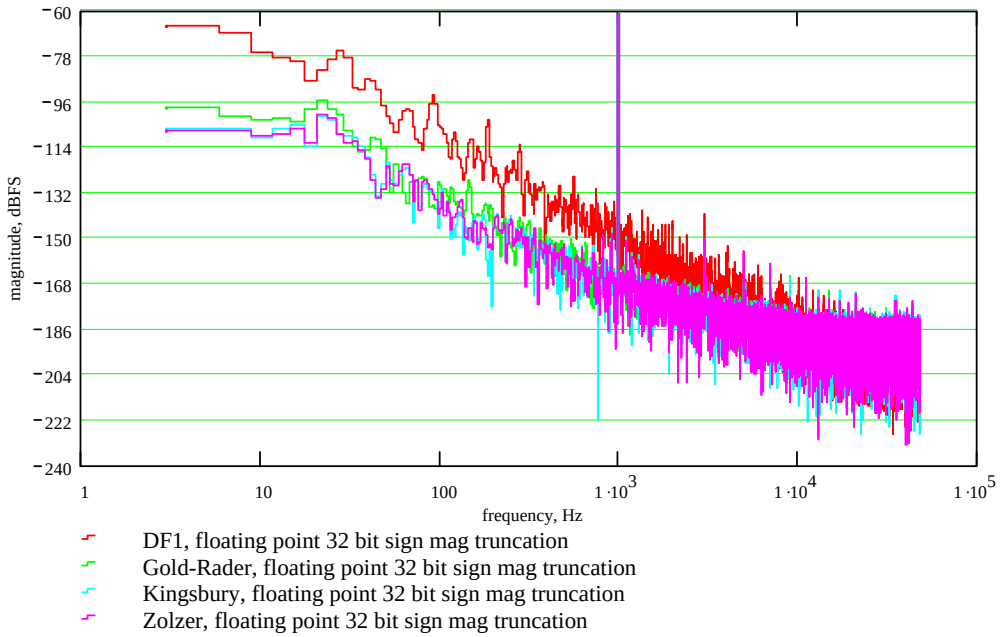

Figure 5-51 Coupled forms and DF1 output spectrum for 32 bit floating point implementations operating at a sampling frequency of 96 kHz, 20 Hz notch filter, 7.5 Hz bandwidth.

Figure 5-52 shows the output spectra for the ladder topologies implementing the 30 Hz bell filter at a 96 kHz sampling frequency. Comparison of Figure 5-52 and Figure 5-32 suggests the 96 kHz implementation produces a noise increase of 3 dB in the tuned frequency region. The rest of the noise spectrum is unaffected by the change of sampling frequency. Consequently the increase in THD+N figures for 96 kHz operation are an increase of approximately 1 dB for both the Massie and Moorer ladder implementations. However the lattice implementation at a sampling frequency of 96 kHz produces a more pronounced noise increase in the tuned frequency region, compare Figure 5-53 and Figure 5-33. The resulting THD+N figure for Lattice (Massie) operating at 96 kHz is –122.3 dBFS. This is 10 dB higher than the figure using a 48 kHz sampling frequency.

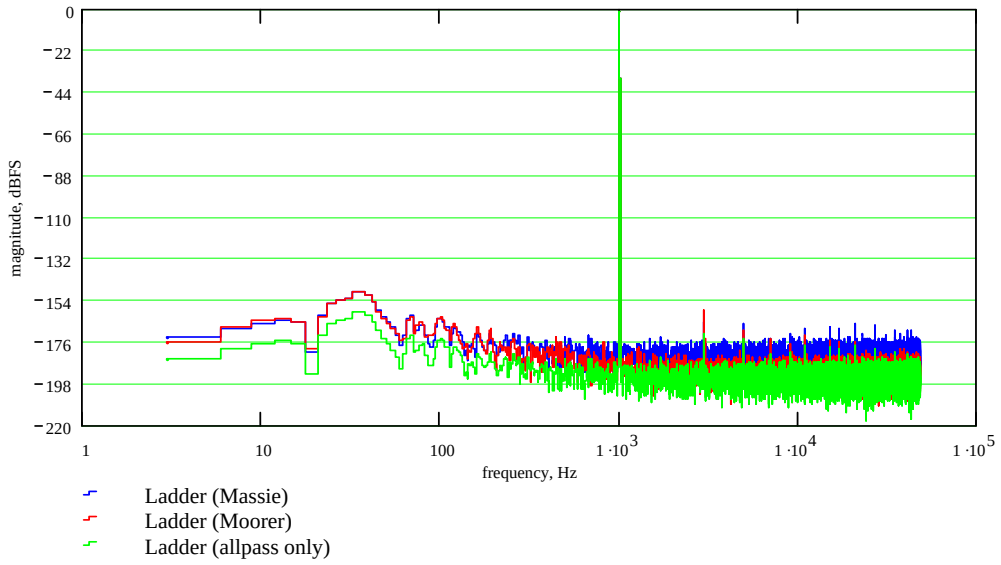

Figure 5-52 Output spectra of ladder structures, using 32 bit floating point implementations operating at a sampling frequency of 96 kHz, 30 Hz bell filter, Q of 3 gain 18 dB.

Figure 5-54 shows the output spectrum of the state-space structure, operating at a sampling frequency of 96 kHz, realising the 20 Hz notch filter example. Comparison with Figure 5-30 shows that the increase in sampling frequency has no noticeable effects on the noise components in the output spectrum. This is substantiated by the THD+N figures. For the three traces shown in the figures, for 32 bit, 40 bit truncation and 32 bit rounding, the THD+N figures produced at a sampling frequency of 96 kHz are within 0.5 dB of that produced at 48 kHz. Note the THD+N figures for the 48 kHz sampling frequency are given in Section 5.4.6).

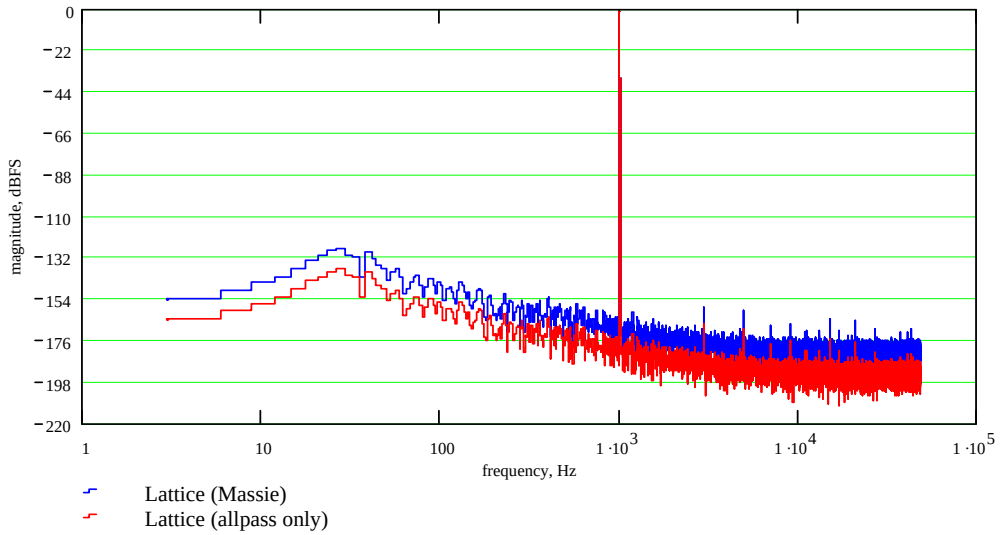

Figure 5-53 Output spectra of lattice structures, using 32 bit floating point implementations operating at a sampling frequency of 96 kHz, 30 Hz bell filter, Q of 3 gain 18 dB.

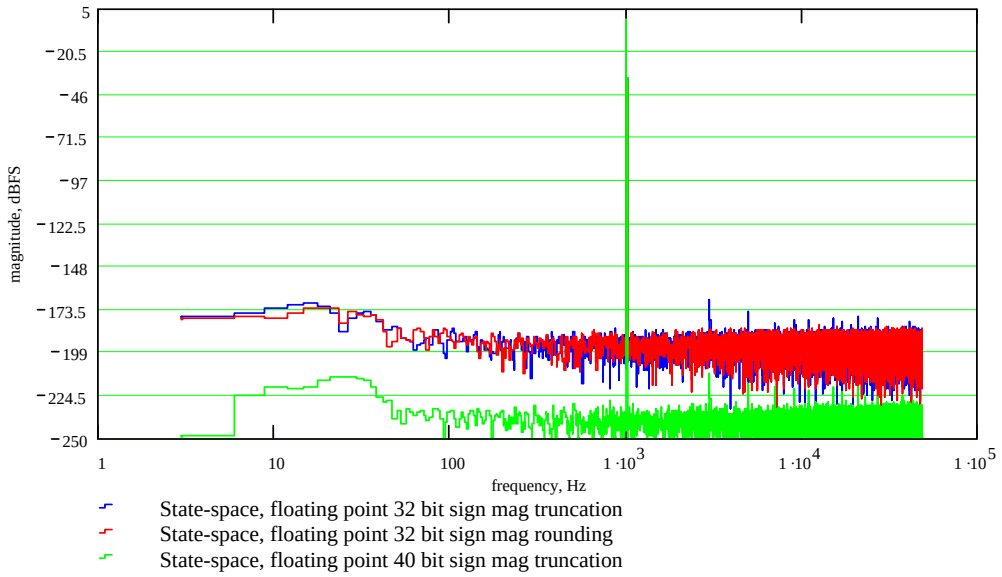

Figure 5-54 Output spectra of state-space structures, using 32 bit floating point implementations operating at a sampling frequency of 96 kHz, 30 Hz bell filter, Q of 3 gain 18 dB.

5.9 Discussion

Direct form ‘zero before pole’ topologies produce poor noise performance when used to realise low frequency, high Q, notch filters. The dc offsets in truncation noise in twos-complement fixed and floating point arithmetic are greatly amplified in ‘zero before pole’ topologies. This has the effect of producing large dc noise in DF1 and DF2T topologies. However, floating point sign magnitude truncation produces no dc offset, improving its dc noise performance. For DF1 and DF2T, 32 bit floating point reduces noise components by 9 dB than that produced by 24 bit fixed point. This is significant because both use 23 fractional bits. DF1 using fixed point 32 bit produces superior noise characteristics to 32 bit floating point. DF1 floating point 40 bit extended precision produces inferior noise performance to that of 32 and 24 bit (double precision) fixed point. Floating point extended precision produces comparably poor noise performance since every multiplication and a majority of the accumulations introduce quantisation. In fixed point implementation DF2T is inferior to DF1 - due to its multiple quantisation points. However in floating point implementation the DF2T is similar to DF1.

Quantisation error transfer functions for ‘pole before zero’ topologies contain poles and zeros. Thus, filters with gain (boost settings) produce worse case noise characteristics. Consequently under floating point implementation DF2 is a low noise topology. However, harmonic distortion components in floating point truncation schemes are subsequently noticeable and the use of rounding produces better THD+N figures. DF1T has multiple quantisation points in single precision implementation, producing an increase in residual noise compared to the DF2. However, the DF1T in 40 bit extended precision produces minimal pole path accumulation quantisation, since it sums the large magnitude ‘pole paths’ prior to summing with the unity bound input. Thus, if the DF2 pole paths are summed first, the extended precision performance of DF2 is similar to DF1T.

Coupled form floating point implementations of low frequency tuned filters improve on DF1 noise performance by 20 dB. For high frequency tuned filters the DF1 produces a 6 dB improvement over the coupled forms. This agrees with the unscaled theoretical noise model in (Zölzer, 1991). The state-space topology produces excellent quantisation noise performance, under floating point rounding. Using truncation reduces its noise performance by 6 dB (due to harmonic distortion). The state-space topology implemented in 40 bit extended precision compares closely in noise performance to DF1 using 24 bit fixed point double precision. The Cabot structure did not produce similar noise performance to the state-space topology as reported by Cabot, (1992). In single precision floating point implementation the noise performance was found to be superior to DF1.

The Moorer ladder implementation produces less quantisation noise than the Massie ladder implementation by approximately, 9 dB. The lattice (Massie) produces more pronounced noise in the frequency boost region, producing a THD+N figure 7 dB worse than the ladder (Massie).

Using low level stimuli, fixed point filters were shown to produce low noise modulation. Floating point filters were shown to produce noise modulation, due to the quantisation noise being proportional to the signal level. DF2T in fixed point implementation was found to produce broad band quantisation noise at low level operation. Since its multiple quantisation points produce an uncorrelated overall noise characteristic. Applying dither to the accumulator (prior to the single quantiser) of the DF1 was explored. Applying dither to the DF1 in floating point implementations produced a much higher noise floor, comparable to the fixed point DF1. For the fixed point DF1, the application of dither did prevent all correlated distortions with the input (harmonic and noise modulation). Although the DF2T appeared to be less sensitive to correlated distortion than the undithered DF1, the dithered DF1 produced a lower noise floor than DF2T.

Floating point sign magnitude truncation was shown to eliminate limit cycle behaviour. Floating point rounding produced limit cycles of negligible magnitudes. Fixed point twos-complement truncation produces less instances of limit cycle behaviour than fixed point rounding. It was found that DF1 dc and Nyquist frequency error feedback schemes can promote high frequency limit cycles. Using truncation, DF2T produces less instances of limit cycle behaviour than DF1. Using rounding, DF2T produces more instances of limit cycle behaviour than DF1.

Floating point 32 bit truncation is shown to produce aliased harmonic distortion components which are larger in magnitude than the noise floor produced by a fixed point 24 bit truncator. A filter using 32 bit floating point twos-complement truncation is found to produce even order IMD products that are above the filters noise floor and of the noise floor of a 24 bit fixed point filter.

Direct Form noise figures increase by 12 dB for a doubling in sampling frequency. However the couple forms increase in noise figure is in the region of 6 dB. A doubling in sampling frequency in the lattice (Massie) topology is found to produce a noise increase of 10 dB. The ladder (Massie) and state-space topologies THD+N figures increase by less than 1 dB for a doubling of sampling frequency.

6 Topology behaviour during coefficient update

6.1 Introduction

Discrete-time varying filters are capable of producing transient distortions (disturbances) at their output during real-time coefficient updates. Parameter and coefficient interpolation techniques exist that reduce disturbance, at the cost of computational load. Mourjopoulos et al (1990) and Hanna (1994) attempt to minimise interpolation rates, without introducing audible distortion. However this previous work has not provided an understanding of the disturbance mechanisms in each of the filter topologies. The previous work does not specify current and target frequency response and input stimulus for worst case disturbance behaviour. Furthermore no study, known to the author, has examined the effects of higher signal sampling frequencies on signal disturbance.

This work emulates filter topologies under various coefficient update scenarios, using various input stimuli, in the Mathcad environment. Initially this work investigates the disturbance behaviour of the various filter topologies under a step state change - using no interpolation. The effects of current and target frequency response and input stimulus on disturbance behaviour at the filter output are examined. The work studies the disturbance effects of existing parameter and coefficient interpolation schemes. The effects of sub-sampled interpolator rates and the disturbance effects of interpolation implemented in finite wordlength arithmetic are investigated. Signal disturbance effects in filters operating at higher signal sampling frequencies are studied.

6.2 Time domain test environment for the observation of filter disturbance

The Mathcad environment was used as a test environment to implement discrete-time varying filter topologies under various filter state changes. No arithmetic quantisation functions were applied to the coefficients, state variables or arithmetic operations in the filter topology implementation. This ensured that the numerical representation of the coefficients, the operating arithmetic and state variables was as high as possible.

The discrete-time varying filter environment permits any coefficient to change every sample period. This was achieved through the use of one-dimensional arrays, for each coefficient in each topology. Each array entry represents the value of that coefficient in a sample instance of the test scenario. The coefficient arrays were pre-calculated. Snap-shot filter state changes are implemented by changing a coefficient set at a particular instance in the array set. Gradual interpolated coefficient data changes were implemented, using the coefficient and parameter interpolation schemes discussed in Chapter 2, Section 2.6. The time-varying filter topology implementations and interpolation algorithms used in this work are given in Appendix D.

It was found through informal listening tests, prior to the investigation, that many controlled filter changes, from one state to another did not always generate the same disturbance when the filter's state changed back to the initial state. Therefore each coefficient set contained two state changes, for example, state A to state B and then back from state B to state A. Sufficient time periods were given between the state changes to allow the filter to reach steady state as to not produce confusing disturbances from the previous state change.

Figure 6-1 shows the time domain representation of the frequency response state changes used through-out the tests. On the 2520^{th} sample in time the frequency response state

changes from state A to B. On the 12600th sample in time the frequency response state changes from state B to A.

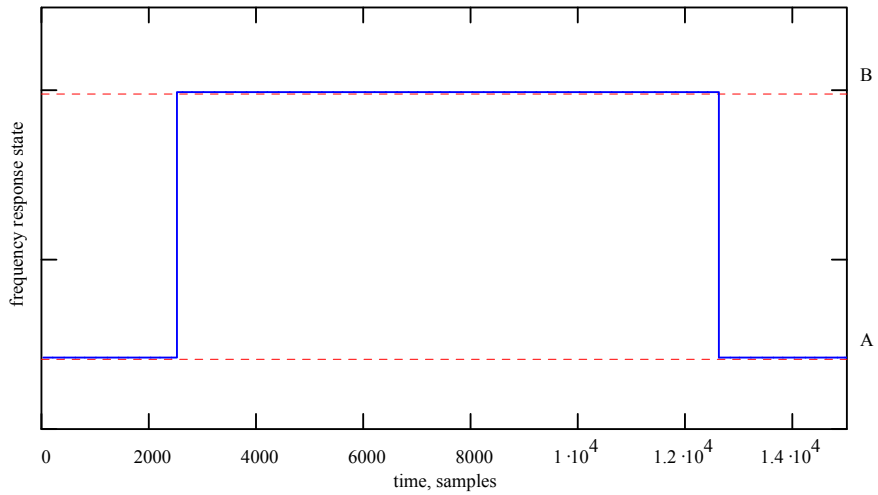

Figure 6-1 Time domain representation of the frequency response state changes A and B.

6.2.1 Coefficient sensitivity issues

The effects of finite wordlength coefficients on frequency response, vary for each filter topology, Chapter 2 Section 2.4.3. If quantised coefficients are used in the disturbance tests described in this chapter, then each filter topology will potentially produce a different magnitude frequency response, making comparison difficult. Furthermore finite wordlength coefficients typically produce a gain deviation at dc in the magnitude response. Therefore step changes in level can occur during state changes. This represents a step change distortion for filters designed to have unity gain at dc. These step changes cause ringing, Figure 6-2. This complicates the study of the actual disturbance effect due to the filter state change. In order to avoid these additional disturbance effects, the coefficients used through-out the tests are not intentionally quantised. However the coefficients are subjected to the intrinsic finite wordlength of Mathcad (Chapter 5, Section 5.2.5).

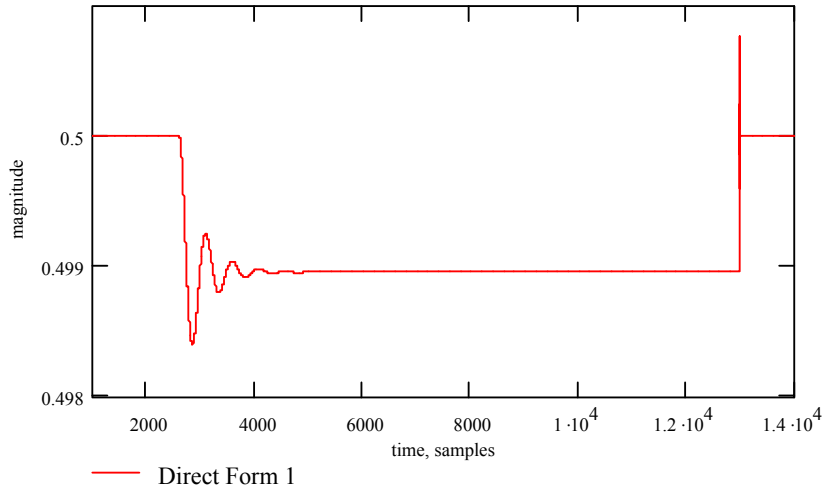

Figure 6-2 DF1 disturbance response to a change in the tuned frequency of a unity gain filter - flat frequency response using coefficients quantised to 24 bits, dc excitation of 0.5.

The effects of coefficient finite wordlength in Mathcad may still produce magnitude response distortions, for different filter topology implementations. Therefore the step response of various filter topologies were assessed with the intention of revealing any differences in the topologies response due the finite wordlength of Mathcad. For this test a fixed filter response was used (Butterworth high pass filter tuned to a frequency of 40 Hz).

The coupled forms (Gold-Rader, Kingsbury and Zölzer) pole responses, under step input excitation were compared to the direct form step input pole response. The pole response difference between the direct form and the Gold-Rader and Zölzer structures are shown in Figure 6-3, Figure 6-4 respectively. Note the Kingsbury and Zölzer structures produce extremely similar pole responses. It is clear that the difference in pole response between the direct forms and coupled forms is extremely small. There are no coefficient sensitivity issues in the zero transfer function implementations since the coupled forms and direct form use the same implementation. The overall response difference of the Zölzer and the DF1, under step input excitation, is shown in Figure 6-5.

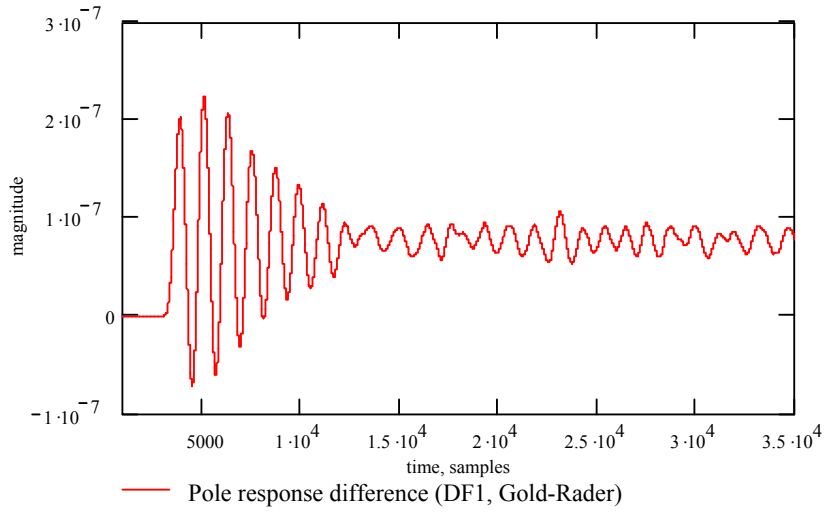

Figure 6-3 Difference in pole response of the DF1 and Gold-Rader implementations. For a 40 Hz high pass filter, Butterworth response.

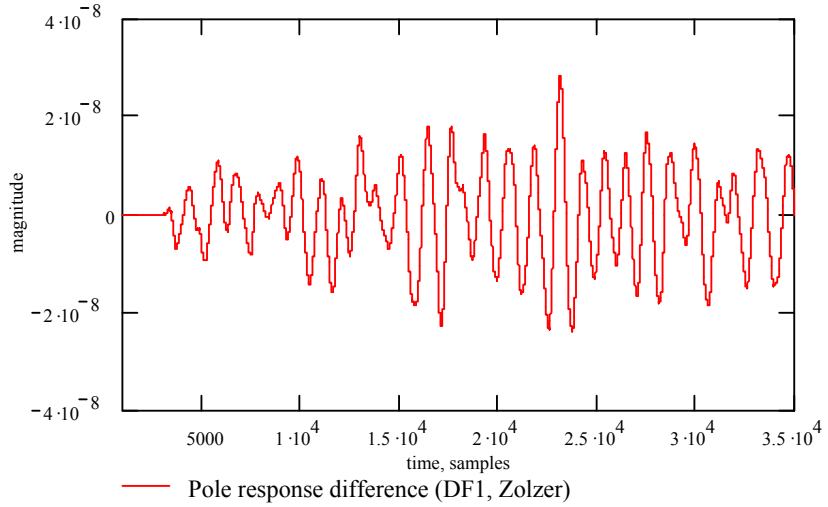

Figure 6-4 Difference in pole response of the DF1 and Zolzer implementations. For a 40 Hz high pass filter, Butterworth response.

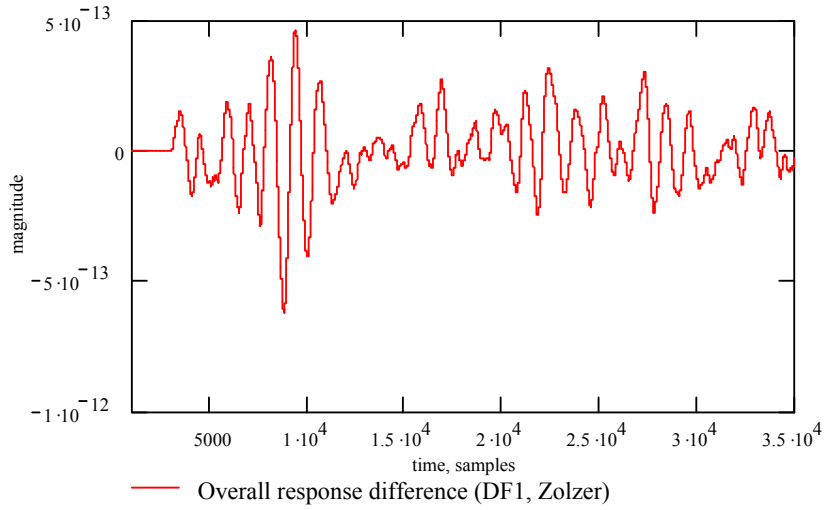

Figure 6-5 Difference in overall response of the DF1 and Zolzer implementations. For a 40 Hz high pass filter, Butterworth response.

The overall response difference between the Cabot and DF1 topologies, for the 40 Hz high pass filter, under a step input excitation, is shown in Figure 6-6. It is clear that the response difference is extremely small. Similar tests were conducted for the ladder, lattice and state-space structures. The overall response differences were of the same magnitude as the examples shown. This suggests that the coefficient finite wordlength constraints of Mathcad produce negligible changes in response for the various filter topologies.

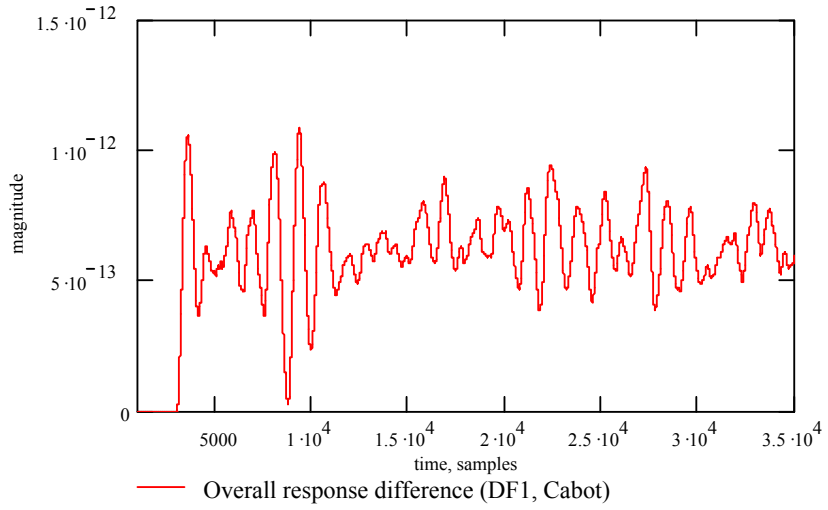

Figure 6-6 Difference in overall response of the DF1 and Cabot implementations. For a 40 Hz high pass filter, Butterworth response.

6.2.2 Input source

Mourjopoulos et al (1990) states that filter disturbance magnitudes are dependent on the type of input stimulus and that sinusoidal inputs produce the most audible disturbances. However, white noise excitation naturally exercises the filter system in all spectral regions and is the preferred source input in (Moorer, 1999). The use of a sinusoidal input stimulus can result in inconsistent results. For example, minimal disturbances can result from the trough of the sinusoid coinciding with the actual filter state change. This is a particular problem for low frequency sinusoids, where magnitude minima span many samples. However the benefits of sinusoidal excitations are that a precise frequency can be selected. This is useful for defining particular frequencies in which worst case disturbances exist, for example, in the maximum gain region of the pole transfer function. Care was taken in placing the filter state change event in a sample region where peak magnitudes occurred in the sinusoidal excitation. Through-out the work it has been found that a dc input excitation is useful in the examination of some filter disturbance scenarios. It is important to note that all of the disturbance effects shown using dc input also occur using low frequency sinusoids. However it was found impractical to use low frequency sinusoids owing to the considerable intervals of samples in the trough of the input excitation which create inconsistent results. It is however obvious that high level dc is not a practical audio input signal. Signal disturbances are proportional to the level of the input stimulus. The dc input excitation level used through-out this work is a magnitude of 0.5. All sinusoidal excitations levels used in this work are –6 dBFS.

6.3 Magnitude frequency responses for disturbance analysis

There are many filter type changes and filter parameter changes that generate signal disturbance at the output of filters. Through empirical tests and the examination of what

critical filter types and parameter changes may occur practically, the following filter change scenarios have been identified.

6.3.1 Filter state change 'Scenario 1', bell filter frequency parameter change (unity gain)

The filter state change introduced in 'Scenario 1' utilises one type of audio filter, the bell filter. Both states have a gain setting of zero dB, resulting in both frequency responses being flat. This scenario provides a state change test that does not, for any input, result in any steady state gain change. The filter parameter settings for the two states A and B are,

State A : Fc = 10 kHz, Q = 8, Gain = 0 dB

State B : Fc = 100 Hz, Q = 8, Gain = 0 dB.

The pole transfer functions for the two states are shown in Figure 6-7. The zero transfer functions are shown in Figure 6-8.

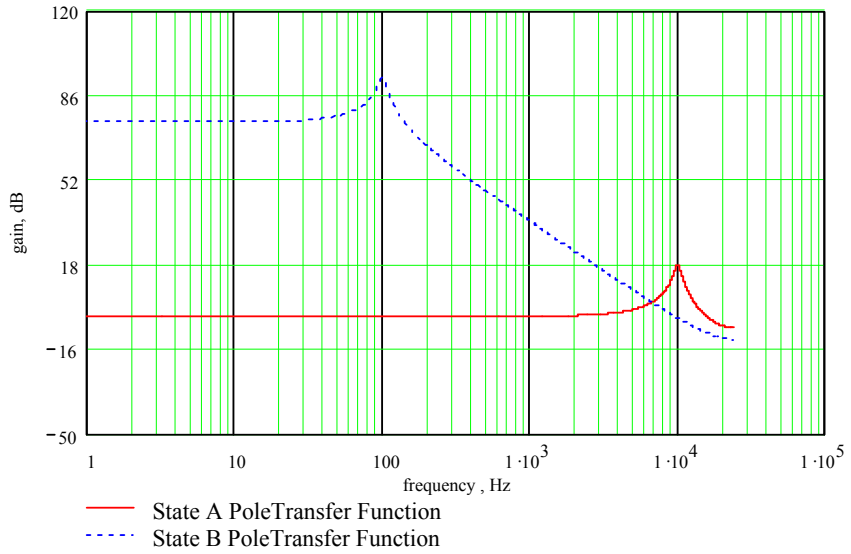

Figure 6-7 Pole transfer functions for filter response change Scenario 1, states A and B.

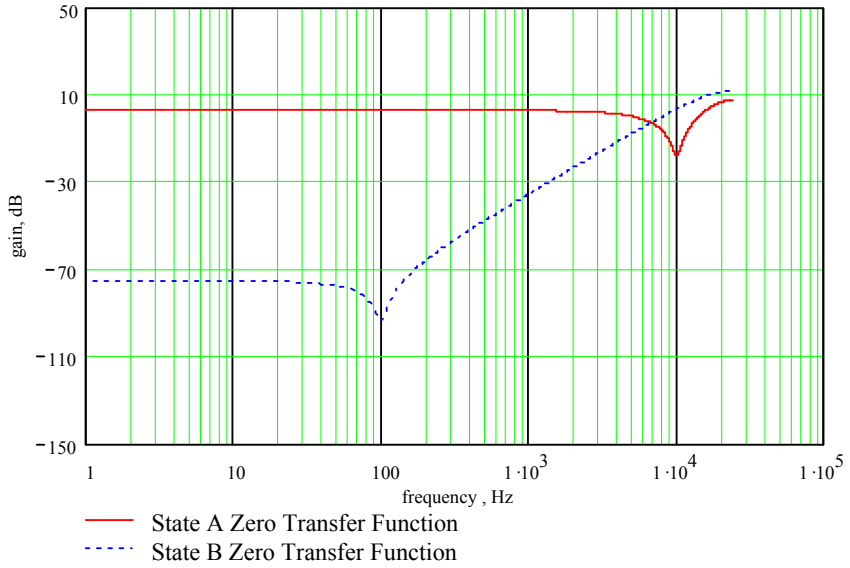

Figure 6-8 Zero transfer functions for filter response change Scenario 1, states A and B.

6.3.2 Filter state change 'Scenario 2', bell filter frequency parameter change (–10 dB gain)

The filter state change introduced in 'Scenario 2' utilises one type of audio filter, the bell filter. Both states have a gain setting of -10 dB and vary in tuned frequency. The filter parameter settings for the two states A and B are,

State A : Fc = 10 kHz, Q = 8, Gain = -10dB

State B : Fc = 100 Hz, Q = 8, Gain = -10dB

This test scenario simulates a common single frequency parameter control sweep. The overall magnitude responses, for the two states are shown in Figure 6-9. The pole transfer functions for the two states are shown in Figure 6-10. The zero transfer functions are shown in Figure 6-11.

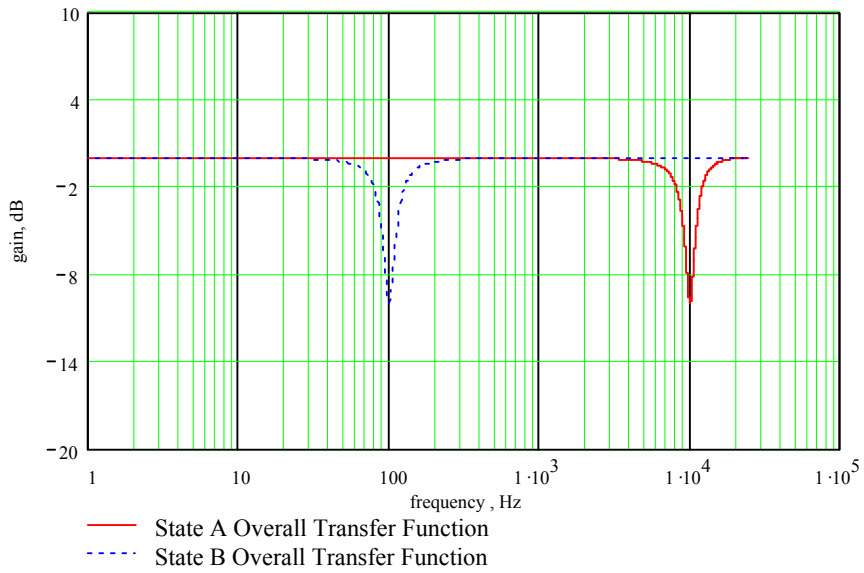

Figure 6-9 Overall magnitude response changes for filter response change Scenario 2, states A and B.

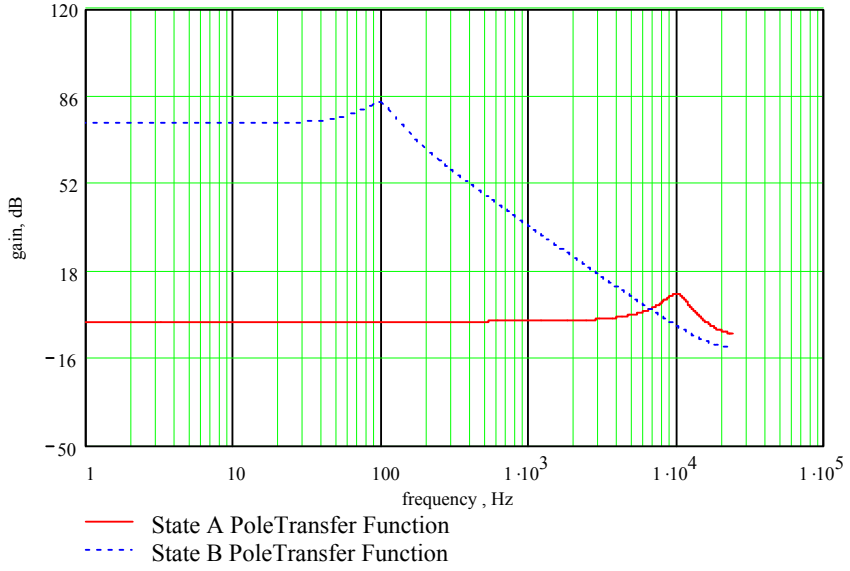

Figure 6-10 Pole transfer functions for filter response change Scenario 2, states A and B.

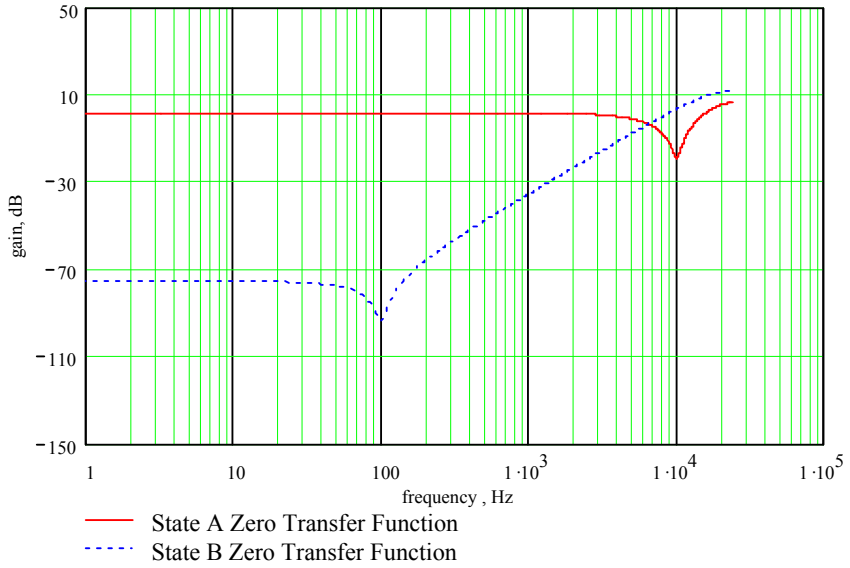

Figure 6-11 Zero transfer functions for filter response change Scenario 2, states A and B.

6.3.3 Filter state change 'Scenario 3', low pass filter frequency parameter change

The filter state change introduced in 'Scenario 3' utilises one type of audio filter, the 2nd order low pass filter (Butterworth response). The –3 dB cut-off frequency is changed from 10 kHz to 100 Hz. The filter parameter settings for the two states A and B are,

State A : Fc = 10 kHz, Q = 0.7071

State B : Fc = 100 Hz, Q = 0.7071.

This test scenario produces a large gain change between states for high frequency input signals. The overall magnitude responses, for the two states are shown in Figure 6-12. The pole transfer functions for the two states are shown in Figure 6-13. The zero transfer functions are shown in Figure 6-14.

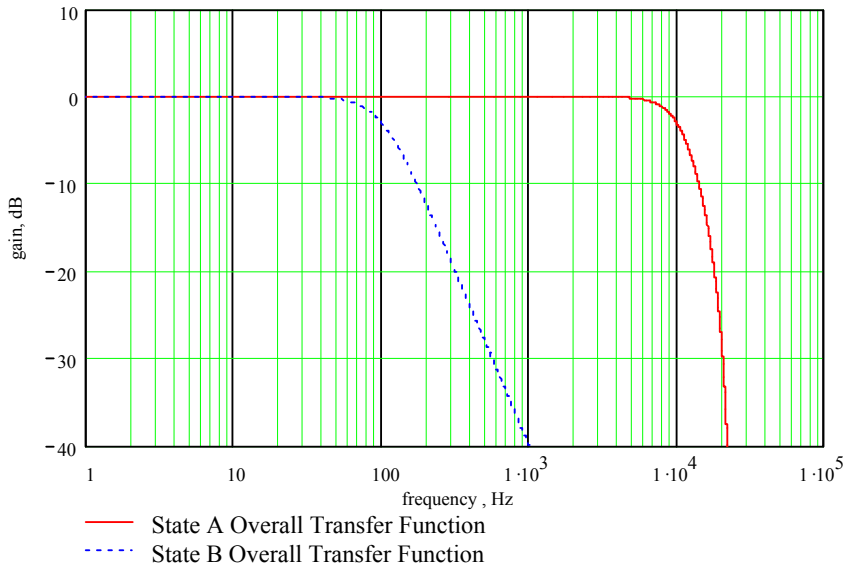

Figure 6-12 Overall magnitude response changes for filter response change Scenario 3, states A and B.

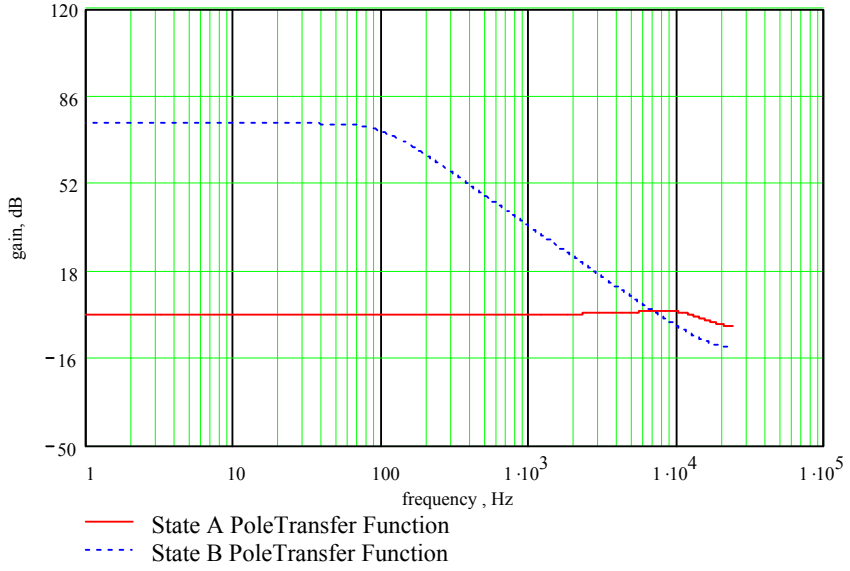

Figure 6-13 Pole transfer functions for filter response change Scenario 3, states A and B.

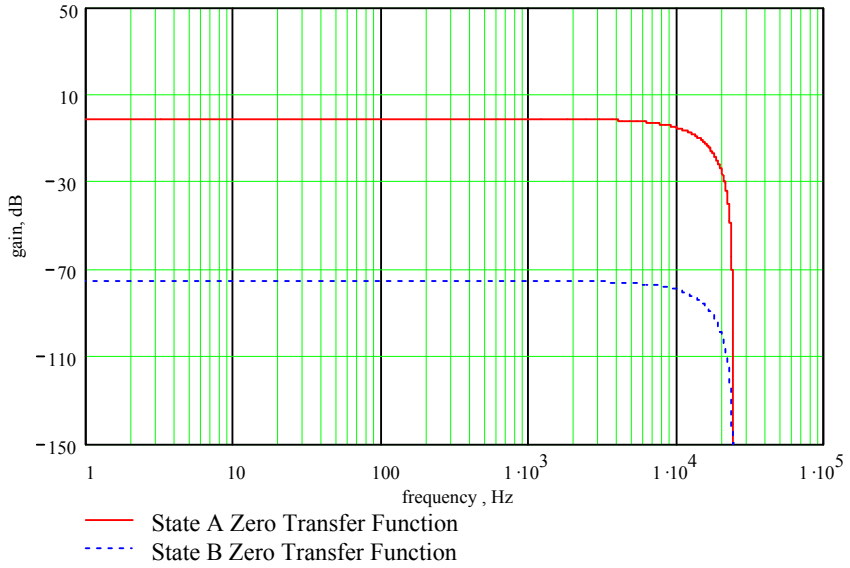

Figure 6-14 Zero transfer functions for filter response change Scenario 3, states A and B.

6.3.4 Filter state change 'Scenario 4', low to high pass filter type change

The filter state change introduced in 'Scenario 4' utilises two types of audio filter, the 2nd order Butterworth Low and High Pass filter. The tuned (–3dB cut-off) frequency is fixed to 100Hz for both types. The filter parameter settings for the two states A and B are,

State A : Low pass, Fc = 100 Hz, Q = 0.7071

State B : High pass, Fc = 100 Hz, Q = 0.7071.

This test scenario is particularly interesting filter type change, common in 'cross-over' filter configuration. It also does not result in any change of the pole transfer function between the two states. The overall magnitude responses, for the two states are shown in Figure 6-15. The pole transfer functions for the two states are shown in Figure 6-16. The zero transfer functions are shown in Figure 6-17.

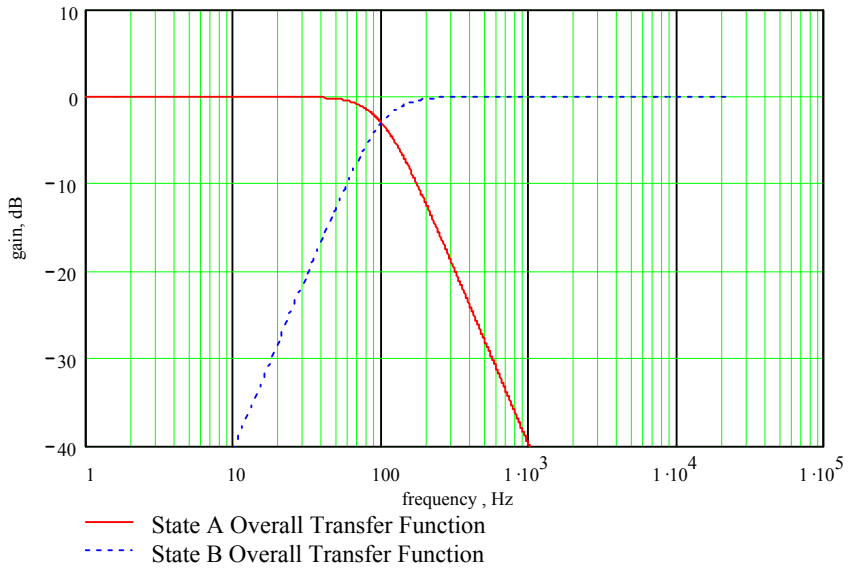

Figure 6-15 Overall magnitude response changes for filter response change Scenario 4, states A and B.

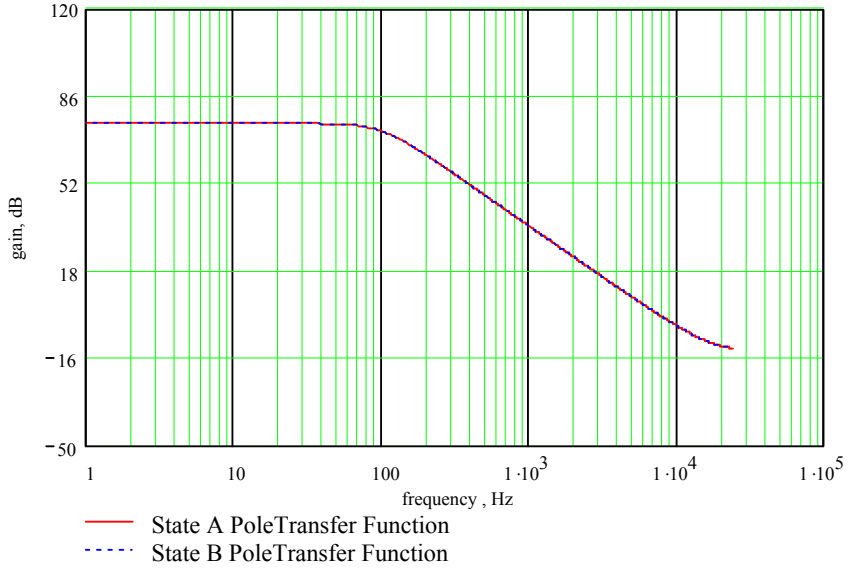

Figure 6-16 Pole transfer functions for filter response change Scenario 4, states A and B.

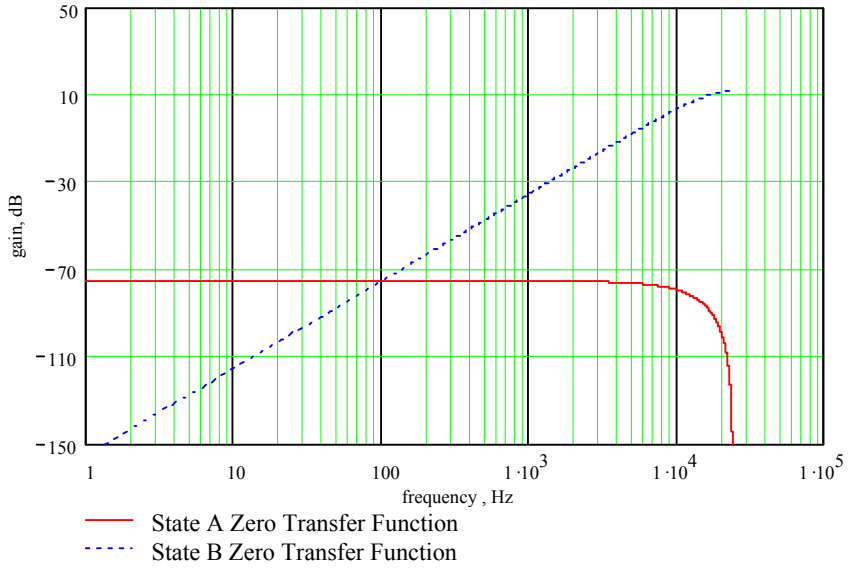

Figure 6-17 Zero transfer functions for filter response change Scenario 4, states A and B.

6.3.5 Filter state change 'Scenario 5', notch to low pass filter type change

The filter state change introduced in 'Scenario 5' utilises two types of audio filter, the 2nd order notch and low pass filter (Butterworth response). The tuned frequency is fixed to 40Hz for both types. The filter parameter settings for the two states A and B are,

State A : Notch , Fc = 40 Hz, Q = 0.7071

State B : High pass, Fc = 40 Hz, Q = 0.7071.

This test scenario is of interest since it may be performed whilst configuring of a speaker system, where notch filters are employed to reduce acoustic resonances from a low frequency driver (bass driver). The overall magnitude response, for the two states are shown in Figure 6-18. The pole transfer functions for the two states are shown in Figure 6-19. The zero transfer functions are shown in Figure 6-20.

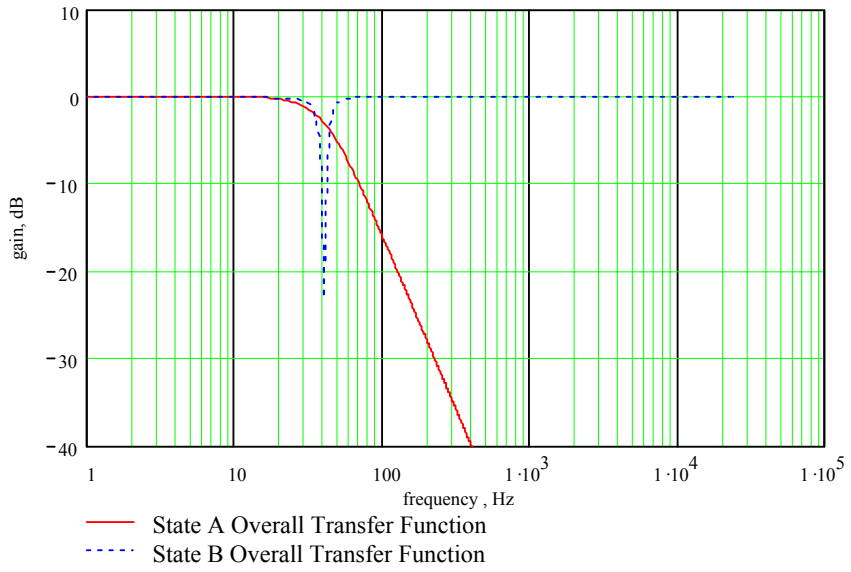

Figure 6-18 Overall magnitude response changes for filter response change Scenario 5, states A and B.

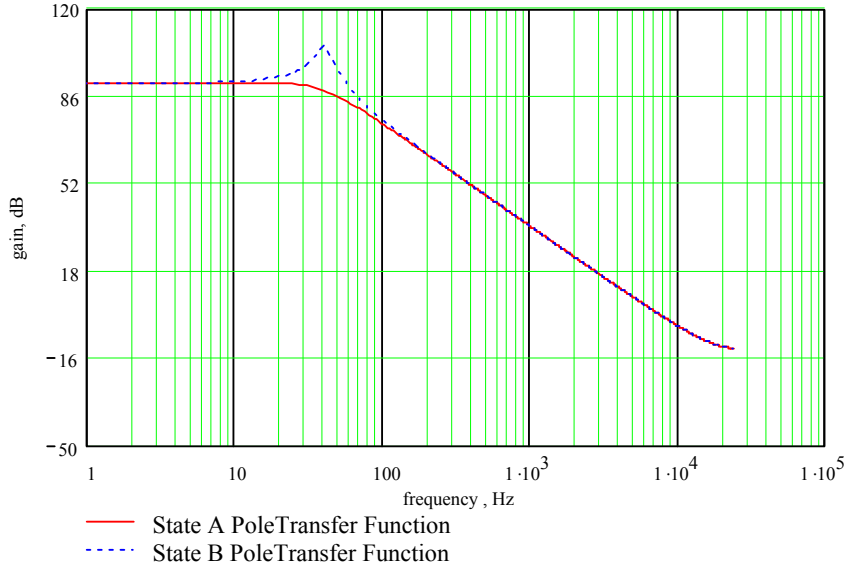

Figure 6-19 Pole transfer functions for filter response change Scenario 5, states A and B.

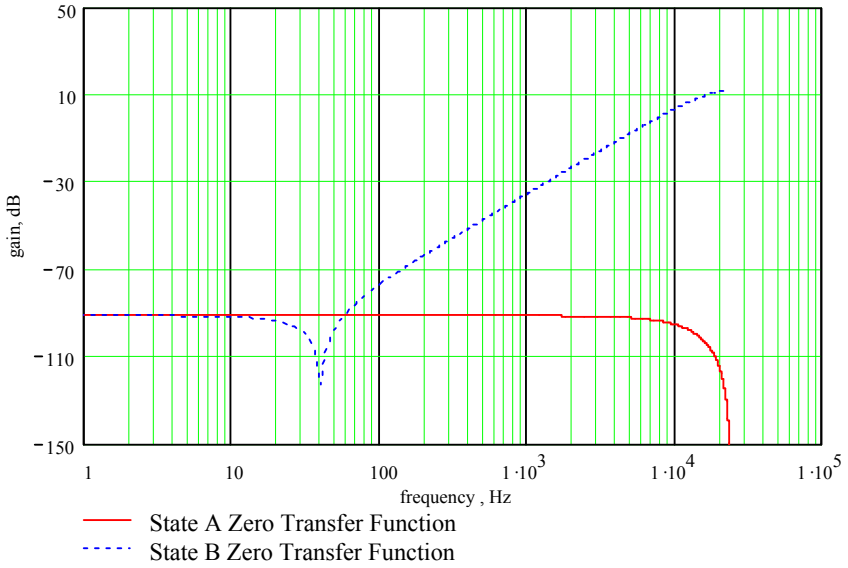

Figure 6-20 Zero transfer functions for filter response change Scenario 5, states A and B.

6.4 Theory and analysis of DF 1 under a step state change

The DF1 is a zero before pole topology, Figure 2-5. Therefore the zero transfer function does not provide any attenuation of the pole response. The pole transfer function implementation is fed by the filter output. Therefore any magnitude change at the output will result in magnitude changes in state variables in the pole paths. These magnitude changes can be considered as a step change to the pole transfer function and potentially generate signal disturbance. The size of the disturbance is dependent on the size of the step change in the pole paths and the potential gain in the pole transfer function at that instance in time. The following sub-sections of section 6.4, describe the disturbance behaviour of DF1 under the five test scenarios.

6.4.1 Analysis of DF1 - Scenario 1

No signal disturbance is evident at the output of DF1 for the Scenario 1 test. This is the case for dc, white noise and 10 kHz sine inputs. This is explained by unity gain response at all frequencies for both filter response states.

6.4.2 Analysis of DF1 - Scenario 2

No signal disturbance occurs using a dc input stimulus, for the Scenario 2 state change test. Both filter response states A and B exhibit unity gain at dc, Figure 6-9. Therefore no step change in magnitude occurs at the output of the filter and consequently no step change occurs in the pole paths.

Signal disturbance under filter state change disturbance is visible using a 10 kHz sinusoidal excitation, Figure 6-21. Inspection of Figure 6-9, shows an overall magnitude response change between state A and B of 10 dB, at 10 kHz. Therefore under 10 kHz sine excitation the pole paths are party to a step change of 10 dB between both state transitions (A to B and B to A). The large disturbance from state change A to B can be attributed to the high gain in the target filter's (state B) pole transfer function, Figure 6-10. State A pole transfer function has much less gain than state B (Figure 6-10). Therefore state change B to A does not produce significant disturbances, Figure 6-21.

Figure 6-22 shows the signal disturbance at the DF1 output, using a white noise input excitation. The disturbance is smaller than that caused by the 10 kHz tone. This is because the 10 kHz sine excitation generates a larger step change at the filter output and than that caused by the noise excitation.

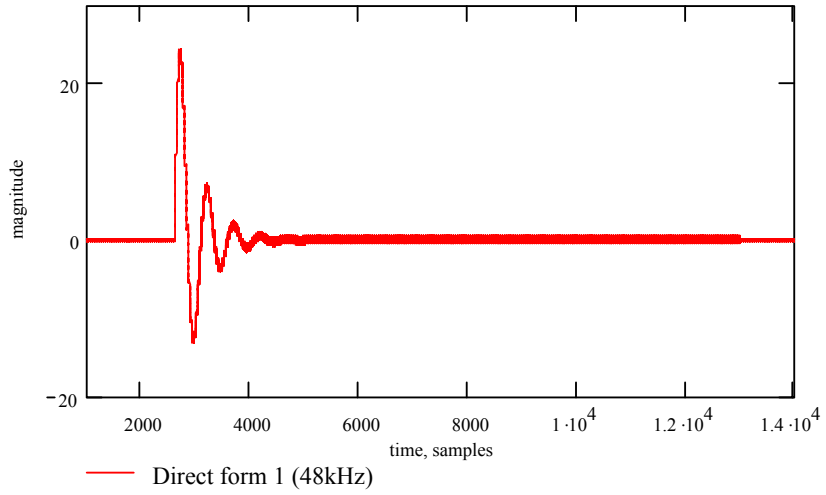

Figure 6-21 DF1 disturbance response to Scenario 2, using 10 kHz sine input excitation.

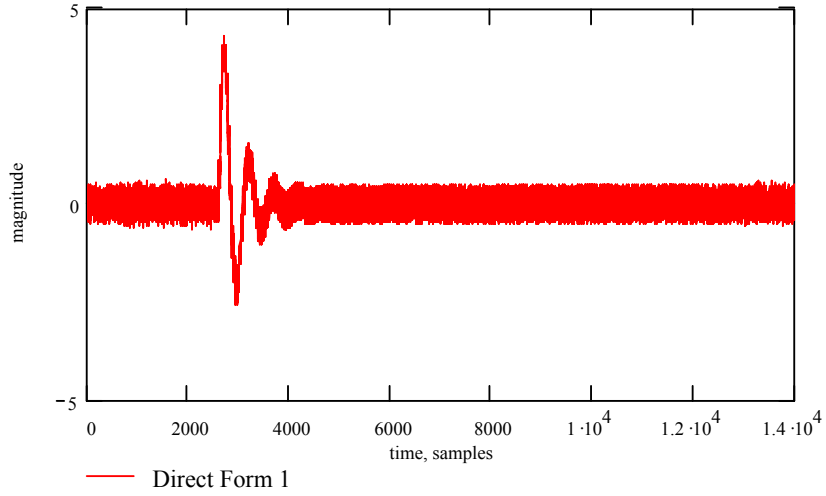

Figure 6-22 DF1 disturbance response to Scenario 2, using white noise input excitation.

6.4.3 Analysis of DF1 - Scenario 3

No disturbance occurs for the Scenario 3 filter response state change, using a dc input excitation. Both states A and B exhibit unity gain at dc, Figure 6-12. Using a 10 kHz sine input stimulus disturbances are produced on the transition from state A to B, Figure 6-23. This is due to the pole path disturbance being amplified by the state B pole transfer function. The state B to A change does not cause a disturbance since the state A pole transfer function produces negligible gain, Figure 6-13.

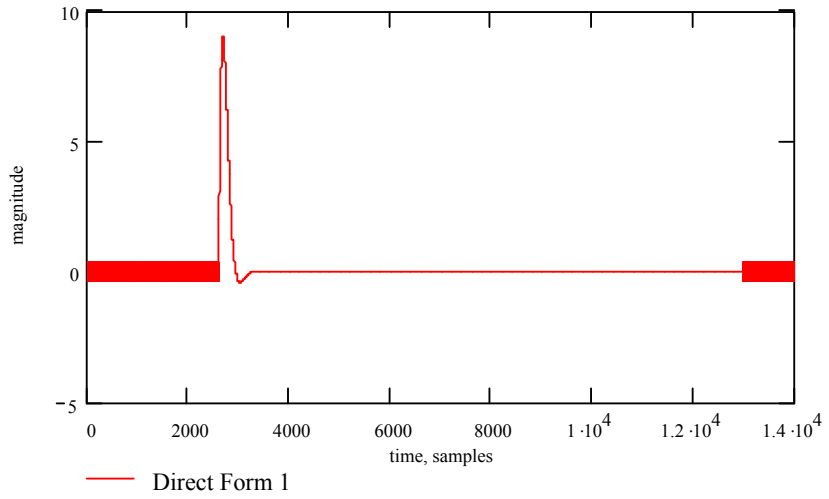

Figure 6-23 DF1 disturbance response to Scenario 3, using 10 kHz sine input excitation.

6.4.4 Analysis of DF1 - Scenario 4

DF1 produces a small disturbance using a dc input excitation, Figure 6-24. The state A magnitude response at dc is unity and for state B, theoretically zero. However this disturbance is smaller than could be expected and is possibly explained by the constant pole transfer function. Using a 10 kHz sine input stimulus the DF1 produces much larger amounts of disturbance on both state changes, Figure 6-25. The notable difference is that the zero transfer function magnitude response at 10 kHz switches between –75 dB to 0 dB. The magnitude response at dc switches between –75 dB to - ∞ dB.

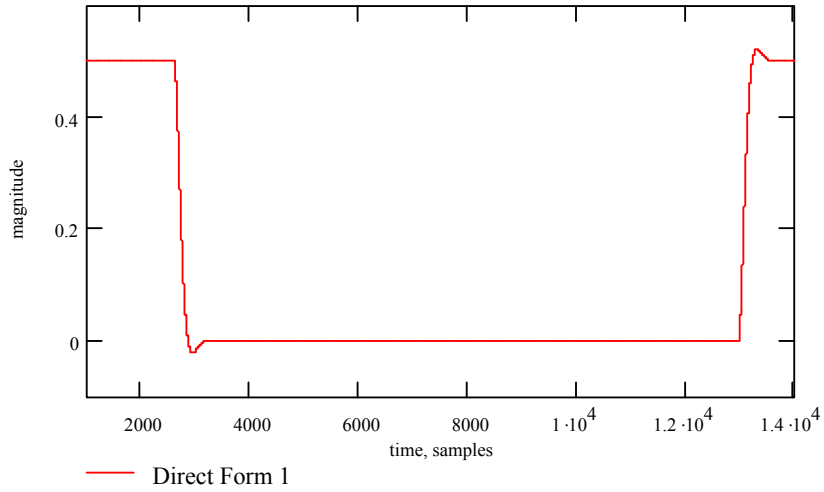

Figure 6-24 DF1 disturbance response to Scenario 4, using dc input excitation.

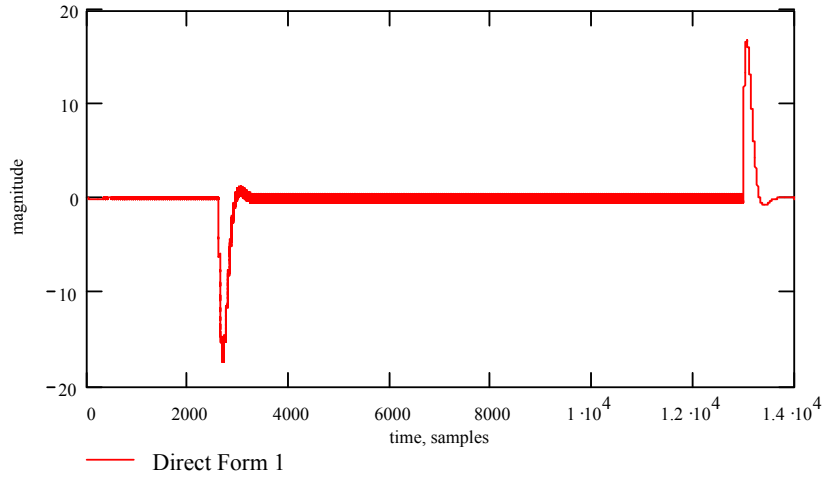

Figure 6-25 DF1 disturbance response to Scenario 4, using 10 kHz sine input excitation.

6.4.5 Analysis of DF1 - Scenario 5

Using a dc input, Scenario 5 produces no disturbance on either of the state transitions. Using a 10 kHz sine input stimulus, the DF1 produces large disturbances from state A to B, and from state B to A, Figure 6-26. The signal disturbance at the frequency response state change A to B can be attributed to the large step change in output level at 10 kHz (Figure 6-18). This step change in level excites the pole paths. The state B pole transfer function supplies 103 dB gain in the pole frequency region, Figure 6-19. The signal

disturbance at the frequency response state B to A can be attributed to the same large step change from unity output to a low level output at 10 kHz. This step change excites the state A pole transfer function. The state A pole transfer function supplies 80dB of gain in the pole frequency region.

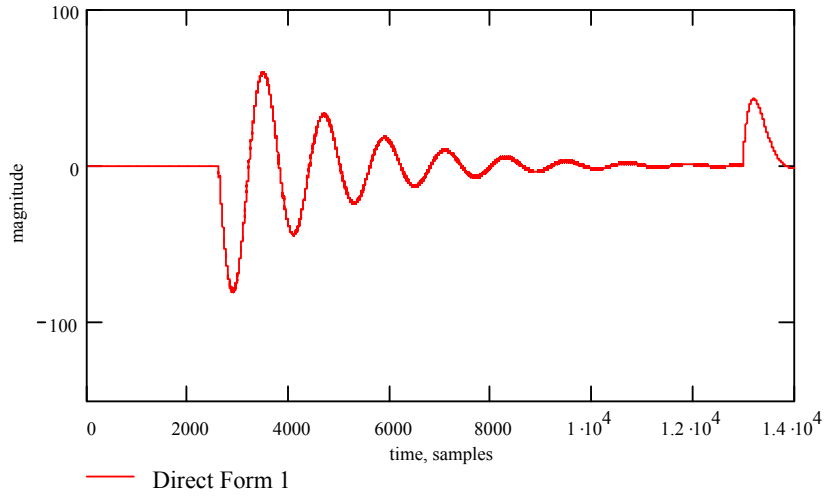

Figure 6-26 DF1 disturbance response to Scenario 5, using 10 kHz sine input excitation.

6.5 Theory and analysis of DF2 under a step state change

The DF2 topology, Figure 2-8, is a pole before zero topology, where the input excitation directly feeds the pole transfer function implementation. The zero transfer function then operates on the intermediate result (pole output), producing the filter output. Signal disturbance behaviour in DF2 therefore decomposes into the two mechanisms, the state change pole response under input excitation and the attenuation capabilities of the zero transfer function.

6.5.1 Analysis of DF2 - Scenario 1

No disturbance is generated under dc or 10 kHz input excitation.

6.5.2 Analysis of DF2 - Scenario 2

Figure 6-27 shows the pole output response under dc input excitation, for the Scenario 2 filter state change. The pole output (intermediate result) can be seen to contain large amounts of ringing. This can be attributed to the high gain in the state B pole transfer function, Figure 6-10. The response change from state B to A also causes ringing, but is attributed to the large step change in pole gain to the state A response. Under dc input there is a gain change of 75 dB in the pole transfer function between the two states. The effects of the state A and B pole disturbances on the actual filter output differ greatly due to the zero transfer functions. From state A to B the pole disturbance is greatly attenuated by the state B zero transfer function. However, from state B to A the pole disturbance is not greatly attenuated by the state A zero transfer function. Since the state A zero transfer function produces little rejection in the pole frequency region. Figure 6-28 shows the resulting disturbances.

Figure 6-29 shows the pole response using a 10 kHz sine input stimulus. A substantially smaller magnitude disturbance occurs between states A to B than that caused by the dc input. This can be explained by the much lower level of gain in the pole transfer function at 10 kHz, Figure 6-10. Thus when the state change from A to B occurs there is substantially less energy in the pole paths – causing less disturbance at the filter state change. Using a 10 kHz input stimulus, the B to A state change produces no disturbance, since the pole transfer function of A has no intrinsic gain to produce noticeable ringing. Figure 6-30 shows the resulting disturbances.

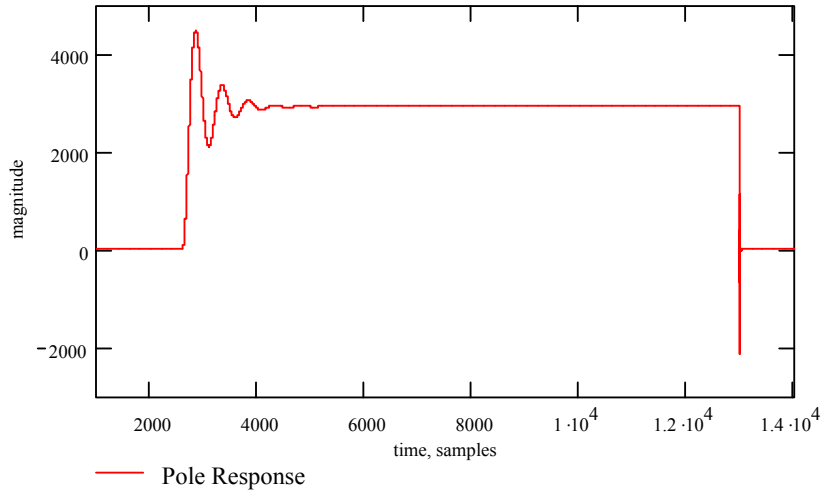

Figure 6-27 Direct Form pole response disturbance to Scenario 4, using dc input excitation.

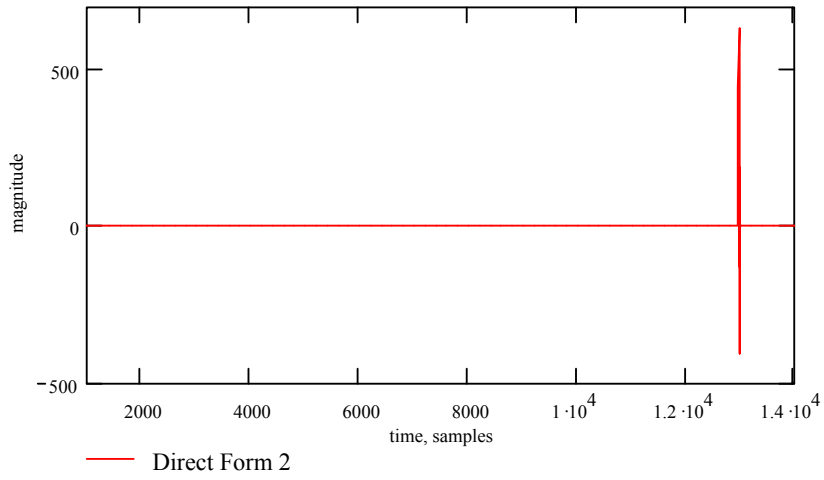

Figure 6-28 DF2 disturbance response to Scenario 2, using dc input excitation.

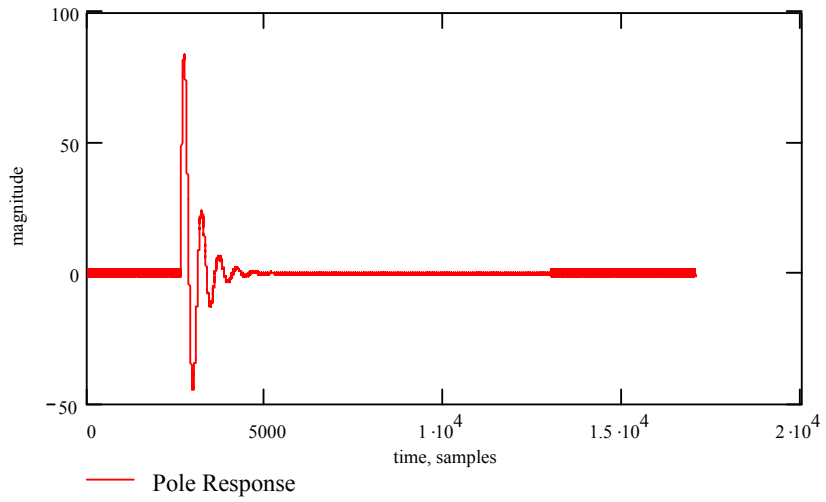

Figure 6-29 Direct Form pole response disturbance to Scenario 2, using 10 kHz sine input excitation.

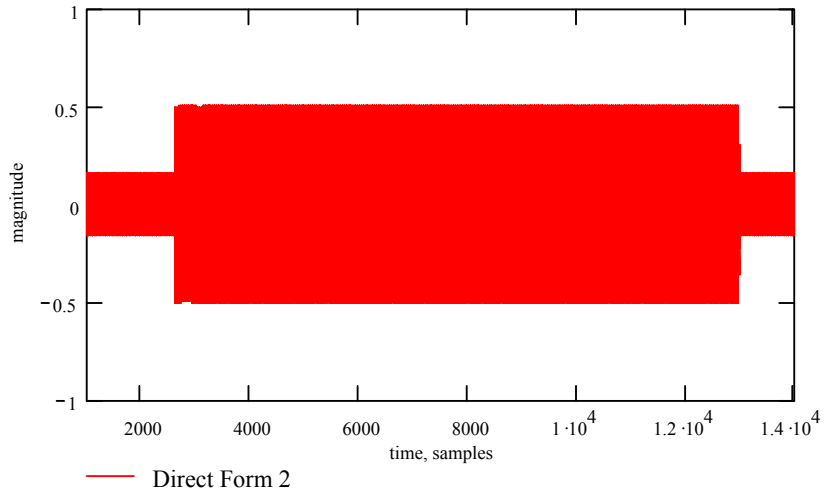

Figure 6-30 DF2 disturbance response to Scenario 2, using 10 kHz sine input excitation.

6.5.3 Analysis of DF2 - Scenario 3

Figure 6-31 shows the pole response to Scenario 3 test, under dc input stimulus. Magnitude changes at dc are large which produces large step changes in the pole response output. The zero transfer function of state B offers proportionally large amounts of broad band attenuation, which suppresses the state A to B pole disturbance. However the large pole response step change generated by state B to A is passed through zero transfer function state A, which is approximately flat at unity gain. Therefore the pole response

disturbance from state B to A is not attenuated by the state A zero transfer function. Therefore a noticeable output disturbance occurs. Using a 10 kHz sine input stimulus, the pole response between state A to B produces a relatively small disturbance, Figure 6-33. However this is also heavily attenuated by the state B zero transfer function, Figure 6-14. Using a 10 kHz input stimulus the state B to A pole response does not produce a disturbance, Figure 6-33. Subsequently no disturbances occur at the output, Figure 6-34.

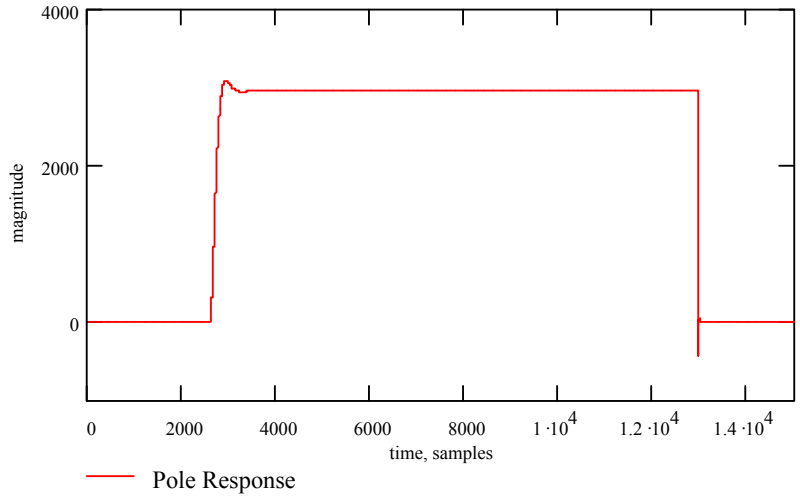

Figure 6-31 Direct Form pole response disturbance to Scenario 3, using dc input excitation.

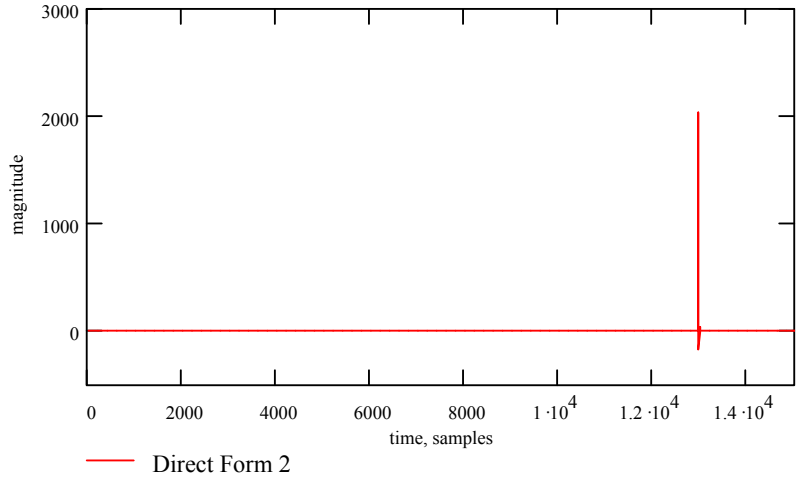

Figure 6-32 DF2 disturbance response to Scenario 3, using dc input excitation.

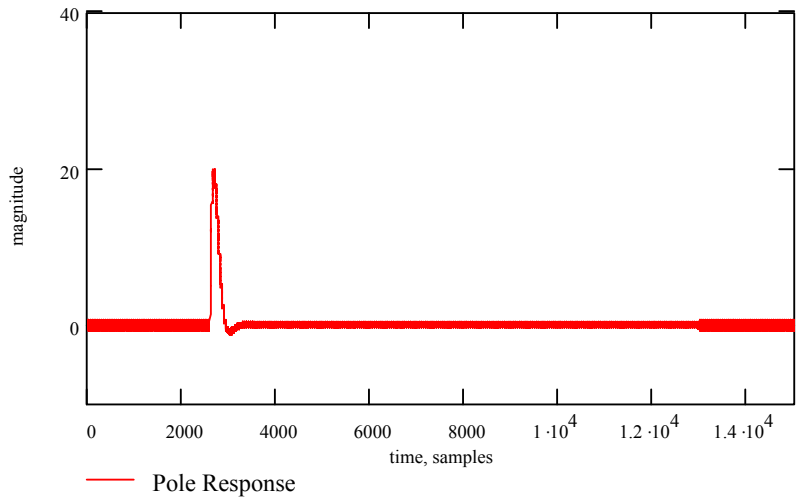

Figure 6-33 Direct Form pole response disturbance to Scenario 3, using 10 kHz sine input excitation.

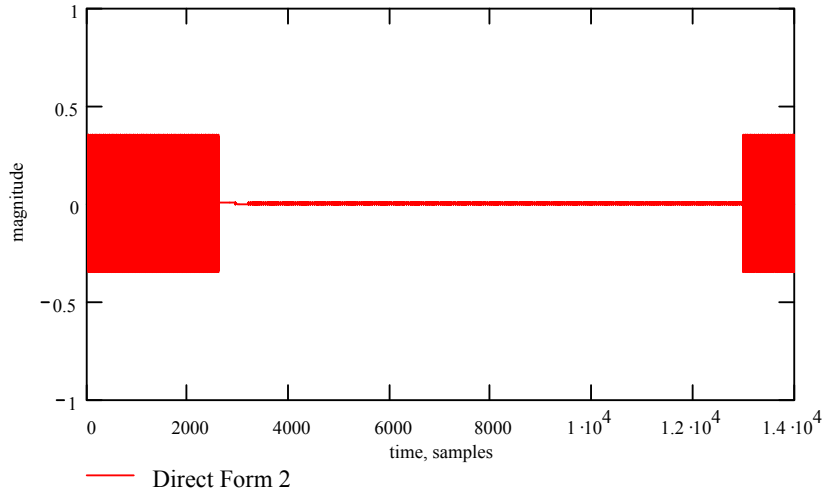

Figure 6-34 DF2 disturbance response to Scenario 3, using 10 kHz sine input excitation.

6.5.4 Analysis of DF2 - Scenario 4

DF2 does not produce any disturbance for dc or 10 kHz input stimulus. The pole transfer function does not alter through-out the state changes, Figure 6-16. Therefore no pole response disturbances are generated.

6.5.5 Analysis of DF2 – Scenario 5

There is no disturbance under dc or 10 kHz sine input stimulus. Despite the pole transfer functions for both states have high peak gains. The actual change in gain between the

state A and B pole transfer functions is relatively small. This results in a high magnitude signal output at the pole output, but under state change the ringing is relatively small (ringing magnitude of 400). The zero transfer functions for both states provide large attenuation in the pole frequency region, resulting in minimal disturbance.

6.6 Direct Form transposed Forms reaction to step-change

Topology diagrams for the transposed direct forms are shown in Figures 2-9 and 2-10. Coefficients are multiplied directly by the input or output and the unit delays operate on the products. If the coefficients are static and arithmetic finite wordlength effects are ignored, a transposed topology is numerically identical to the direct form. However the transposed topology applies a unit delay on the product, not on the input or output sample. This results in a 'time offset' if the coefficients vary with time. This can be explained by Equation (6-1). The equation is true, if $a1_i$ is constant for all instances of i. Once the coefficients become time varying ($a1_i$ is not constant), Equation (6-1) is no longer true.

$$a1_i \cdot x_{i-1} = \left(a1_i \cdot x_i\right)_{i-1}. \tag{6-1}$$

Further inspection of Figures 2-9 and 2-10, indicates that the actual transposed zero and pole transfer functions in the DF1T and DF2T are identical. The simple difference between DF1T and DF2T is the 'pole before zero' and 'zero before pole' ordering. Therefore a useful technique used in the following two sections is to examine the actual transposed zero and pole transfer functions reaction to coefficient state change separately. This information can then be re-applied to the DF1T and DF2T to establish the overall disturbance effects.

6.6.1 Direct Form 1 Transposed

The DF1T is a 'pole before zero' topology. Therefore its filter state change disturbance behaviour is more similar to the DF2 topology. For example, its disturbance response to Scenario 2, using a dc input excitation is similar to the DF2, Figure 6-28.

However there are major disturbance differences, caused by the transposed nature of the zero and pole transfer functions. Figure 6-35 shows the disturbance caused by DF1T Scenario 4, using a dc input excitation. DF2 does not produce any disturbances under these conditions. This is an interesting example, since the Scenario 4 test has a static pole transfer function. The source of the disturbance is a relatively small disturbance in the zero path response, Figure 6-36. The large energy produced by the static high gain pole transfer function is injected into the transposed zero transfer function implementation. This results in an overall disturbance much larger than the zero transfer function disturbance. The DF1T does not generate any disturbances for Scenario 4, using a 10 kHz input excitation. This is similar to DF2.

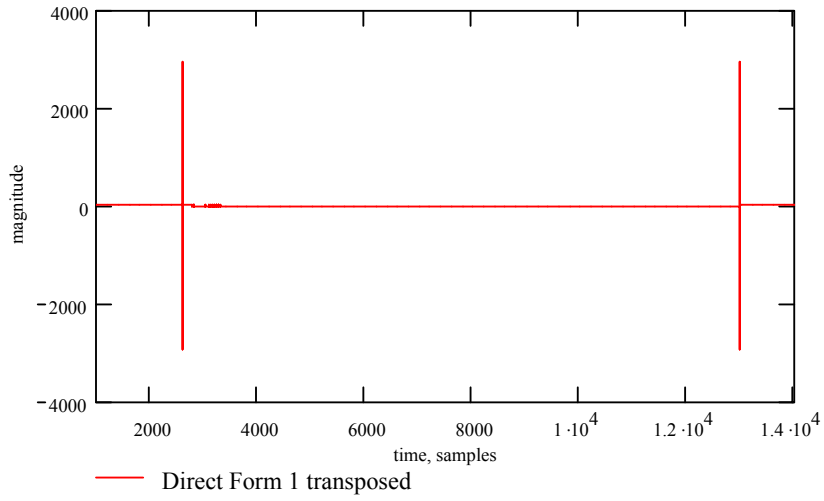

Figure 6-35 DF1T disturbance response to Scenario 4, using dc input excitation.

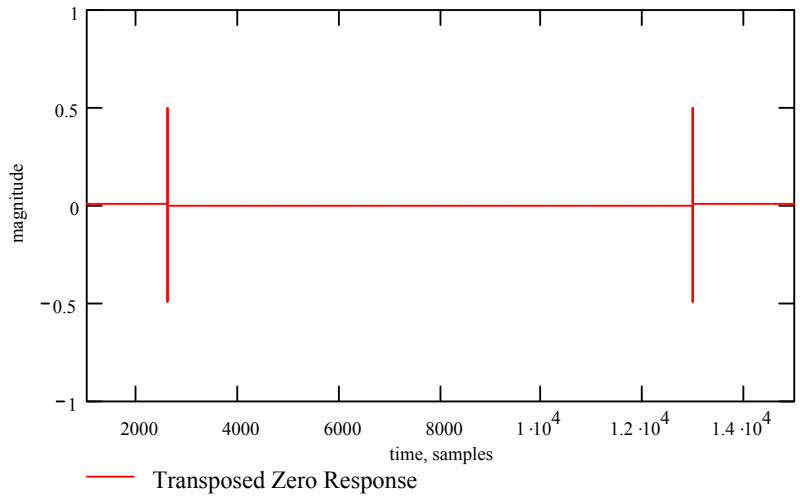

Figure 6-36 Transposed Direct Form zero response disturbance to Scenario 4, using dc input excitation.

The DF1T pole response disturbance to the Scenario 5 test is different to the DF2 pole response, Figure 6-37. Figure 6-38 shows the large resulting disturbance at the output of DF1T, for the Scenario 5 test, using a dc input. For this test example DF2 causes no disturbance.

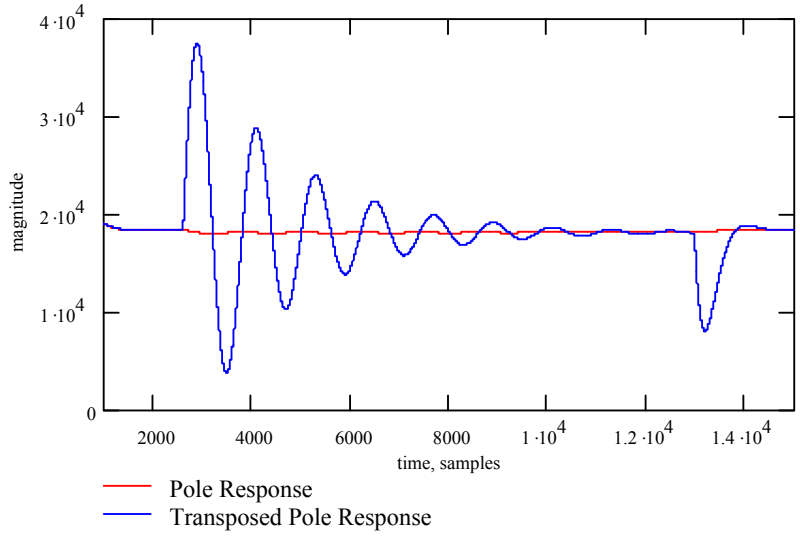

Figure 6-37 Direct Form and Transposed Direct Form pole response disturbances to Scenario 5 using dc input excitation.

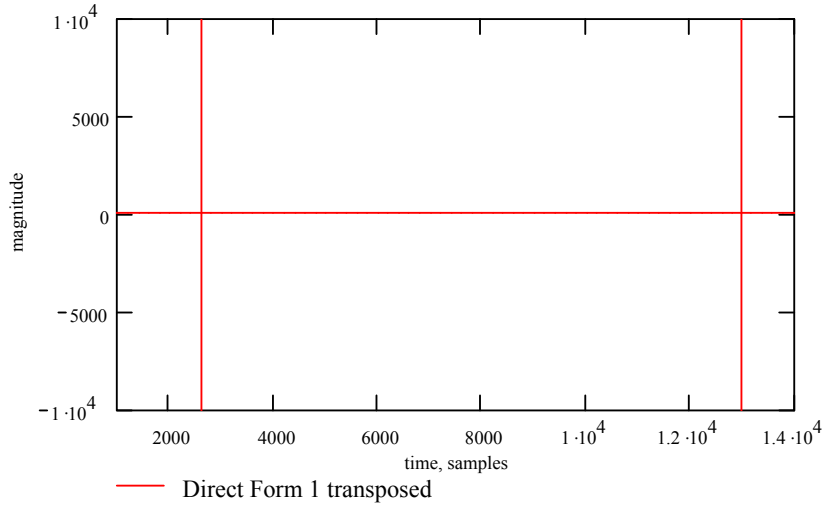

Figure 6-38 DF1T disturbance response to Scenario 5, using dc input excitation.

6.6.2 Direct Form 2 Transposed

DF2T produces a larger disturbance magnitude than DF1 for the Scenario 4 test, using a dc input excitation, Figure 6-39. The disturbance can be attributed to the zero transfer function disturbance, shown in Figure 6-36. The DF2T also produces a disturbance for the Scenario 5 state change test, using a dc input, Figure 6-40. DF1 produced no disturbance for this test. The DF2T disturbance for the Scenario 5 test is a product of the transposed pole transfer function disturbance, shown in Figure 6-37.

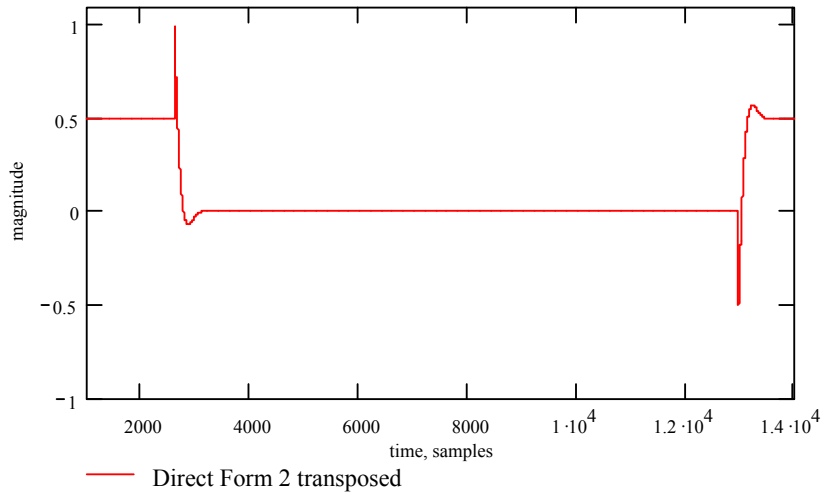

Figure 6-39 DF2T disturbance response to Scenario 4, using dc input excitation.

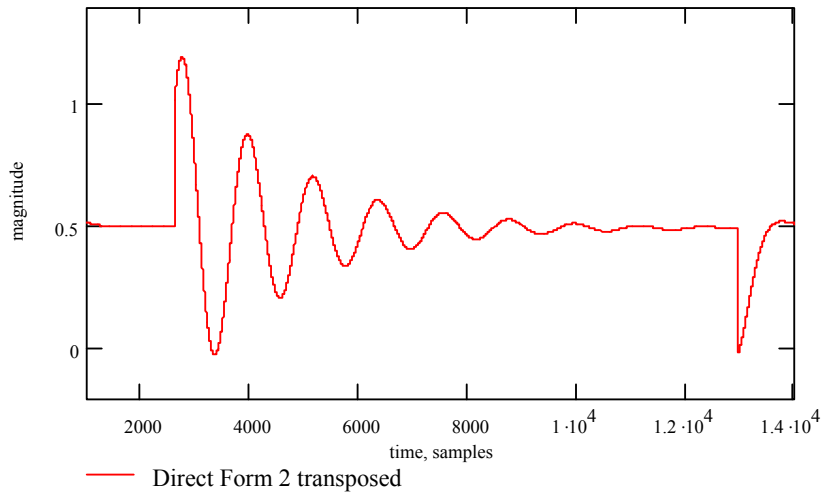

Figure 6-40 DF2T disturbance response to Scenario 5, using dc input excitation.

6.6.3 Transposed topologies with delay compensated coefficients

The previous two sections have shown some of the disturbances caused by the transposed zero and pole transfer function implementations. As described earlier in this section, transposed disturbances are caused by applying the unit delay operator on the product - not the input or output sample instance. The use of time varying coefficients results in a temporal misalignment between the coefficient variable and the input output data instance. This alters the effective transfer function for a few samples (depending on the filter order - number of delays in the transfer function). It is possible to correct the coefficient misalignment through a coefficient delay compensation scheme. By time shifting the coefficients by the inverse of the temporal mis-alignment, the transposed topologies disturbance behaviour is identical to the direct form. For example, if the coefficients in the DF1 are delay compensated, as shown in Equation (6-2), the disturbance behaviour resembles DF2T. If the DF2T uses a delay compensated coefficient set as shown in Expression (6-3) its disturbance behaviour matches that of DF1. If the DF1T uses a delay compensated coefficient set as shown in Expression (6-3) its disturbance behaviour matches that of DF2.

This coefficient delay compensation scheme was emulated, by applying the relevant delay compensation to the coefficient sets of the DF1T and DF2T. Figure 6-41 shows the differences in disturbance response for the DF2 (using no coefficient compensation) and DF1T (with coefficient delay compensation) for the Scenario 4 test, using dc input. Figure 6-42 shows the disturbance difference of DF1 (no coefficient compensation) and DF2T (with coefficient delay compensation) for the Scenario 5 test, using dc input. Both disturbance differences are negligible and within the noise boundaries of the simulation environment Mathcad (see Chapter 4).

$$y_i = a0_i \cdot x_i + a1_{i-1} \cdot x_{i-1} + a2_{i-2} \cdot x_{i-2} + b1_{i-1} \cdot y_{i-1} + b2_{i-2} \cdot y_{i-2} \tag{6-2}$$

$$a0_{i-2}\ ,\ a1_{i-1}\ ,\ a2_i\ ,\ b1_{i-1}\ ,\ b2_i \tag{6-3}$$

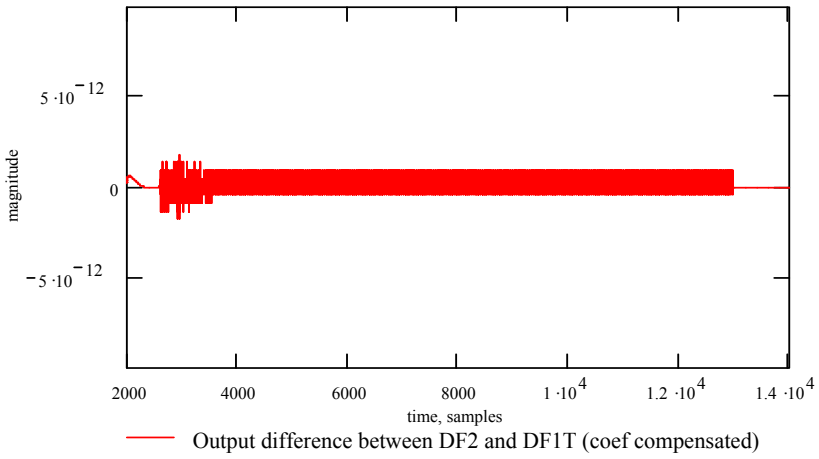

Figure 6-41 Difference in disturbance response for the DF2 and the DF1T (using coefficient delay compensation) for the Scenario 4 test, using a dc input excitation.

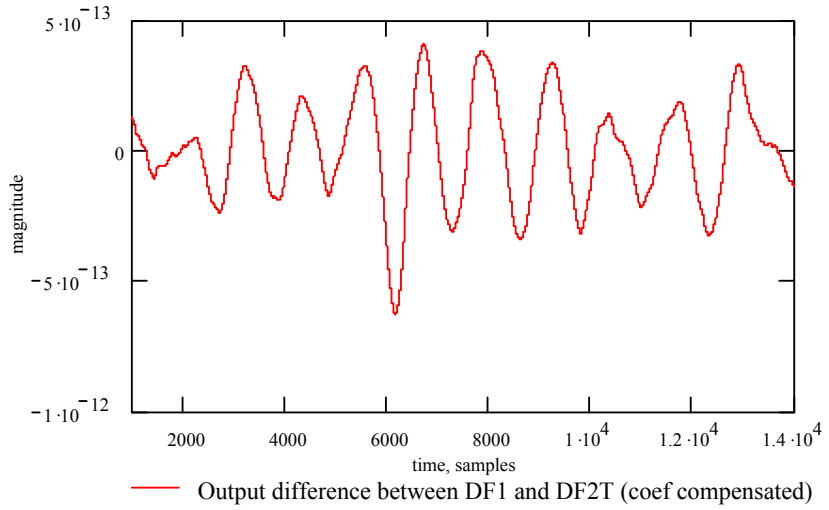

Figure 6-42 Difference in disturbance response for the DF1 and the DF2T (using coefficient delay compensation) for the Scenario 5 test, using a dc input excitation.

6.7 Ladder and lattice structures

The use of ladder and lattice allpass filters as sub-sections in audio equalisers is discussed in Section 2.5.7. The lattice and ladder structures are 'pole before zero' (Figure 2-16 and Figure 2-17). Two ladder implementations are considered in this work, Chapter 2, Section 2.5.7. The ladder with appended zeros, Figure 2-18, (Moorer structure) and the allpass ladder structure embedded in a gain network, Figure 2-17, (Massie structure). The lattice is only considered as an allpass structure embedded in a gain structure (Massie, 1993).

The first objective of this section is to show that both ladder implementations actually result in similar disturbance behaviour under coefficient state change. Figure 6-43 shows the difference between the disturbances of the Massie and Moorer ladder implementations for the Scenario 2 test, using a dc input excitation. The difference is essentially a small dc level shift - which causes some ringing. It is clear that the resultant difference is negligible and within the internal noise boundaries of the Mathcad environment.

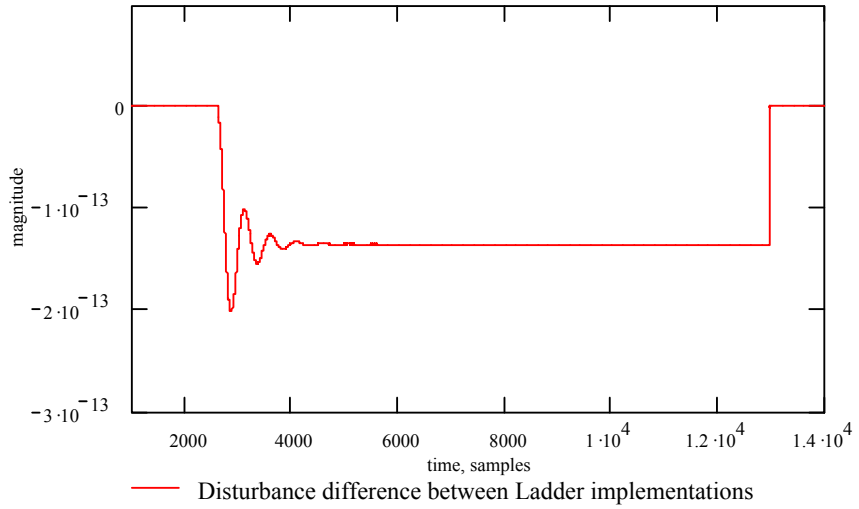

Figure 6-43 Difference in disturbance response for the two Massie and Moorer ladder implementations for the Scenario 2 test using a dc excitation.

Using the Scenario 2 test with a 10 kHz sine input stimulus the ladder and lattice structures produce minimal disturbance, similar to the DF2 response (previously shown in Figure 6-30). Figure 6-44 shows the disturbance produced by the lattice (Massie) for the Scenario 2 test, using a dc input excitation. The lattice disturbance behaviour closely matches that of the DF2 topology, Figure 6-28. Figure 6-45 shows the disturbance produced by the ladder (Massie) for the Scenario 2, using a dc input excitation. Due to the L_2 scaling at each accumulator node, the ladder topology state variables are not large in magnitude. Subsequently the ladder produces a smaller disturbance than the unscaled DF2 and lattice topologies. Note, the state A to B disturbance shown in the ladder response, Figure 6-45, (in the sample region of 2500, on the time axis) is not visible, due to graph scale, on the DF2 and lattice disturbance plots.

The ladder and lattice allpass output disturbances are actually larger in magnitude than the final structures output disturbance. This is because the surrounding gain elements attenuate the allpass output disturbance, for the example filter response shown.

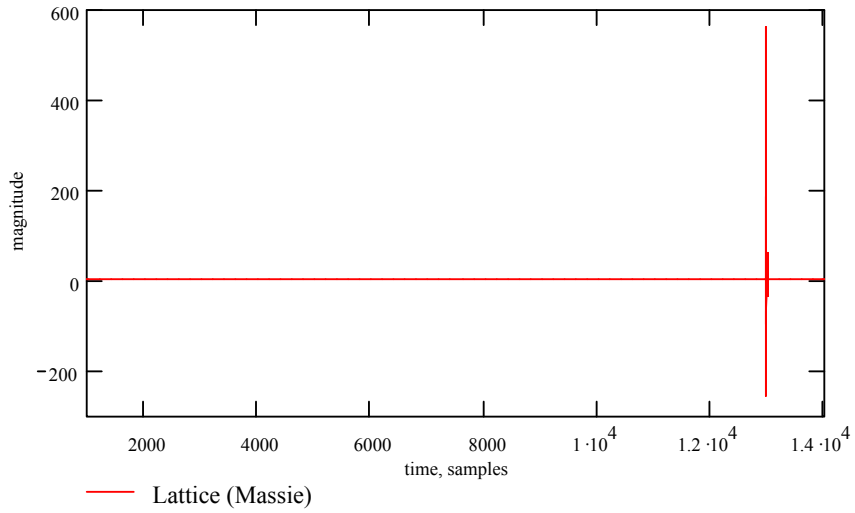

Figure 6-44 Lattice (Massie) disturbance response to Scenario 2, using dc input excitation.

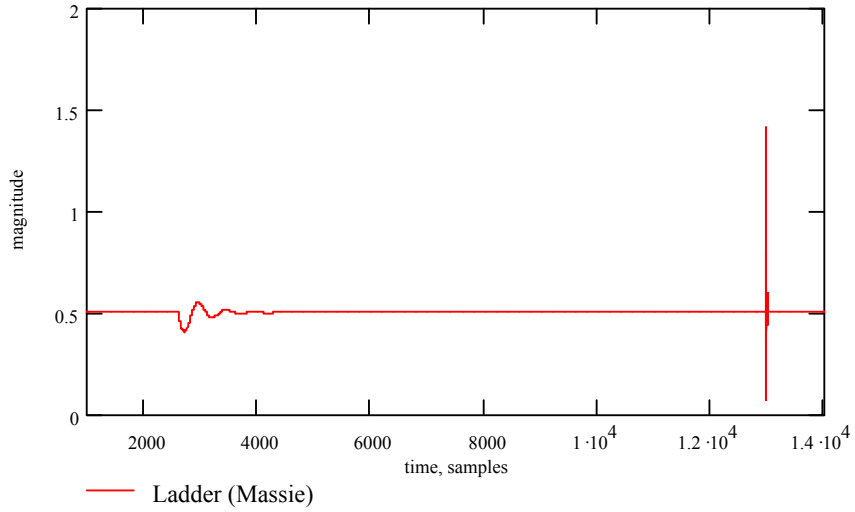

Figure 6-45 Ladder (Massie) disturbance response to Scenario 2, using dc input excitation.

6.8 Coupled forms Gold-Rader, Kingsbury, Zölzer

The Gold-Rader, Zölzer and Kingsbury topologies are introduced in Chapter 2, Section 2.5.4. These structures are 'zero before pole' topologies and will be closely compared to the DF1 topology. Large disturbances were found to be produced in DF1 using the Scenario 5 test and is an interesting test scenario for the coupled forms. Using a 10 kHz sine input stimulus, the three coupled forms all produce extremely similar disturbance characteristics to that of the DF1. Figure 6-46 shows the disturbance similarities of the

DF1 and the coupled forms, for the Scenario 5 test, using a dc input excitation. However, large disturbance differences are visible for dc input excitation, Figure 6-47. The Zölzer and Kingsbury disturbances are similar and larger than the Gold-Rader disturbance. The DF1 disturbance is minimal. Figure 6-48 shows an example of coupled form disturbance differences, where the Zölzer disturbance is much larger than the other topologies. Coefficient correction schemes were explored with the objective of minimising the disturbance differences between the coupled forms and the DF1, under dc excitation. No schemes were found to meet this objective. It is suggested that the integrators in the Kingsbury and Zölzer structures are influential in the disturbance differences for dc excitation.

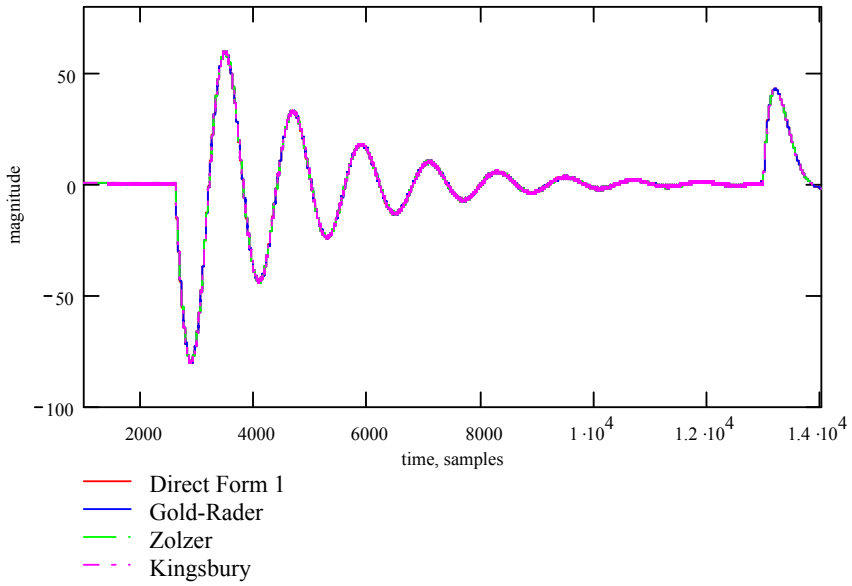

Figure 6-46 Coupled forms and DF1 disturbance responses for the Scenario 5 test, using 10 kHz input excitation.

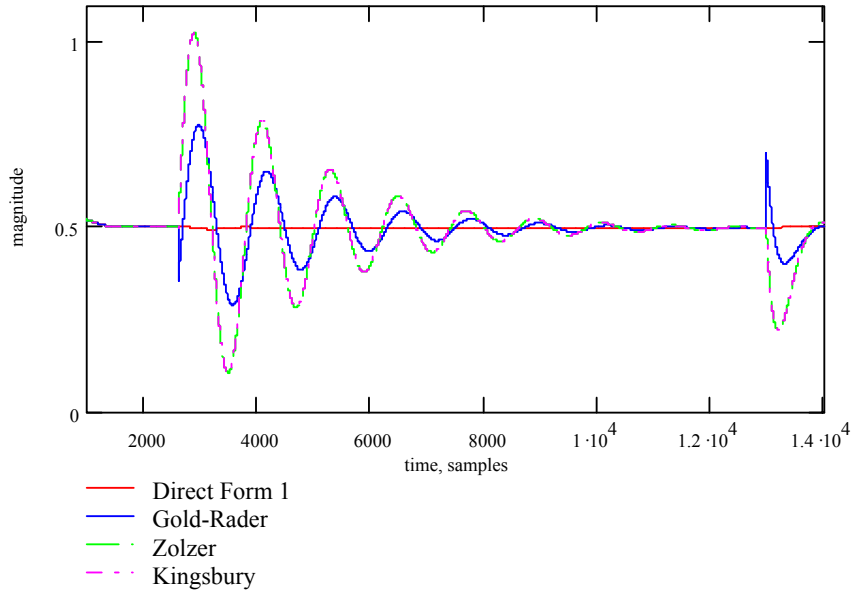

Figure 6-47 Coupled forms and DF1 disturbance responses for the Scenario 5 test, using dc input excitation.

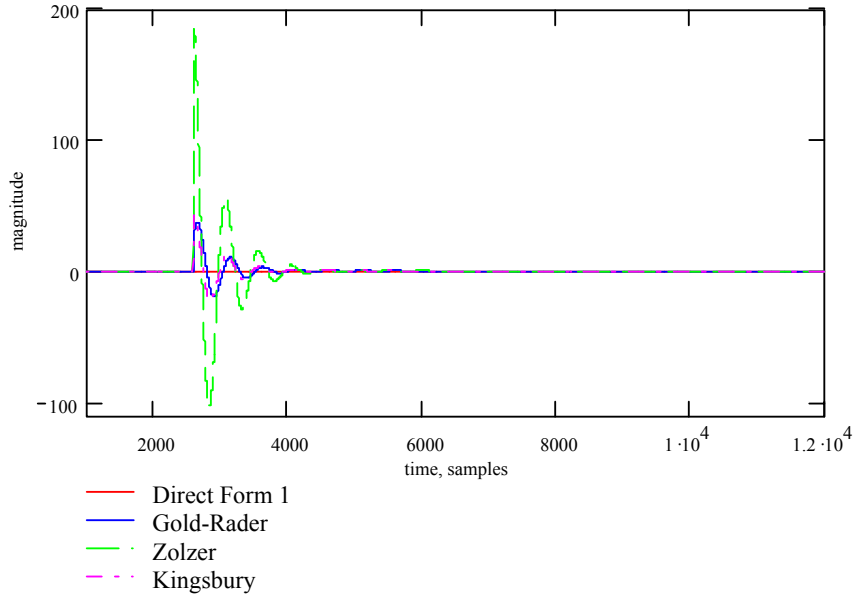

Figure 6-48 Coupled forms and DF1 disturbance responses for the Scenario 2 test, using dc input excitation.

6.9 State-space hybrid (Cabot) structure

The Cabot 'state-space' hybrid topology is introduced in Chapter 2 Section 2.5.6. The structure is a 'zero before pole' topology and uses the direct form zero transfer function

implementation coupled with the state-space pole transfer function. Using a 10 kHz sine input stimulus, the Cabot topology produces extremely similar disturbance characteristics to that of the DF1 and coupled forms, for the scenario 5 test. The disturbance is very similar to the disturbance shown in Figure 6-46. However like the coupled forms, larger disturbance differences become visible under dc input excitation, Figure 6-49. Figure 6-50 shows the disturbance for Scenario 2, dc input. The disturbance is larger than that of DF1, but not as large as the coupled form disturbances, Figure 6-48. It is suggested that the complex feedback paths in the pole transfer function implementation causes these disturbance differences for dc input excitation.

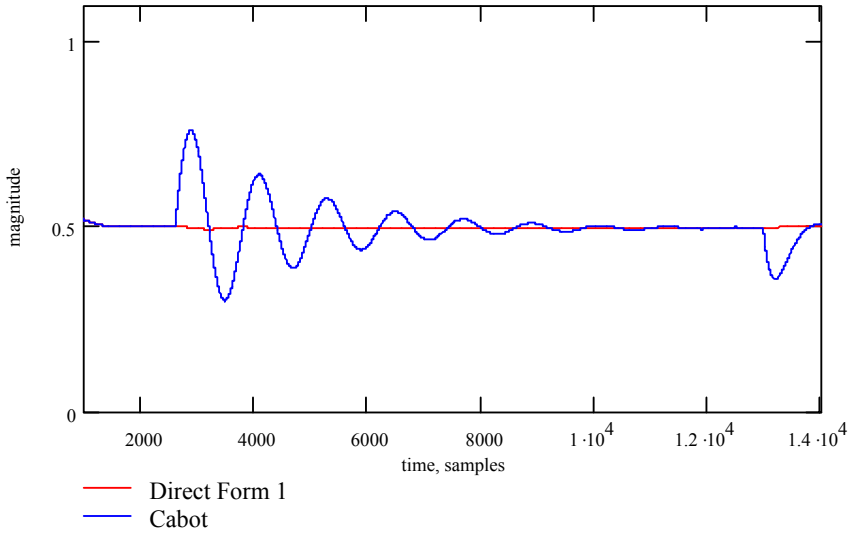

Figure 6-49 DF1 and Cabot structure disturbance responses for the Scenario 5 test, using dc input excitation.

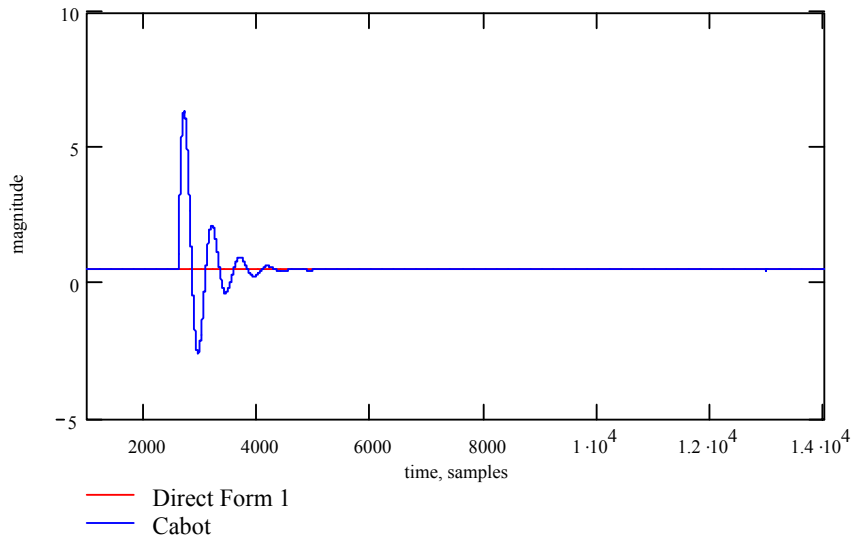

Figure 6-50 DF1 and Cabot structure disturbance responses for the Scenario 2 test, using dc input excitation.

6.10 State-space topology

The state-space topology is introduced in Chapter 2 Section 2.5.5. The pole transfer function is implemented in the middle of the topology. The zero transfer function implementation is partly in parallel with the pole transfer function implementation. The internal accumulation modes are scaled using the L_2 norm. Using a 10 kHz sine input stimulus, the topology produces negligible disturbance at the output. The state-space topology is most susceptible to disturbance using low frequency input excitations with a similar disturbance behaviour to 'pole before zero' topologies. The state-space disturbance behaviour for the Scenario 2 test using a dc input excitation is shown in Figure 6-51. This disturbance behaviour is similar to the ladder topology disturbance shown in Figure 6-45. The ladder and state-space topologies both use internal L_2 scaling schemes, both producing a much smaller but similar disturbance characteristic to the unscaled DF2 and lattice topologies.

The state-space topology also produces additional disturbance effects that do not occur in the DF2, ladder and lattice topologies. Figure 6-52 shows the disturbance resulting from

the Scenario 5 test, using a dc input excitation. It is clear that no noticeable disturbance occurs in the DF2. This state-space disturbance is similar to the transposed pole path disturbance, and that of the overall disturbance produced by the DF2T, Figure 6-40. The state-space topology disturbance (Figure 6-52) using dc excitation, is also similar to the disturbance produced by the Zölzer and Kingsbury coupled form topologies under the same test parameters, Figure 6-47. Coefficient alignment schemes were investigated but were found not to reduce the disturbance shown.

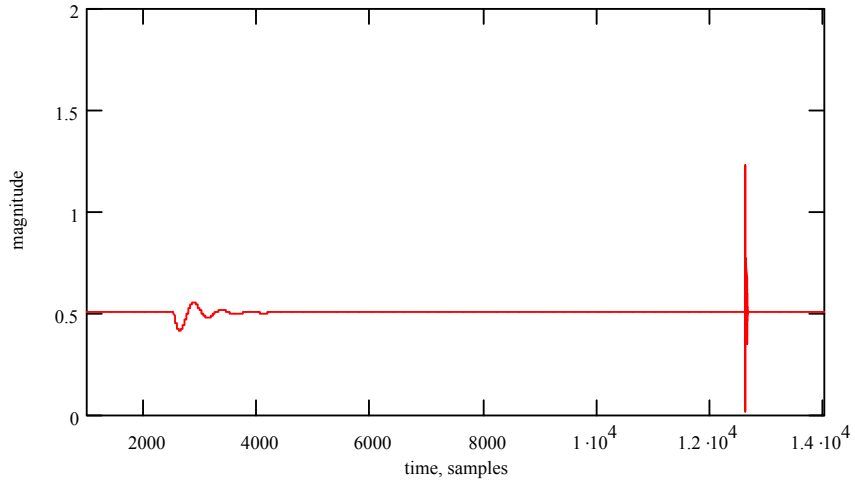

Figure 6-51 State-space structure disturbance response for the Scenario 2 test, using dc input excitation.

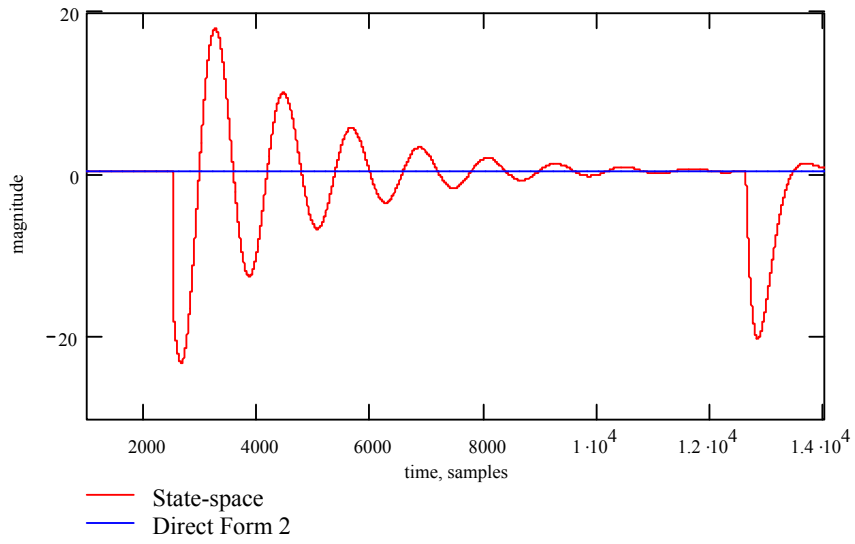

Figure 6-52 State-space structure disturbance response, for the Scenario 5 test, using dc input excitation.

6.11 Common interpolation techniques to reduce step change disturbance

This section aims to assess coefficient and parameter interpolators operating at the same sampling rate as the signal sampling frequency, 48 kHz. A background into the fundamentals of interpolation schemes to reduce filter state change disturbances is given in Chapter 2 Section 2.6. Coefficient and parameter interpolation techniques reduce filter state step change disturbances by decomposing the large step change into many smaller step changes. Despite this large disturbance reduction, smaller disturbance artefacts are introduced into the filter system.

Parameter interpolation cannot be used directly for scenarios that introduce non-continuous parameter types between filter state changes. Techniques exist that employ parameter interpolation for non-continuous parameter types (Zölzer, 1993) and are discussed in Chapter 2 Section 2.6.5. The Scenario 2 filter state change uses continuous parameter types and can exploit standard parameter interpolation techniques. Scenario 2 was also found to produce large step change disturbances as described in the preceding sections of this chapter.

Figure 2-21 shows the linear, exponential, sinusoidal and parameter interpolation schemes operating on a filter coefficient. Linear interpolation has a fixed rate of change and step-size. Exponential interpolation has its largest rate of change and step-size at the start of the interpolation period. Sinusoidal interpolation has its largest rate of change and step-size in the middle of the interpolation period. Parameter interpolation has a rate of change that is dependent on the parameter control law (for example frequency controls are typically logarithmic) and the parameter to coefficient mapping.

From the work described earlier in this chapter it is evident that any step change in a pole transfer function with high gain is potentially capable of generating a disturbance. The various interpolators produce their largest step-size at different instances of time in the interpolator period and therefore on different intermediate transfer functions. For this reason particular frequencies as sine input excitations may not highlight potential disturbance problems for a certain interpolator type. For example, the use of a 10 kHz sinusoidal stimulus was shown to produce large disturbances for the Scenario 2, step state change. Once interpolation is introduced, the Scenario 2 test does not produce its worst case disturbances using a 10 kHz sinusoidal input. Furthermore a 2 kHz sinusoidal input stimulus produces a worst-case disturbance using interpolation operating at 48 kHz, for the Scenario 2 test. This topic of input frequency, interpolator and resulting disturbance is discussed in the following sub sections.

6.11.1 State change interpolation for DF1

Figure 6-54 to Figure 6-56 show the DF1 disturbances generated by the four interpolation techniques for the Scenario 2 test, using a 2 kHz sine input stimulus. The linear interpolation produces a noticeably larger disturbance than the other interpolation schemes.

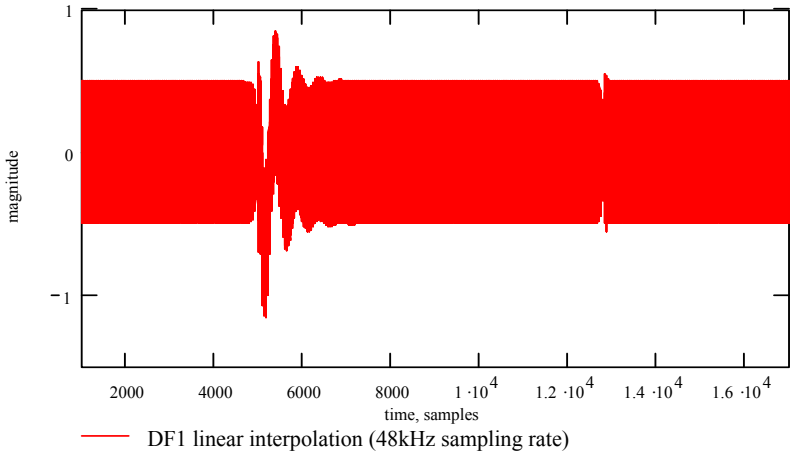

Figure 6-53 DF1 disturbance response to Scenario 2, using linear interpolation, operating at 48 kHz, using a 2 kHz sine input excitation.

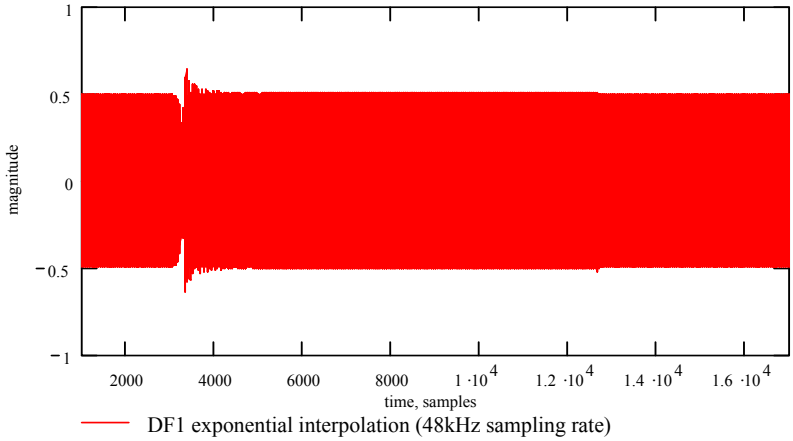

Figure 6-54 DF1 disturbance response to Scenario 2, using exponential interpolation, operating at 48 kHz, using a 2 kHz sine input excitation.

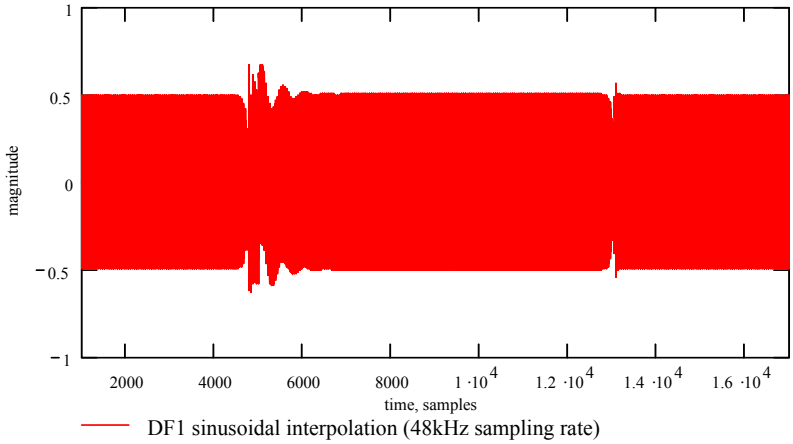

Figure 6-55 DF1 disturbance response to Scenario 2, using sinusoidal interpolation, operating at 48 kHz, using a 2 kHz sine input excitation.

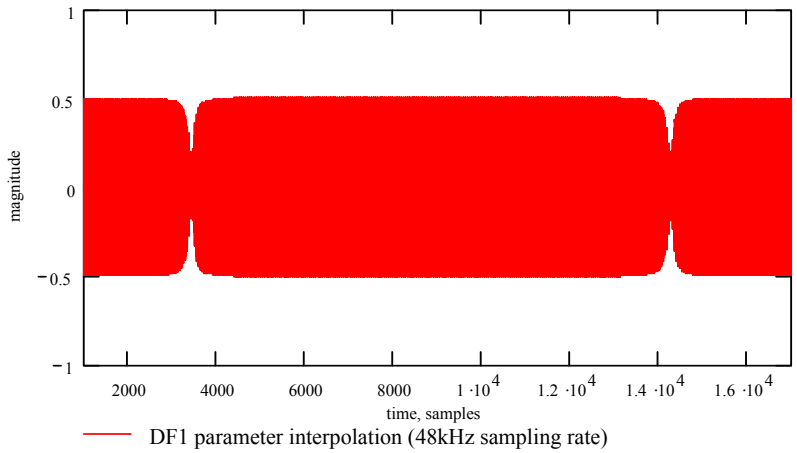

Figure 6-56 DF1 disturbance response to Scenario 2, using parameter interpolation, operating at 48 kHz, using a 2 kHz sine input excitation.

Figure 6-57 shows an expanded view of the linear interpolation disturbance. Marker X shows where the disturbance starts, sample 4975. Marker Y is where the interpolation period ends and the coefficient becomes static at its final value, sample 5041. It is clear that the disturbances start in the latter stages of the interpolation period. The visible ringing in the linear interpolation disturbance is caused by the large step sizes at the end of the interpolation, Figure 6-58. This acts as a disturbance excitation source for the high gain, final state B pole transfer function, see Figure 6-10. Figure 6-59 shows the frozen-time pole transfer functions using the intermediate linear interpolated coefficient sets at the sample instances X and Y. It is clear that the changes in the pole transfer function are considerable at this point of the interpolation.

Figure 6-54 shows the resulting disturbance for the exponential interpolator. The disturbance is noticeably smaller than that produced by the sinusoidal and linear interpolators. Disturbances produced under exponential interpolation occur at the start of the interpolation period, Figure 6-54. For the example shown the pole paths do not possess large amounts of gain at the start of the interpolation period, since the state A pole transfer function is much lower in gain than in state B, Figure 6-10.

The exponential interpolator produces the largest disturbance for the Scenario 5 test, using a 10 kHz input stimulus Figure 6-60. The disturbance occurs at the start of the interpolation period where the step change is largest (sample region 2500), Figure 6-60. The input frequency of 10 kHz was empirically found to be the worst case frequency for this particular test. This is attributable to the transfer function energy changes in the 10 kHz region at the start of the interpolation period. The Scenario 5, state A and B pole transfer functions are both similar high gain responses Figure 6-19. The large steps, produced by the exponential interpolator at the start of the interpolation period, are subjected to high gain in the pole transfer function. The Scenario 5 linear interpolation disturbance, Figure 6-53, is smaller than the exponential interpolator disturbance, Figure 6-60, since the initial linear steps are smaller than the exponential steps.

Using Scenario 2, sinusoidal interpolation generates disturbances in the latter half of the interpolation period, Figure 6-55. Despite the largest steps being in the middle of the interpolation period, the disturbance occurs at the end of the interpolation period. This is due to the increasing gain in the pole transfer function towards the end of the interpolation period. Parameter interpolation produces no noticeable disturbance effects, Figure 6-56.

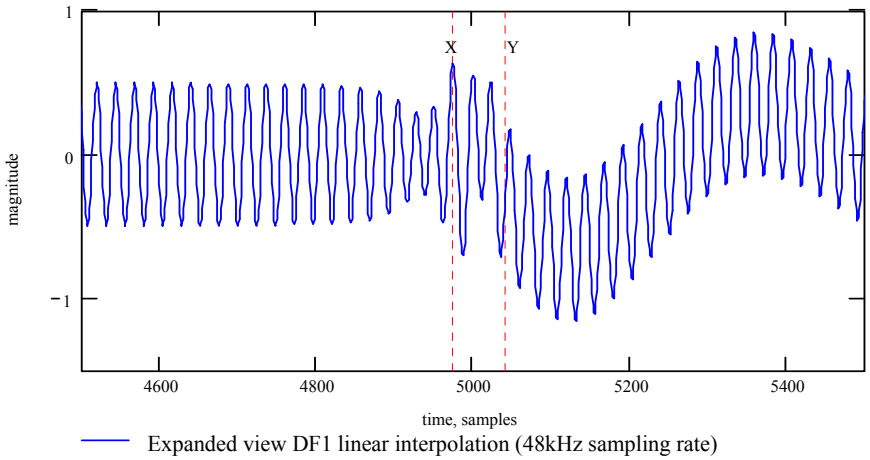

Figure 6-57 Expanded time view of Figure 6-53.

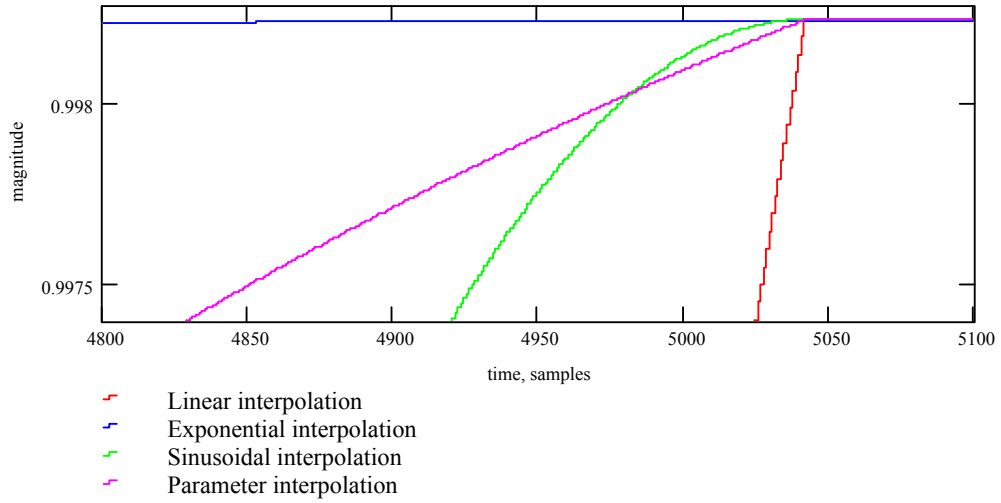

Figure 6-58 Final stages of an interpolation period for the various interpolation schemes.

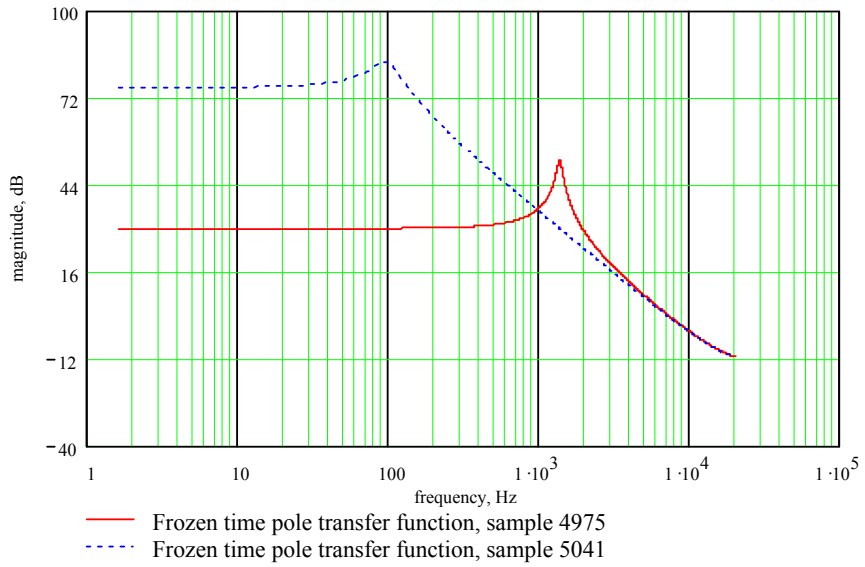

Figure 6-59 Two frozen-time pole transfer functions of intermediate linear interpolated coefficient towards the end of the interpolation period (using the Scenario 2 test).

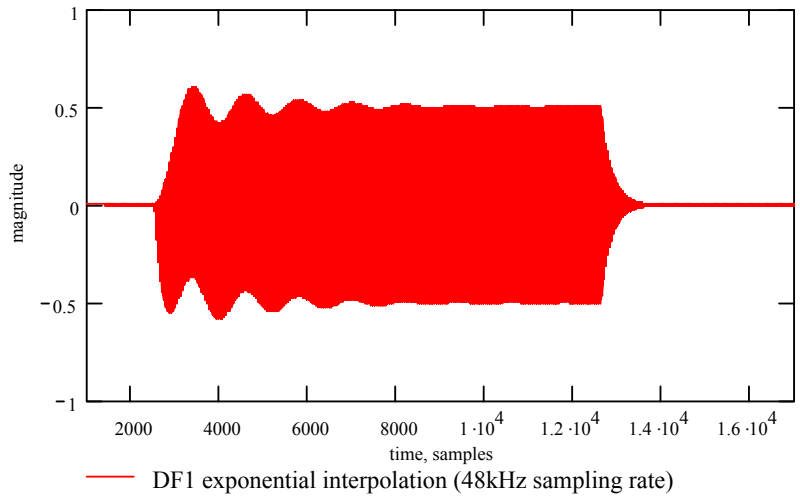

Figure 6-60 DF1 disturbance response to Scenario 5, using exponential interpolation, operating at 48 kHz, using a 10 kHz sine input excitation.

6.11.2 State change interpolation for DF2

Worst case DF2 step change disturbances were caused by the Scenario 3 test. Scenario 3 uses continuous filter parameter types and parameter interpolation can be easily implemented. Figure 6-61 to Figure 6-64 show the disturbances generated by Scenario 3, using a dc input excitation. The interpolators greatly reduce the DF2 step-change disturbance, shown earlier in Figure 6-32. The exponential interpolation produces the largest disturbance owing to the largest rate of change at the beginning of the state change B to A. Linear interpolation produces the second largest disturbance, Figure 6-62. The sinusoidal interpolation has a gradual rate of change at the start of the interpolation phase and consequently produces the smallest disturbance of the three coefficient interpolation techniques. The parameter interpolation produces the smallest disturbance, owing to its rate of change at the start being much smaller than that of any of the other interpolation techniques. The DF2 using the various coefficient interpolation schemes still produces disturbance peak magnitudes much greater than unity. The parameter interpolation scheme produces a disturbance peak magnitude of less than unity. This disturbance is of a similar magnitude to the DF1 disturbances using the coefficient interpolation schemes.

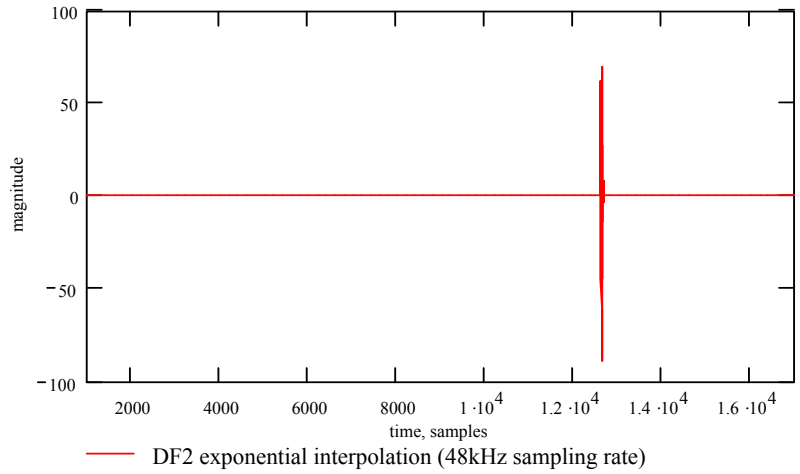

Figure 6-61 DF2 disturbance response to Scenario 3, using exponential interpolation, operating at 48 kHz, using a dc input excitation.

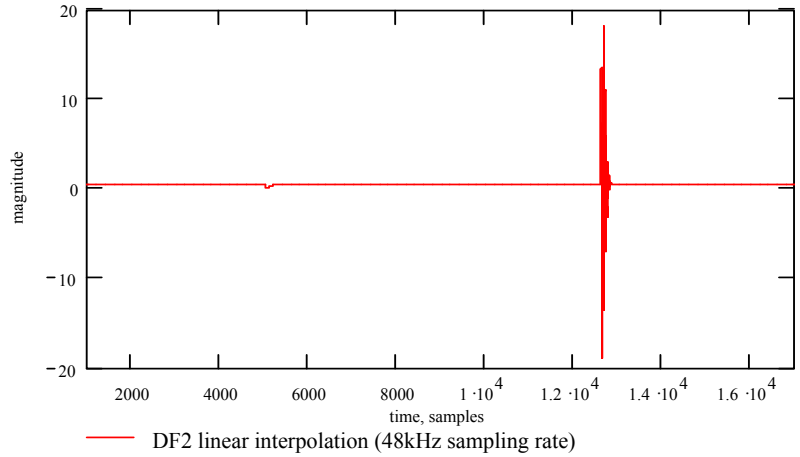

Figure 6-62 DF2 disturbance response to Scenario 3, using linear interpolation, operating at 48 kHz, using a dc input excitation.

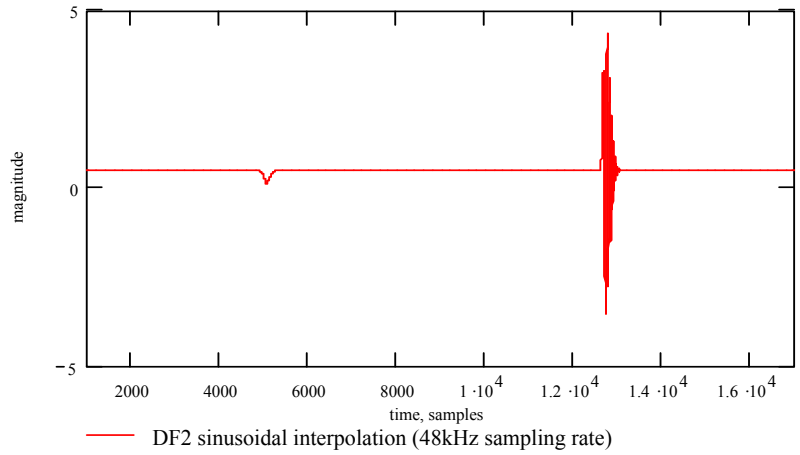

Figure 6-63 DF2 disturbance response to Scenario 3, using sinusoidal interpolation, operating at 48 kHz, using a dc input excitation.

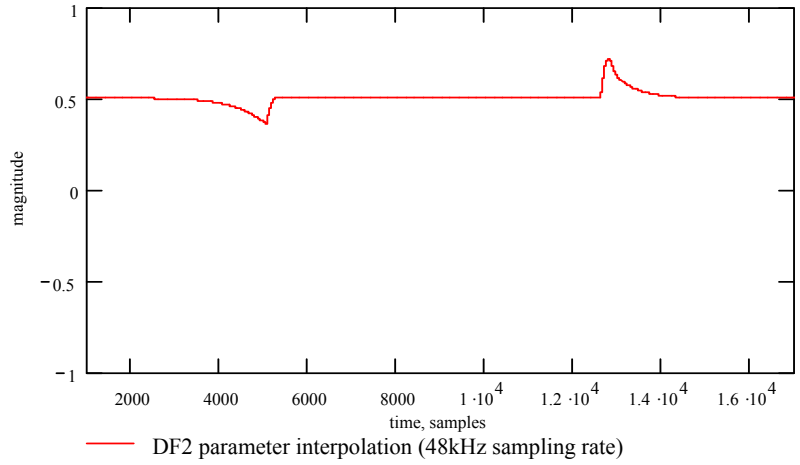

Figure 6-64 DF2 disturbance response to Scenario 3, using parameter interpolation, operating at 48 kHz, using a dc input excitation.

6.11.3 State change interpolation for other topologies

Disturbance behaviour for the various other topologies using parameter and coefficient interpolation was studied. The lattice structure produced similar disturbance behaviour to that of the DF2, discussed in the previous section. Using parameter interpolation for the Scenario 2 state change test and interpolating at the signal sampling frequency (48 kHz) the couple forms, Cabot, ladder, state-space topologies all produced negligible disturbances. Figure 6-65 shows the minimal disturbance at the output of the Zölzer structure implementing parameter interpolation for the Scenario 2 state change test, using a dc input excitation. It is evident that the Zölzer structure Scenario 2 state change disturbance, Figure 6-48, is greatly reduced through parameter interpolation.

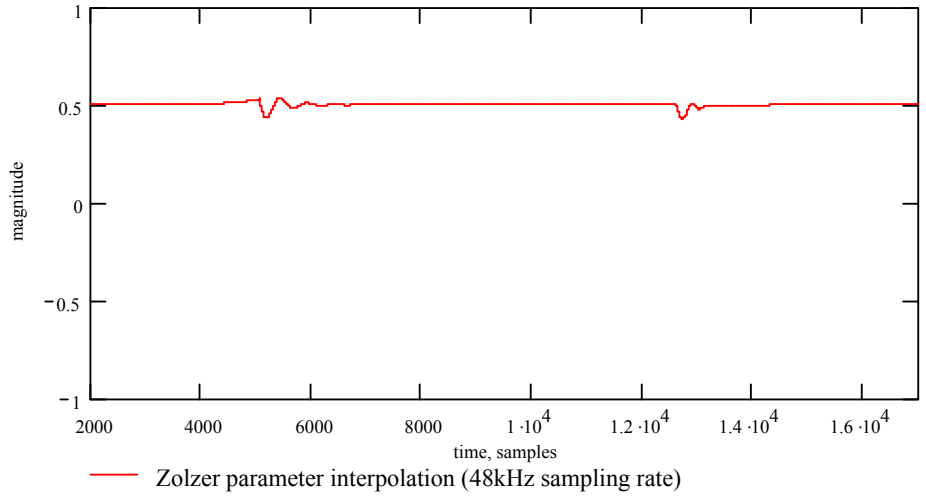

Figure 6-65 Zölzer disturbance response to Scenario 2, using parameter interpolation, operating at 48 kHz, using a dc input excitation.

The various topologies were implemented using coefficient interpolation schemes between response state changes. Coefficient interpolation has been found to be problematic for some of the topologies. Producing unstable intermediate coefficient sets. For example, the Zölzer coupled forms produces a noticeably large disturbance instability using coefficient interpolation. Linear and exponential interpolation schemes were studied and were both found to produce the large disturbance behaviour, shown in Figure 6-66.

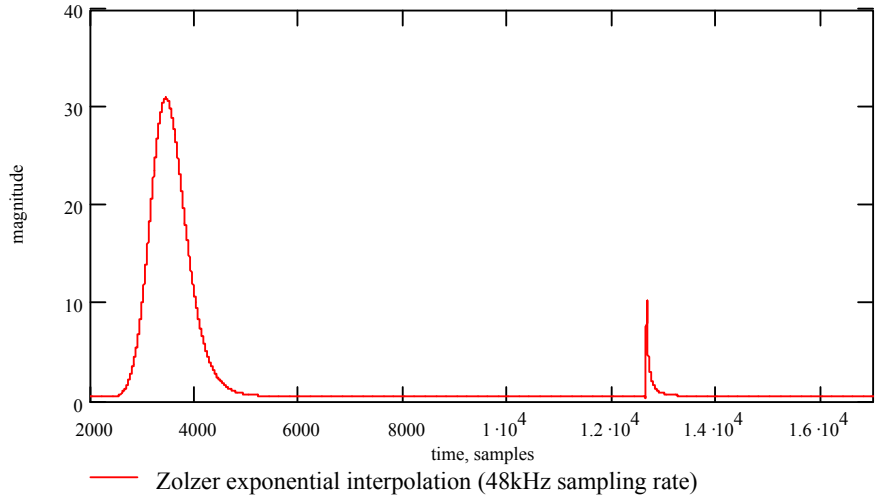

Figure 6-66 Zölzer disturbance response to Scenario 2, using exponential interpolation, operating at 48 kHz, using a dc input excitation.

The state-space topology was found to be most susceptible to instabilities using coefficient interpolation schemes. The worst-case found was for the Scenario 5 state change test. Linear and exponential interpolation scheme were both found to produce instabilities. An example of the instability during exponential coefficient interpolation, operating at 48 kHz is shown in Figure 6-67. The ladder and Cabot and other coupled forms were found not to produce noticeable instabilities using coefficient interpolation schemes for the Scenario 2 and 5 state change tests.

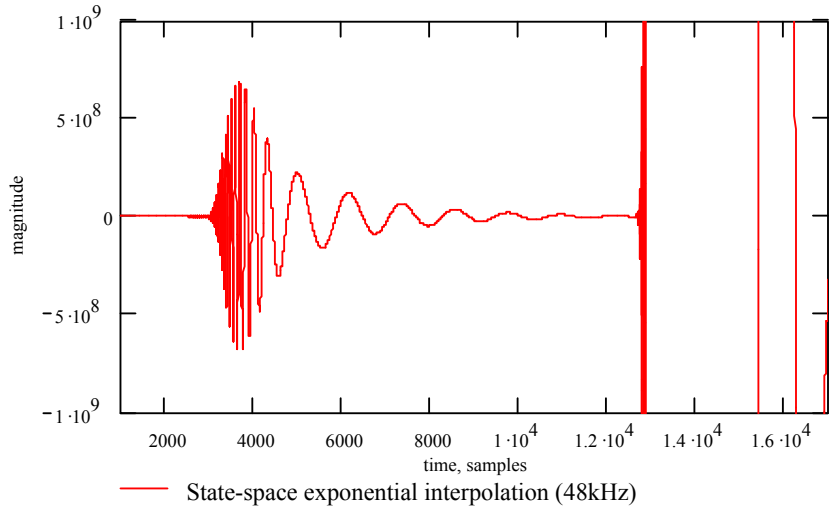

Figure 6-67 State-space disturbance response to Scenario 5, using exponential interpolation, operating at 48 kHz, using a dc input excitation.

6.12 Sub-sampled interpolators

It is common practice to sub-sample the interpolator with respect to the signal sampling frequency to reduce the computational load of the interpolator, Chapter 2 Section 2.6.6. Parameter interpolation is computationally exhaustive, involving a parameter to coefficient mapping every interpolation sample. Linear and exponential interpolators incur a lower computational overhead, but due to finite signal processing resource, sub-sampling is still common. Sinusoidal interpolators are efficiently implemented through look-up tables and sub-sampling offers small resource savings, Chapter 2, Section 2.6.4.

The aim of this section is to explore the disturbances produced in sub-sampled linear, exponential and parameter interpolation.

The sub-sampled interpolator still operates over the same fixed interpolation period since the control surface sampling interval is fixed (approximately 40 ms). Therefore the sub-sampled interpolator produces larger interpolation steps than an interpolator operating at the signal sampling frequency. For example, if the interpolator operates at one hundredth of the signal sampling frequency the resulting step sizes will be one hundred times larger than that of an interpolator operating at the signal sampling frequency. Figure 6-68 shows the sub-sampled interpolator schemes operating at a sub-sampling frequency of 480 Hz (one hundredth of the signal sampling frequency, 48 kHz). The resulting step sizes are all one hundred times larger than the step-sizes discussed in Section 6.11 and shown in Figure 6-58. This increase in step-size potentially increases the disturbance magnitude.

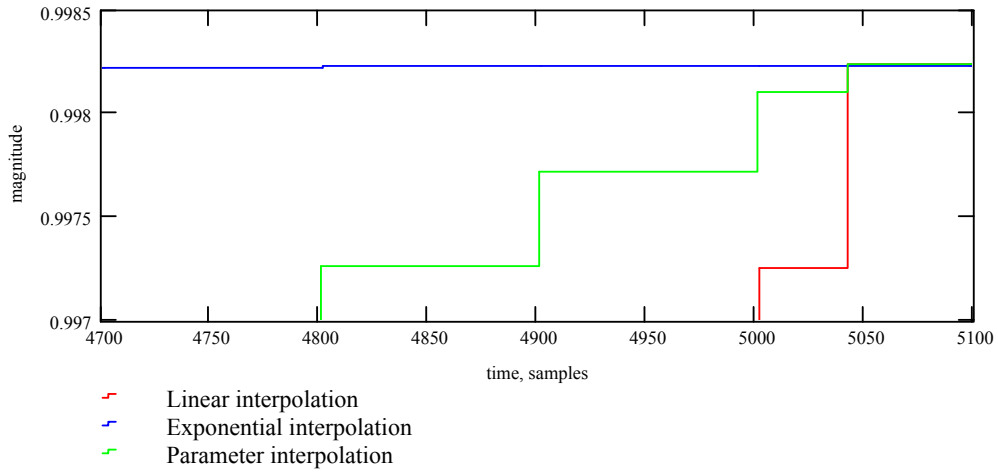

Figure 6-68 Final stages of the interpolation period for the various sub-sampled interpolation schemes - interpolator sub-sampling rate of 480 Hz.

An input sinusoid of 2 kHz was used in the last section to evaluate interpolators operating at a signal sampling frequency (48 kHz). It was found through-out the sub-sampled interpolator experiments that the linear interpolator disturbance effects are sensitive to input excitation frequency. For example, shifting the input excitation frequency from

2000 Hz to 2002 Hz produced a noticeable difference in the resulting disturbance. It was also found that the parameter interpolator disturbance was also influenced by this slight change in input excitation frequency. Despite this sensitivity to input frequency all disturbances were small, a typical disturbance shown in Figure 6-71. It is suggested that the disturbance sensitivity to such small changes in frequency can only be attributed to the input magnitude at the instance in time of the step change.

Disturbance observations of the sub-sampled interpolators, using the Scenario 2 test were conducted. The exponential interpolator was found to be insensitive to the frequency of the input excitation and produced a fairly constant level of disturbance for this test, Figure 6-69. Figure 6-70 shows the noticeably larger disturbances produced by the sub-sampled linear interpolator compared to the linear interpolator operating at 48 kHz, Figure 6-53.

Decreasing the interpolator sampling frequency to 240 Hz (200 times slower than the signal sampling frequency) produces some interesting results. The linear interpolation step-size becomes so large that the intermediate frozen-time pole and zero transfer functions produce large errors and gain fluctuations in the overall response. This results in large ringing disturbance once the interpolation period ends (ringing magnitudes of 400). This disturbance is larger than the step change disturbance (using no interpolation), Figure 6-21. It was found that the exponential and parameter interpolation disturbances only increase slightly at the sampling frequencies of 240 Hz.

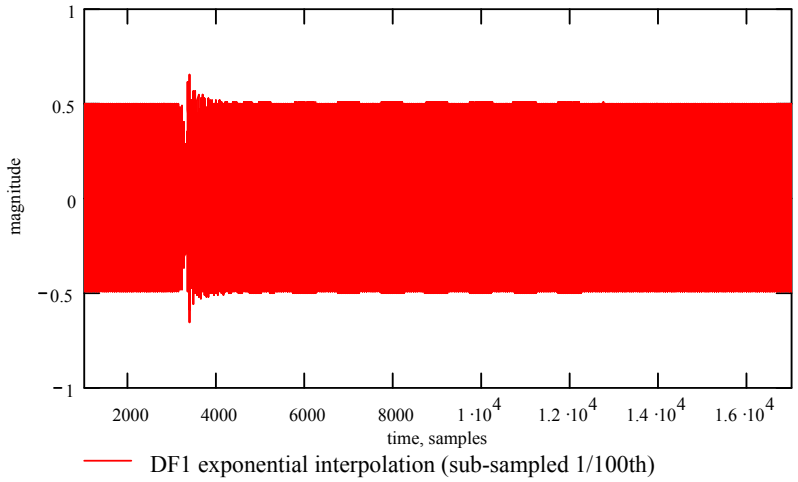

Figure 6-69 DF1 disturbance response to Scenario 2, using exponential interpolation, operating at 480 Hz, using a 2002 Hz sine input excitation.

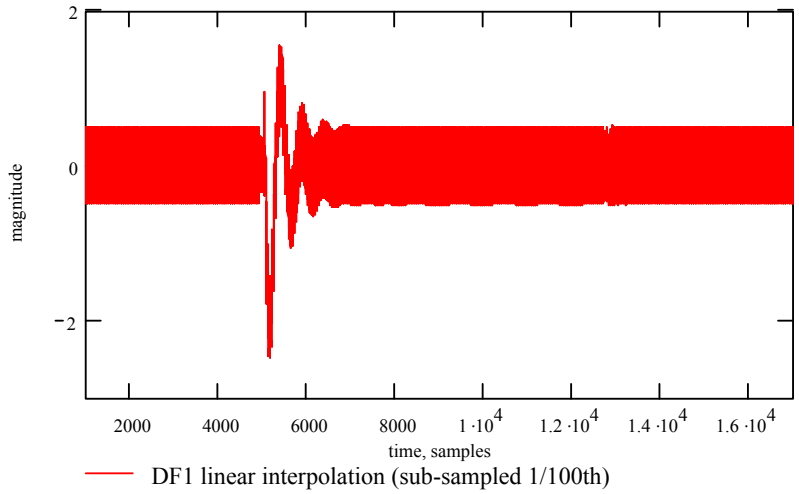

Figure 6-70 DF1 disturbance response to Scenario 2, using linear interpolation, operating at 480 Hz, using a 2002 Hz sine input excitation.

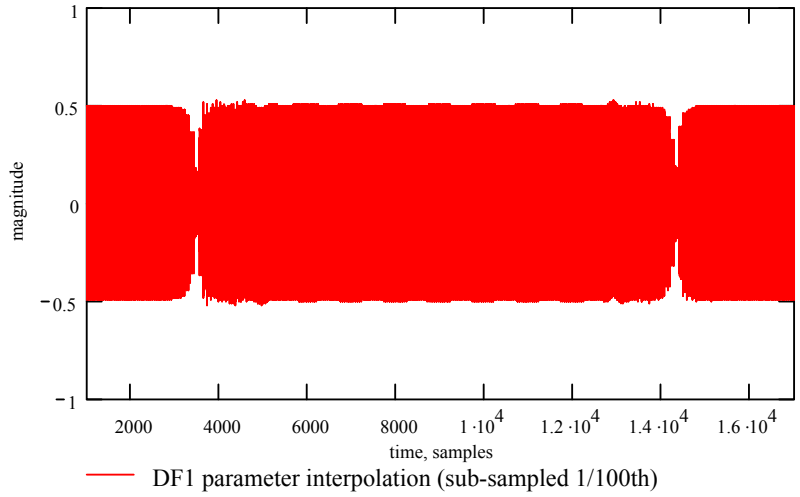

Figure 6-71 DF1 disturbance response to Scenario 2, using parameter interpolation, operating at 480 Hz, using a 2002 Hz sine input excitation.

Figure 6-72 shows the disturbance produced by the Zölzer structure using parameter interpolation sub-sampled at 480 Hz, one hundredth of the signal sampling frequency. The disturbance is noticeably worse than that of the DF1 parameter interpolation example, shown in Figure 6-71. This Kingsbury and Gold-Rader structures produced similar disturbances in that shown in Figure 6-72. Further to this the coupled forms also produce noticeable disturbance effects using dc input exictation, Figure 6-73. Note the DF1 does not produce any disturbance for this test example.

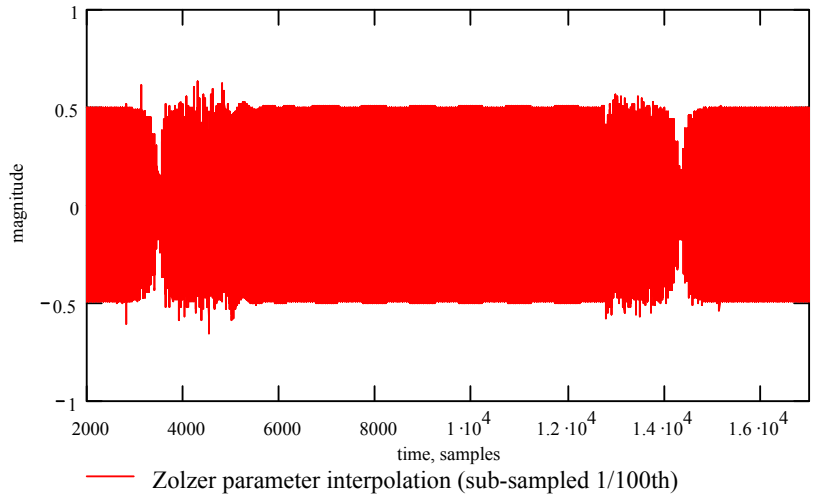

Figure 6-72 Zölzer disturbance response to Scenario 2, using parameter interpolation, operating at 480 Hz, using a 2002 Hz sine input excitation.

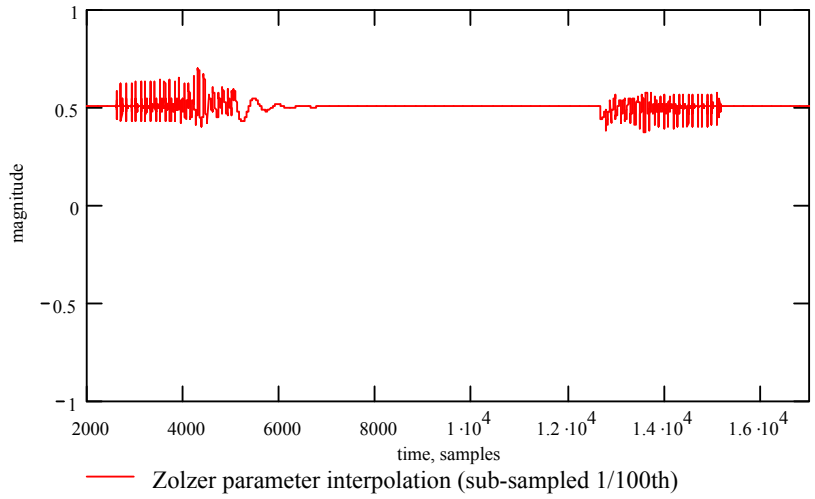

Figure 6-73 Zölzer disturbance response to Scenario 2, using parameter interpolation, operating at 480 Hz, using a dc input excitation.

The Scenario 5 test was previously shown to produce noticeable disturbance for DF1 using the exponential interpolator, Figure 6-60. Figure 6-74 shows the resulting disturbance for the DF1 sub-sampled exponential interpolator operating at 4800 Hz. The disturbance is considerably larger than the exponential interpolator operating at 48 kHz, Figure 6-60 indicating that the exponential interpolator disturbances are sensitive to sub-sampling frequencies.

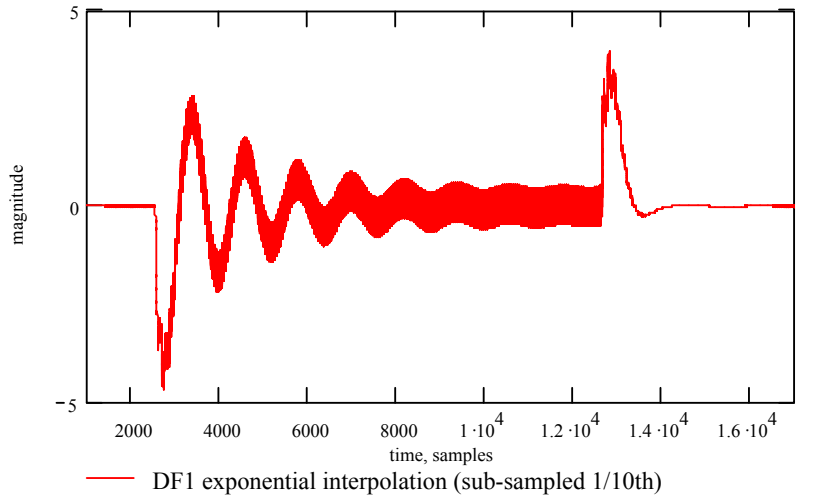

Figure 6-74 DF1 disturbance response to Scenario 5, using exponential interpolation, operating at 4800 Hz, using a 10 kHz sine input excitation.

6.13 Interpolator finite wordlength issues and related disturbance effects

The implementation of interpolator algorithms under finite wordlength arithmetic produces errors in the interpolated coefficients. Since most interpolators are recursive, using the previous interpolated value, errors can accumulate, resulting in a final error at the end of the interpolation period. For sensitive coefficients, filters with poles and zeros close to dc or the Nyquist frequency, any small error in a static coefficient set can have a large effect in the resulting pole zero positions on the z-plane and therefore the frequency response. Since the interpolator must produce the exact final value coefficient, 'clamping' techniques are commonly used to detect that the interpolation period is finished or, in the case of the exponential ramp, that the steady state error has been reached. Once these states have been detected the exact final value coefficient (target coefficient) can be placed directly into the interpolator system to eradicate any accumulated error in the coefficient. Therefore coefficient 'clamping' schemes are employed to correct any interpolator finite wordlength effects and frequency response errors. A side effect of coefficient clamping are the potential step-changes at the end of the interpolation period, which have the potential to cause disturbance. This section aims to implement the interpolator schemes under various forms of finite wordlength arithmetic. The resulting finite wordlength errors and associated clamping errors will be examined. Disturbance differences resulting from interpolator implementation under finite wordlength arithmetic will be examined.

6.13.1 Linear interpolator finite wordlength effects

A quantisation model for the linear interpolator is shown in Equation 6-4. There are three sources of quantisation. The fixed increment (step size, δ) quantised by the memory wordlength or the accumulator input wordlength. Accumulation quantisation (Qacc), relevant to floating point arithmetic and quantisation due to the storage wordlength of c_i.

$$\delta q \leftarrow Q(\delta)$$
$$c_i \leftarrow Q\left(Qacc\left(c_{i-1} + \delta q\right)\right)$$

(6-4)

Linear interpolation was implemented using the various fixed and floating point arithmetic functions, described in Chapter 4. Floating point arithmetic was not found to produce a smaller final value error then fixed point arithmetic. By representing the step size, δ, and state variables, c_i, in 24 bit fixed point double precision (46 fractional bits) and 40 bit floating point extended precision (31 fractional bits), the final value error accumulated at the end of the interpolation is negligible compared to the step-size. Implementing the linear interpolator in fixed or floating point, single precision, the final value error is of a similar magnitude to the step-size. A typical step size, δ, being in the region of 10^{-4} (for an interpolation period of 50ms at a 48 kHz sampling frequency), Equation (2-14). Further inspection of linear interpolated coefficient sets shows that the final value errors range from fractions of the step-size magnitude to approximately double the typical step size, Figure 6-75. The use of rounding, as opposed to truncation, was found to reduce the final error step by a factor of two. However, the linear interpolation error is accumulative, rounding could produce coefficient data greater in magnitude than the ideal coefficient. Therefore the use of rounding in linear interpolation for extremity coefficient regions, for example 0.9999997, ($1-2^{-23}$) could cause overflows and instabilities.

The finite wordlength linear interpolator, using single precision fixed point arithmetic was used for a Scenario 2 test, using the DF1 with a 2 kHz sine input stimulus. The resulting disturbance was very similar to the disturbance for the same test using the ideal interpolator, Figure 6-53. The disturbance difference (the difference between the ideal interpolator and finite wordlength interpolator disturbances) is shown in Figure 6-76.

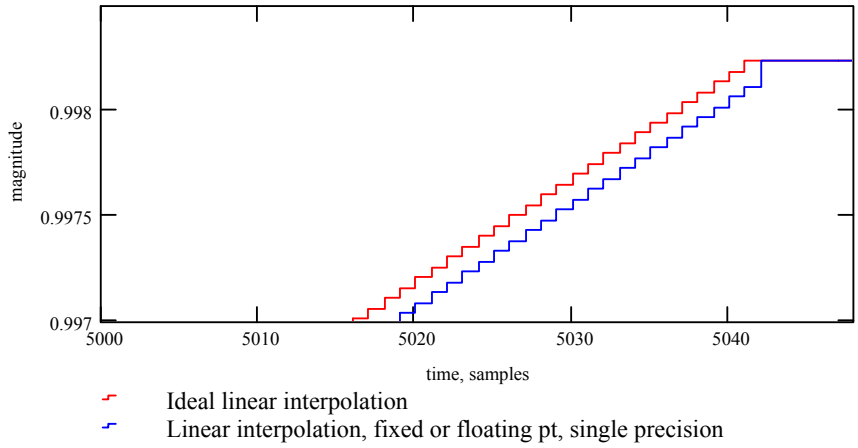

Figure 6-75 Finite wordlength effects on a linear interpolated coefficient, using single precision 24 bit fixed or 32 bit floating point arithmetic.

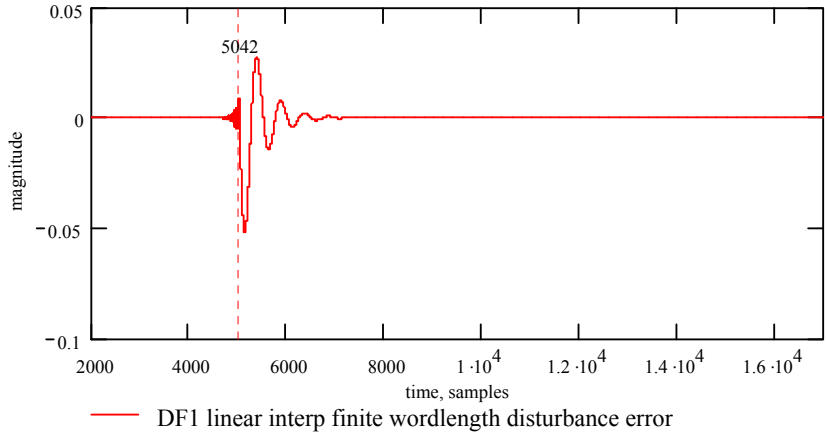

Figure 6-76 Difference in the DF1 disturbance response between an ideal linear interpolator and a linear interpolator, implemented in single precision 24 bit fixed point arithmetic (Scenario 2 test using a 2 kHz input excitation).

6.13.2 Exponential interpolator finite wordlength effects

There are several ways to implement the exponential interpolator algorithm. Three techniques, of similar computational efficiency, are shown in (6-5).

$$c_i \leftarrow c_{i-1} + k\cdot\left(target_i - c_{i-1}\right) \quad \text{A}$$

$$c_i \leftarrow c_{i-1} - k\cdot c_{i-1} + k\cdot target_i \quad \text{B}$$

$$c_i \leftarrow \left[c_{i-1}\cdot(1-k)\right] + \left(k\cdot target_i\right) \quad \text{C} \qquad (6\text{-}5)$$

The three techniques were implemented using the finite wordlength emulation functions described in Chapter 4. Figure 6-77 shows the finite wordlength effects of 24 bit single precision fixed point on the various exponential interpolators. As the 'current' value coefficient tends towards the 'target' final value coefficient the step change gets exponentially smaller. As the diminishing step-size tends to within the realms of the finite wordlength limits the current coefficient no longer changes and reaches, what is often termed, a steady state error. The steady state error is dependent on the finite wordlength and the time constant, k. For example, as k approaches unity (small time constant) the error approaches zero. Thus the smaller k is, the larger the steady state error.

Implementation C was found to produce a cyclic/hunting effect once the interpolator had been clamped at the final value. This is due to quantisation destroying the relationship between the coefficients, k and 1-k, resulting in non-static behaviour after the interpolator reaches the clamped target value. Implementation C was therefore not favoured. Implementations A and B produce near identical results with no cyclic/hunting effects. Implementation B was also implemented using a rounding quantiser for the final result c_i. The rounding did produce the smallest final value error as shown in Figure 6-77. However finite wordlength exponential interpolation towards zero, behaves differently to interpolation towards one. Truncation accelerates the interpolation towards zero, increasing the final step sizes and eliminating the steady state error and associated clamping step. However, coefficients in pole transfer functions with high gain tend to have coefficient magnitudes approaching one or two (depending on scaling).

Exponential interpolation was also implemented under both sign magnitude and twos-complement 32 bit single precision floating point arithmetic. Both were found to produce the same final value errors. It was found that the 32 bit single precision floating point steady state error was typically half of that produced through 24 bit single precision fixed point arithmetic.

The ideal exponential interpolator was compared to a 24 bit single precision fixed point implementation, using the Scenario 5 test. The disturbance differences were small. However this does not represent a worst case test for the finite wordlength interpolator. The clamping error is at the end of the interpolation period. The Scenario 5 test has large amounts of gain at the start of the interpolation period.

Using the Scenario 2 test, the 24 bit single precision fixed point exponential interpolator also produces an extremely small difference in disturbance compared to the ideal interpolator, Figure 6-78. It can be concluded that the finite wordlength effects of the exponential interpolator cause subtle differences in the disturbance at the end of the interpolation period. These finite wordlength disturbance differences are small compared to the disturbances caused at the start of the exponential interpolation period in the Scenario 5 test.

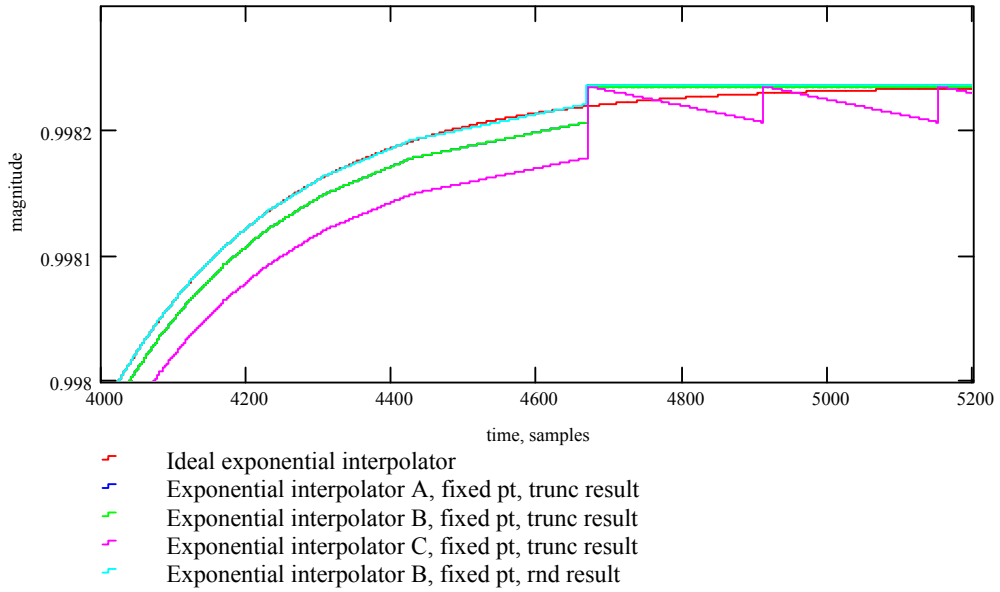

Figure 6-77 Final stages of the interpolation period for various fixed point implementations of the exponential interpolator.

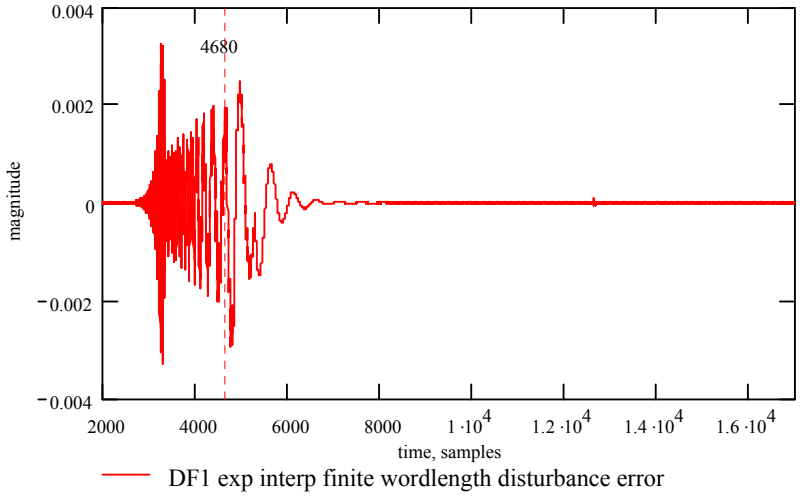

Figure 6-78 Difference in the DF1 disturbance response between an ideal exponential interpolator and an exponential interpolator implemented in single precision 24 bit fixed point arithmetic (Scenario 2 test using a 2 kHz input excitation).

6.13.3 Sinusoidal interpolator finite wordlength effects

As discussed in Chapter 2, Section 2.6.4 sinusoidal interpolation is efficiently implemented through the use of a look-up table, which circumvents problems associated with finite wordlength. Implementation of the sinusoidal interpolation using an on-line approximation is computationally intensive and is sensitive to quantisation effects. However a quantisation model for the algorithm suggested for sinusoidal interpolation in Chapter 2, Section 2.6.4 is given in Equation (6-6). For the sampling rates and interpolation periods discussed in this work the value of 'Const' can typically be of the order of 10^{-10}, see Section 2.6.4 . Implementation in 24 bit single precision fixed point arithmetic results in the interpolator not operating. Implementation in 24 bit double precision fixed point arithmetic produces an operable interpolator with minimal clamping error. Implementation in 32 bit single precision floating point, produces an operating interpolator since the floating point arithmetic can efficiently represent 'Const' and variable 'ddv'. The clamping error in a sinusoidal interpolator implemented in 32 bit floating point arithmetic is shown in Figure 6-79. The disturbance difference produced by

32 bit floating point sinusoidal interpolation as opposed to the ideal sinusoidal interpolator using the DF1 Scenario 2 test, with a 2 kHz sine input is shown in Figure 6-80. The sinusoidal interpolator disturbance is mainly caused by the large steps in the middle of the interpolation period, Section 6.11. Therefore the disturbances through the clamping error are minimal.

$$\begin{aligned}&\text{for } i \in \text{MaximumSample}\\&\quad v_i \leftarrow Q\left(v_{i-1} + dv_{i-1}\right)\\&\quad dv_i \leftarrow Q\left(dv_{i-1} + ddv_{i-1}\right)\\&\quad ddv_i \leftarrow Q\left(ddv_{i-1} + \text{Const}\right)\\&\text{return } v\end{aligned} \tag{6-6}$$

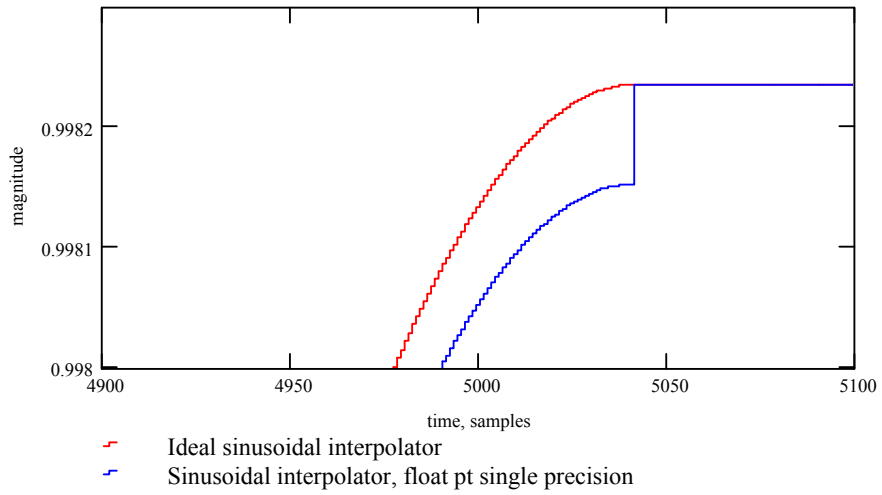

Figure 6-79 Final stages of the interpolation period, showing the clamping error, for a 32 bit floating point implementation of the sinusoidal interpolator.

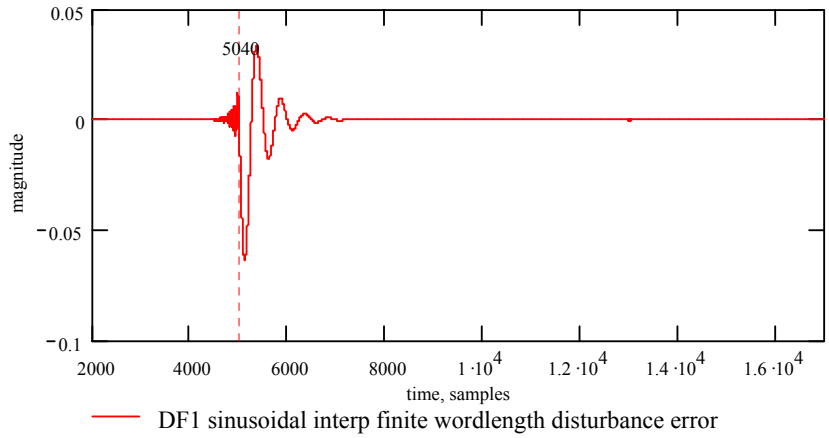

Figure 6-80 Difference in the DF1 disturbance response between an ideal sinusoidal interpolator and a sinusoidal interpolator, implemented in single precision, 32 bit float point arithmetic (Scenario 2 test using a 2 kHz input excitation).

6.13.4 Parameter interpolator finite wordlength effects

The numerical precision of the parameter to coefficient mapping is assumed to be far greater than that of the coefficient wordlength to avoid the introduction of any static frequency response errors, see Section 2.4.2 . Coefficient quantisation errors under a fixed point 24 bit wordlength are in the order of 10^{-7}. Coefficient quantisation is not likely to introduce any noticeable disturbance effects using parameter interpolation schemes, since the most sensitive coefficient steps through parameter interpolation are typically of the order of 10^{-6} (assuming filters tuned to low frequencies with respect to the sampling frequency). Figure 6-81 shows the disturbance difference for an ideal parameter interpolator (using no coefficient quantisation functions) and a parameter interpolator passing its data through a 24 bit fixed point coefficient quantiser. The test uses the Scenario 2 test, the DF1 topology, using a 2 kHz sine input stimulus, shown in Figure 6-81. The difference in disturbances is negligible.

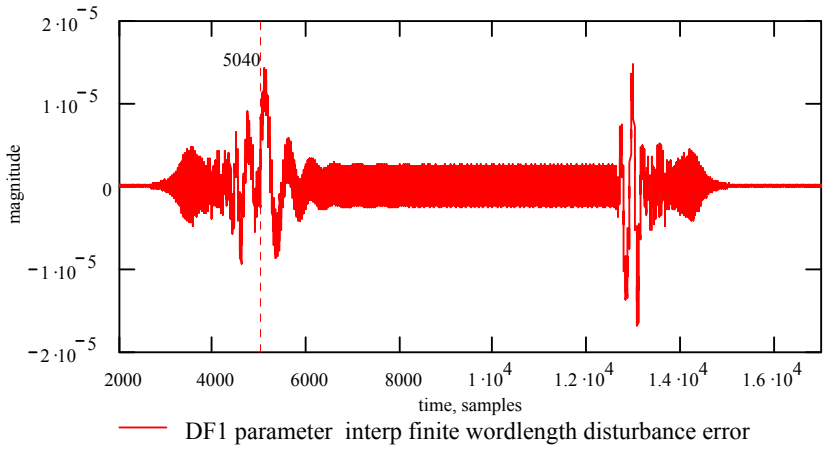

Figure 6-81 Difference in the DF1 disturbance response between an ideal parameter interpolator and a parameter interpolator, quantised to 24 bits (Scenario 2 test using a 2 kHz input excitation).

6.14 Disturbance effects at higher sampling frequencies

This section examines the disturbance effects of time varying filters at the higher standardised sampling rates of 96 and 192 kHz. Figure 6-10 displays the pole transfer functions for the Scenario 2 test, state A and B filters, at a sampling frequency of 48 kHz. The state B filter setting produces a maximum pole gain of approximately 83 dB, in the 100 Hz region. At a sampling frequency of 96 kHz the maximum peak pole gain for the same filter is 95 dB and at a sampling frequency of 192 kHz, the peak pole gain is 107 dB. This suggests the potential of larger magnitude disturbances for filter state changes operating at higher sampling frequencies. This is corroborated by the pole transfer function response to the Scenario 2 step state changes, under dc input, operating at 192 kHz, Figure 6-82. The disturbance is sixteen times (24 dB) larger than that of the same test at a sampling frequency of 48 kHz, Figure 6-27.

However, these pole response measurements are generated from dc inputs. As discussed earlier in Section 6.4 the overall filter state change disturbance is not just dependent on pole path gain but also dependent on signal level changes feeding the pole paths. The difference between adjacent sample instances for a 96 kHz sampling system is half that of

a 48 kHz sampled system (the rate of change between two samples is halved because the sampling frequency is doubled). Thus a signal disturbance in a 96 kHz sampled system has half the magnitude effect at the accumulation of the pole path state variables of that in a 48 kHz sampled system. This consequently has effects in the overall disturbance response. This section also aims to examine finite wordlength effects of interpolators operating at higher sampling rates.

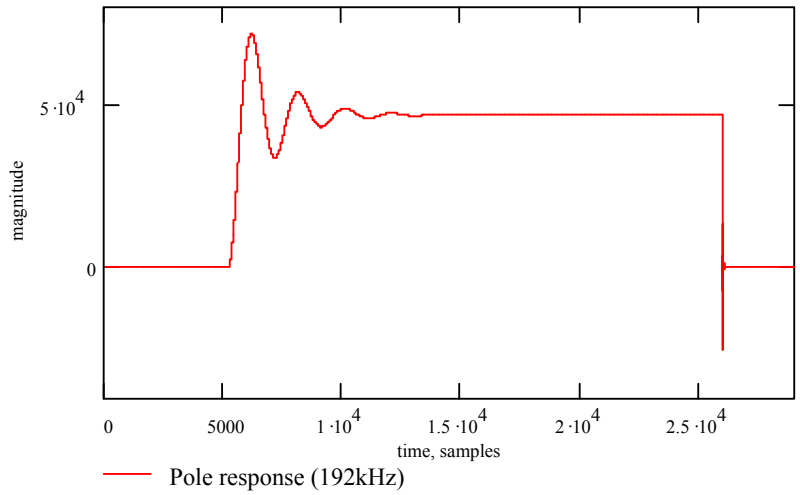

Figure 6-82 Direct Form pole response disturbance to Scenario 4, using dc input excitation operating at a sampling frequency of 192 kHz.

6.14.1 DF1 at higher sampling frequencies

Scenario 2 step state change observations were made, at a sampling frequency of 48 kHz for DF1 in Section 6.4.2, Figure 6-21, producing a peak to peak disturbance magnitude of approximately 37. It was found that 10 kHz sine input excitation produced the maximum level change at the filter output and this subsequently resulted in the largest disturbance. Using the Scenario 2 test the DF1, operating a sampling frequency of 96 kHz, produces a peak to peak magnitude disturbance of 39. Despite a 12 dB increase in pole gain for a doubling in sampling frequency (96 kHz), this is a similar disturbance to the same test conducted at a sampling frequency of 48 kHz.

However, it is not sensible to assume higher sampling rates cannot produce larger disturbances than systems sampling at 48 kHz. The increase in the pole gain at the higher sampling frequencies can produce larger disturbances if the system excitation frequency is higher. Furthermore systems operating at a higher sampling frequency have more signal bandwidth and accommodate higher frequency input signals.

A modified 96 kHz, Scenario 2 test was conducted to illustrate this. The input sinusoid was increased to 20 kHz. In order to produce a 10 dB change in output magnitude, the state A filter was changed from a centre frequency of 10 kHz to 20 kHz. It must be noted that the change of the state A centre frequency to 20 kHz does not increase the state A pole transfer function gain. Figure 6-83 shows the results of the modified Scenario 2 test operating at a sampling frequency of 96 kHz. The disturbance has a peak to peak magnitude disturbance of 79. It must be noted that larger disturbances are also possible by increasing the input signal frequency in a 48 kHz sampled system. The modified Scenario 2 test operating at 48 kHz (state B centre frequency of 20kHz) with a 20 kHz input sinusoid produces a peak to peak disturbance of 57.

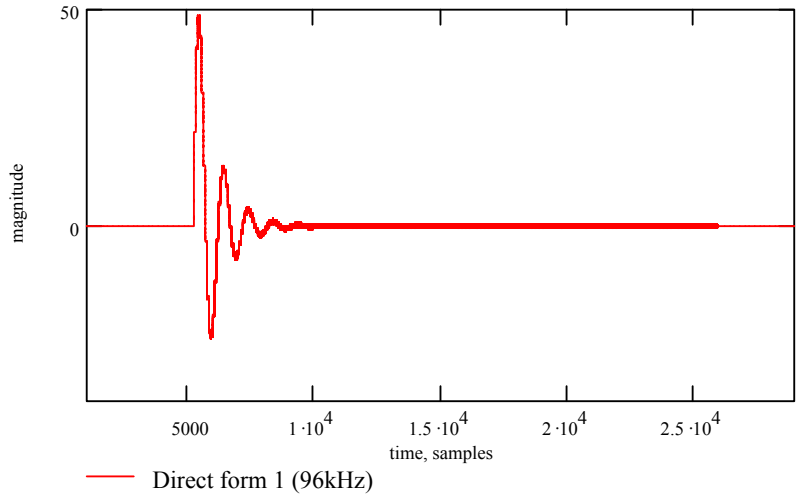

Figure 6-83 DF1 disturbance response to the modified Scenario 2 test, using 20 kHz sine input excitation, system operating at a sampling frequency of 96 kHz.

The disturbance effects for an exponential interpolator operating at a sampling frequency of 96 kHz were examined. Worst-case disturbances using the exponential interpolator were previously found using the Scenario 5 test, Figure 6-60. Figure 6-84 shows the disturbance produced in the DF1 Scenario 5 test using an exponential interpolator implemented in 24 bit fixed point arithmetic, operating at a sampling frequency of 96 kHz. Comparison with the same test conducted at a sampling frequency of 48 kHz, Figure 6-60, shows that the increase in sampling frequency has not produced any noticeable increase in disturbance magnitude. This suggests that finite wordlength effects on the exponential interpolator at a sampling frequency of 96 kHz does not increase disturbance magnitude. This is of interest since the increase in sampling frequency has effectively increased the clamping error and coefficient sensitivity.

The disturbance produced for the Scenario 2 test linear interpolator implemented in 24 bit fixed point, operating at a sampling frequency of 96 kHz, is shown in Figure 6-85. This disturbance is approximately thirty percent larger than the identical test operating at 48 kHz, Figure 6-53. It is suggested that this is due to the larger interpolation steps operating at the sampling frequency of 96 kHz.

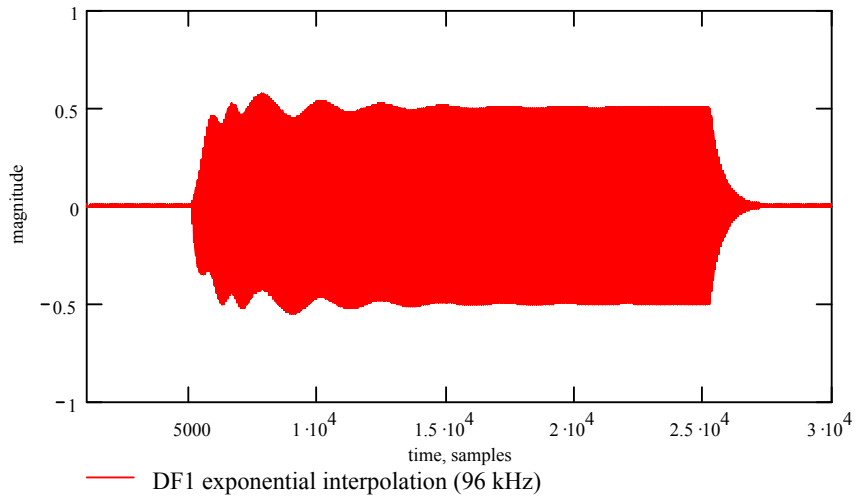

Figure 6-84 DF1 disturbance response to Scenario 5, using exponential interpolation, implemented in 24 bit fixed point, operating at 96 kHz, using a 10 kHz sine input excitation.

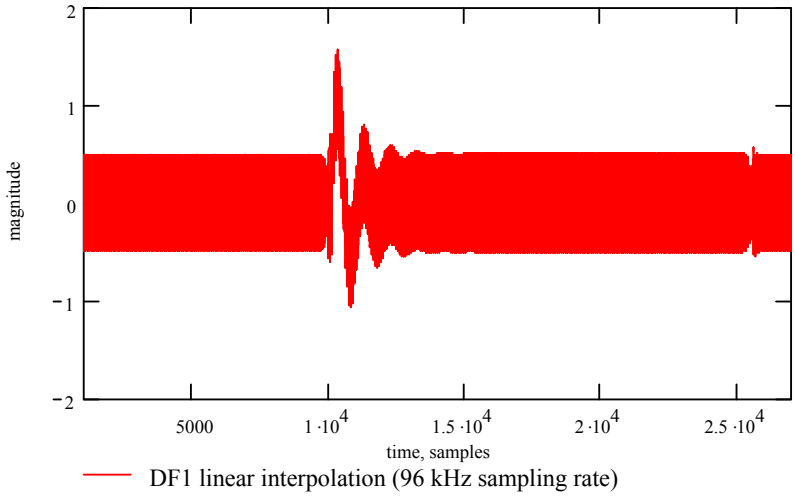

Figure 6-85 DF1 disturbance response to Scenario 5, using linear interpolation, implemented in 24 bit fixed point, operating at 96 kHz, using a 2 kHz sine input excitation.

6.14.2 Other topologies at higher sampling frequencies

The coupled form (Gold-Rader, Zolzer and Kingsbury) structures disturbance behaviour to state change were studied at the higher sampling frequencies. Their disturbance magnitudes for the Scenario 5 test, using a dc input excitation, operating at a sampling frequency of 96 kHz, are shown in Figure 6-86. Comparison of Figure 6-86 with the equivalent test at 48 kHz sampling frequency, Figure 6-47, shows that the disturbance magnitude decreases with an increase in sampling frequency. It must be noted that this is under dc excitation, which is the worst-case signal for disturbance evaluation at higher sampling frequencies (due to the rate of change between samples for ac signals).

The Cabot structure also produces a similar reduction in disturbance. At a sampling frequency of 48 kHz the Scenario 5 test produces a peak to peak disturbance of 0.76, Figure 6-49. The peak to peak disturbance magnitude is 0.68 for the same test at a sampling frequency of 96 kHz and 0.57 at 192 kHz.

A doubling in sampling frequency in the lattice (Massie) and DF2 structures, using a dc input excitation, increases the peak to peak disturbance magnitude by approximately 40

percent. However the ladder and state-space structures operating at a sampling frequency of 96 kHz, using a dc input excitation produce similar disturbance to ladder and state-space structures operating at a sampling frequency of 48 kHz.

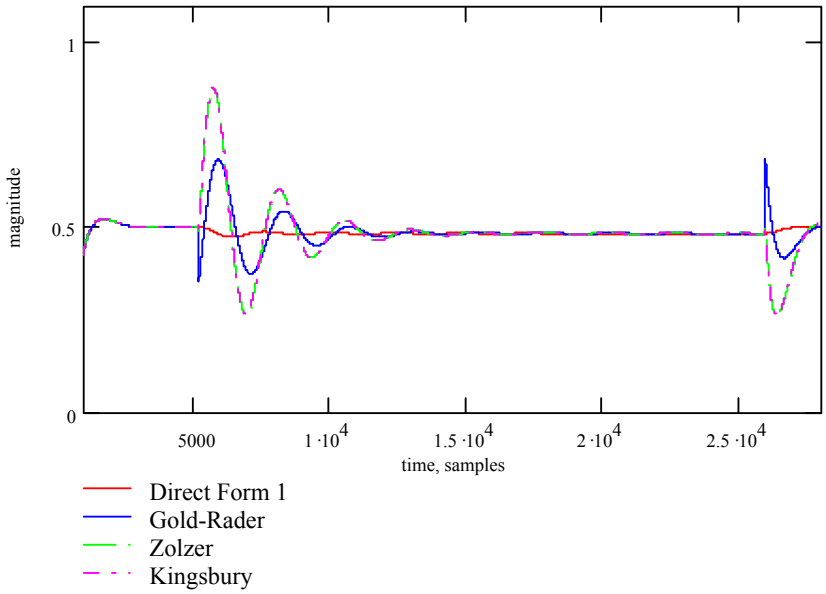

Figure 6-86 DF1 and coupled form disturbance responses to Scenario 5 state change test, operating at 96 kHz, using a dc input excitation.

6.15 Discussion

It is evident that signal disturbances are caused by magnitude changes at the DF1 output – the source of the pole paths. Critical disturbances occur in instances of high gain in the target pole transfer function (typically low frequency high Q filters). State change scenarios 2, 4 and 5 are shown to produce large disturbances (peak disturbance magnitudes in the region of 80).

'Pole before zero' topologies, for example DF2, produce larger disturbances. Scenario 3 produce peak disturbance magnitudes of 2000. DF2 disturbances are caused by gain changes in the pole transfer function, producing ringing at the pole output. The pole response ringing may be attenuated by the zero transfer function depending on subsequent

zero transfer function attenuation characteristics. Large pole gain changes occur between low and high frequency tuned filters. Where dc excitation causes maximum pole disturbance and a high frequency tuned target response, provides minimal attenuation in the zero transfer function.

DF1T is a 'pole before zero' topology, producing similar disturbance behaviour to DF2. DF2T is a 'zero before pole' topology, producing similar disturbance behaviour to DF1. It is shown that unstable coefficient sets occur under time varying conditions in transposed forms. This can result in large spike disturbances. A coefficient delay compensation scheme is developed to re-align the coefficients in the transposed topologies. The resulting disturbance behaviour of DF1T and DF2T using the compensation scheme is identical to DF2 and DF1 respectively.

The coupled forms (Gold-Rader, Kingsbury, Zölzer) produce similar disturbances to DF1 for the five scenario tests, for a 10 kHz input stimulus. However using a dc input the coupled forms produce large disturbances, whereas the DF1 produces negligible disturbance. Kingsbury and Zölzer topologies produce the largest disturbances under dc input. This is thought to be attributable to the integrators in their pole transfer function implementations.

The lattice are 'pole before zero', producing similar disturbance responses to DF2. Ladder implementations (Moorer and Massie) are found to produce similar disturbance responses. However the ladders accumulator L_2 scaling reduces its disturbance peak magnitudes by a factor of 400, compared to DF2. The lattice has unbounded nodes and produces state change disturbances of the same magnitude as DF2. The state-space topology was found to produce similar disturbance behaviour to the ladder structures. However, using dc excitation the state-space topology produces similar disturbances to the Zölzer and Kingsbury topologies. The Cabot structure produces similar disturbances to the Gold-Rader topology.

State change disturbance reduction through coefficient and parameter interpolation schemes is examined. Linear coefficient interpolation produces a continuously large step increment and is most susceptible to disturbance. A pole transfer function producing high gain at any instance in time is likely to be excited by this increment. Exponential interpolation produces the largest step-size of the schemes, at the beginning of the interpolation period. This large step coincides with the initial high gain pole transfer function in the Scenario 5 test, producing a critical disturbance. However exponential and sinusoidal interpolation appeared to be produce less signal disturbance than linear interpolation. Parameter interpolation produced the best disturbance rejection. Using parameter interpolation, operating at 48 kHz, the DF1 was found to produce minimal signal disturbances. Parameter interpolation reduces the DF2 state change disturbance (peak magnitude of 2000) to a peak magnitude of 0.7. The DF2 state change disturbance is also greatly reduced by the coefficient interpolation schemes. However the resulting DF2 peak magnitude disturbances for all of the coefficient interpolation schemes are still considerably larger than unity (worst case peak magnitude of 70). Coefficient interpolation schemes did however generate large disturbances in the Zölzer and state-space topologies, due to intermediate coefficient sets producing instability. Whereas parameter interpolation evidently cannot produce unstable coefficient sets.

Sub-sampled interpolators, operating one tenth and one hundredth of the signal sampling frequency, 48 kHz, produced noticeably larger disturbances. In particular the exponential interpolator operating a 4800 Hz produced disturbances over ten times the magnitude of the 48 kHz interpolator disturbances. The finite wordlength effects of the interpolation schemes were investigated. Final value ‘clamping ‘ errors at the end of the interpolation periods were found to produce negligible differences in the disturbance behaviour.

Disturbance behaviour of filters at higher signal sampling frequencies were examined. It is shown doubling the sampling frequency increases the direct form pole transfer function

gain by 12 dB. However the doubling of the sample frequency, halves the magnitude changes between adjacent samples and thus half the amount of energy change in the pole paths. Using identical test parameters for a 48 kHz and 96 kHz sampling filter, the resulting disturbance magnitudes are the same. However if the input excitation frequency is also doubled then the DF1 disturbances are considerably larger at a sampling frequency of 96 kHz. Coupled forms and the Cabot structure produce smaller disturbance magnitudes for an increase in sampling frequency, under dc input excitation. The lattice and DF2 structures produce increases in disturbance magnitude for an increase in sampling frequency, under dc input excitation. The ladder and state-space disturbance response appear to be insensitive to sampling frequency, producing the same disturbance magnitude at 96 kHz that was produced at a 48 kHz sampling frequency.

7 Review, future work and conclusions

This chapter gives a review of the three main topics in this work, namely, coefficient calculation, topology behaviour under finite wordlength arithmetic and topology behaviour during coefficient update. Future work is also suggested and conclusions of the project are given.

7.1 Review

7.1.1 Techniques for calculating digital equaliser coefficients

The work described in Chapter 3 is an investigation of coefficient calculation schemes for discrete-time audio equalisers. The aim of the work was to minimise filter response distortion and increase computational efficiency for on-line implementation. Magnitude and phase response distortions for BZT and MZT–based filters were examined. Offset response distortions in the MZT-based filters have been identified and offset correction schemes developed for the LF, HF shelving and bell filters. Image response distortions (excessive peak gain) are found in the MZT-based bell filter. Correction schemes were developed to improve the MZT bell filter image response distortion. It is shown that pre-warping the peak gain in the MZT improves the magnitude frequency response with a relatively affordable increase in transform computation. It is also shown that for high frequency tuned filters the stability and noise characteristics of the MZT-based filter is superior to that of a BZT-based filter. This is even true for a flat (unity gain) filter.

Filter response distortions using existing BZT pre-warping techniques at a sampling frequency of 48 kHz were investigated. BZT-based LF shelving filters produce negligible response distortion and are computationally efficient. BZT pre-warping techniques produce large response distortions in HF shelving filters. Prewarping Fc and Q is found to be the optimal BZT technique for the bell filter. However, prewarping Fc is found to be the optimal BZT technique for LPF and HPF.

MZT-based bell, LF and HF shelving filters have been shown to produce less response distortion than any of the BZT techniques at sampling frequencies of 48 kHz, 96 kHz and 192 kHz. However, the response distortions are considerably smaller at the higher sampling frequencies. At sampling frequencies of 96 and 192 kHz, it was found that the MZT pre-warped peak gain technique is redundant and the MZT (with offset scaling) is sufficient since there is no significant overshoot in the peak gain. At a sampling frequency of 96 kHz the BZT (pre-warped Fc/Q) response distortions are extremely small. At a sampling frequency of 192 kHz the BZT (pre-warping Fc/Q) and MZT (with offset scaling) response distortions are negligible furthermore the BZT (without pre-warping) could be used, reducing the computational load considerably. However, the bell filter response error at a sampling frequency of 96 kHz for the BZT (no pre-warping) is considerable.

The high frequency magnitude and group delay/phase errors discussed in this work are small, where the ear is less sensitive to distortion, (Everest, 1994). It has not been an objective of this work to assess audible perception of magnitude and phase distortions. However, this is an important task and is described in the section of future work.

The work has produced an analytical account of the response distortions and computational efficiency for the various MZT and BZT schemes for filters operating at sampling frequencies of 48, 96 and 192 kHz. This is significant in the understanding of the performance of these coefficient calculation schemes. Further, the work has led to the

development of techniques to improve the MZT response distortion. It has been demonstrated that MZT shelving and bell filters can approximate more closely to their s-plane parents than BZT-based filters. This part of the project has led to a conference paper, (Clark et al, 1996) and has been published as a journal paper, (Clark et al, 2000), included in Appendix E.

7.1.2 Topology behaviour under finite wordlength arithmetic

In Chapter 5 the investigation into the behaviour of static frequency response filter topologies under various forms of finite wordlength arithmetic is described. The aim of the work was to assess the noise behaviour of filter topologies under finite wordlength constraints, using various input stimuli, operating at sampling frequencies of 48, 96 and 192 kHz.

Direct form 'zero before pole' topologies produce poor noise performance realising filter responses with high pole gain (typically low frequency, high Q filters). The dc offsets produced in twos-complement truncation noise can be greatly amplified in 'zero before pole' topologies, producing poor noise characteristics. However floating point sign magnitude truncation does not produce a dc offset, producing better dc noise performance in 'zero before pole' topologies. For DF1 and DF2T topologies, 32 bit floating point arithmetic reduces noise components by only 9 dB than that produced by 24 bit fixed point arithmetic. This is significant since both of these fixed and floating point formats use 23 fractional bits. It has been shown that the DF1 topology, using 32 bit fixed point, produces superior noise characteristics to 32 bit floating point. Furthermore, the DF1 topology using floating point 40 bit extended precision produces inferior noise performance compared to that of 32 and 24 bit (double precision) fixed point arithmetic. Floating point extended precision produces comparably poor noise performance since every multiplication and a majority of the accumulations introduce quantisation.

Furthermore, extended precision using truncation is limited by floating point harmonic distortion. Extended precision rounding produces notably better THD+N figures.

Quantisation error transfer functions for 'pole before zero' topologies contain poles and zeros. Thus, filters with gain (boost settings) produce worse case noise characteristics. Consequently, under floating point implementation DF2 is a low noise topology. Unfortunately, harmonic distortion components in floating point truncation are noticeable and can limit the THD+N noise performance of such topologies. In these instances rounding is favourable to truncation. Due to the multiple quantisation points in DF1T and DF2T topologies single precision implementations result in poor noise performance compared to DF2 and DF1 respectively. However the DF1T topology, using 40 bit extended precision, produces a minimal pole path accumulation quantisation, since it sums the large magnitude 'pole paths' prior to summing with the unity bound input. This can be emulated in the DF2 implementation. If the pole paths are summed prior to summation with the input, the extended precision performance of DF2 is similar to DF1T.

Coupled form floating point implementations of low frequency tuned filters improve on DF1 noise performance by 20 dB. However for high frequency tuned filters the DF1 produces a 6 dB improvement over the coupled forms. The state-space topology produces excellent quantisation noise performance, using floating point rounding. Using truncation reduces its noise performance by 6 dB (due to harmonic distortion components). The state-space topology implemented in 40 bit extended precision compares closely in noise performance to the DF1 using 24 bit fixed point double precision. The Cabot structure did not produce similar noise performance to the state-space topology as reported by Cabot, (1992). However, in single precision floating point implementation the noise performance was found to be superior to DF1 and coupled forms. The Moorer ladder implementation produces less quantisation noise than the Massie ladder implementation by approximately,

9 dB. The lattice (Massie) produces more pronounced noise in the frequency boost region than the ladder (Massie).

The emulation of filters under finite wordlength constraints, as opposed to theoretical noise models, has given the opportunity to study the effects of various input stimuli on topologies implemented in fixed and floating point. At low input level fixed point filters were shown to produce low noise modulation. Floating point filters were shown to produce noise modulation, due to the quantisation noise being proportional to the signal level. DF2T in fixed point implementation was found to produce broad band quantisation noise at low level operation. Since its multiple quantisation points produce an uncorrelated overall noise characteristic. The application of triangular dither to the accumulator of DF1 in fixed point implementation was found to prevent any correlated distortions, at the cost of a slight increase in broad band noise. Although DF2T produces minimal correlated distortion for low level inputs, its broad band noise floor is still higher than the dithered DF1 (in fixed point implementation). In floating point implementation the dithered DF1 produced noise characteristics similar to the DF1 fixed point implementation.

Zero input limit cycles were not produced in floating point sign magnitude truncation implementations. Floating point rounding produced limit cycles of negligible magnitudes. Fixed point twos-complement truncation produces less instances of limit cycle behaviour than fixed point rounding. It was found that error feedback schemes can promote high frequency limit cycles. The number of rounding quantisers in the topology pole paths increases the possibility of limit cycle behaviour.

High frequency stimuli of a floating point 32 bit truncator is shown to produce aliased harmonic distortion components. These components are larger in magnitude than the flat noise floor produced by a fixed point 24 bit truncator. A filter using 32 bit floating point twos-complement truncation is found to produce even order intermodulation distortion

products that are above the 32 bit filter noise floor and the noise floor of a 24 bit fixed point filter.

It was found that higher sampling frequencies increase noise in the lattice and direct form topologies. This is caused by an increase in gain in the pole transfer function. However, coupled forms produce smaller increases in noise since their error transfer function are not as sensitive to sample frequency. It was found that topologies using L_2 norm scaling (state-space and ladder structures) have noise characteristics that are insensitive to increases in sampling frequency.

This investigation has produced a knowledge base of topology noise performance in floating and fixed point arithmetic and an understanding of the noise characteristics of topologies operating at different sampling frequencies. An outcome of this section of work was a conference paper, discussing optimal multiply accumulate wordlengths for various topologies (Clark et al, 1995), included in Appendix E. The work also provided the basis for the design of equalisation algorithms in two commercial digital audio mixing systems (Allen & Heath *DR series* and the *Icon* digital mixer).

7.1.3 Topology behaviour during coefficient update

The aim of work described in Chapter 6 was to investigate the behaviour of various topologies under coefficient update using various input stimuli, filter settings and sampling frequencies.

It is evident that magnitude changes at the DF1 topology output act as the disturbance excitation. This is because the output is the source to the pole paths of the topology. Critical disturbances occur in instances where high gain pole transfer functions exist in the target frequency response, typically low frequency high Q filters. This behaviour is fundamental in 'zero before pole' topologies.

'Pole before zero' topologies (DF2 and lattice) produce large disturbances. These disturbances are created by large gain changes in the pole transfer function. However in some cases the disturbance is attenuated by the zero transfer function attenuation characteristics. Large pole gain changes occur between low and high frequency tuned filters. Where low frequency excitation causes maximum pole disturbance and a high frequency tuned target frequency response has minimal attenuation in the zero transfer function.

DF1T is a 'pole before zero' topology, producing similar disturbance behaviour to DF2. DF2T is a 'zero before pole' topology, producing similar disturbance behaviour to DF1. Unstable coefficient sets occur under time varying conditions in transposed forms, resulting in large spike disturbances. A coefficient delay compensation scheme is developed to re-align the coefficients in the transposed topologies. The disturbance behaviour of DF1T and DF2T using coefficient delay compensation is identical to DF2 and DF1 respectively. Ladder implementations (Moorer and Massie) are found to produce similar disturbance responses. The ladder is 'pole before zero' and has disturbance characteristics to DF2, however the L_2 scaling greatly reduces its disturbance magnitudes. The lattice uses no scaling, resulting in similar disturbance magnitudes to DF2.

Using high frequency stimuli the coupled forms produce similar disturbances to DF1. However the coupled forms, Cabot and state-space topologies produce large disturbances using dc input excitation. The Kingsbury, Zölzer and state-space topologies produce the largest disturbances, thought to be attributable to the integrators in their pole transfer function implementations.

State change disturbance reduction through coefficient and parameter interpolation schemes is examined. Linear coefficient interpolation produces a continuously large step increment and is the most likely interpolator to cause disturbance. Despite large step-sizes initially in the interpolation period, the exponential interpolator produces small

disturbances. Using parameter interpolation operating at 48 kHz the DF1 produces negligible disturbances. The large state change disturbances in DF2 are greatly reduced by all interpolation schemes. Despite this, disturbances are still considerable – although small for parameter interpolation. Coefficient interpolation schemes can cause instabilities in the Zölzer and state-space topologies causing large disturbances. Parameter interpolation did not cause such instabilities.

Sub-sampled interpolators, operating at one tenth and one hundredth of the signal sampling frequency, 48 kHz, produced noticeably larger disturbances. The finite wordlength effects of the interpolation schemes were investigated. Final value 'clamping' errors at the end of the interpolation periods were found to produce negligible differences in the disturbance behaviour.

Disturbance behaviour of filters at higher signal sampling frequencies was examined. It is shown doubling the sample frequency increases the direct form pole transfer function gain by 12 dB. However, doubling the sample frequency, halves the magnitude changes between adjacent samples and thus halves the energy change in the pole paths. Using identical test parameters for a filter operating at a sampling frequency of 48 kHz and 96 kHz produces similar disturbance magnitudes. However, if the input excitation frequency is also doubled then the DF1 disturbances are considerably larger at a sampling frequency of 96 kHz. Coupled forms and the Cabot structure produce smaller disturbance magnitudes for an increase in sampling frequency. The lattice and DF2 structures produce increases in disturbance magnitude for an increase in sampling frequency. The ladder and state-space disturbance response appears to be insensitive to sampling frequency, producing the same disturbance magnitude at 96 kHz that was produced at a 48 kHz sampling frequency.

Previous work (Mourjopoulos et al, 1990; Hanna, 1994) has attempted to optimise interpolation rates – to achieve minimal audible distortion. However, previous work has

not provided an understanding of the disturbance mechanisms in each of the filter topologies or the effects of current and target frequency response and input stimuli. Furthermore, no study has been undertaken to examine the effects of higher signal sampling frequencies on disturbance. This work contributes to the knowledge base on topology behaviour under coefficient update and specifically provides an analytical understanding of topology disturbance behaviour and what filter responses produce worst case signal disturbance in each of the topologies.

The work also suggests that higher sampling frequencies may increase disturbance effects in some topologies for high frequency input stimuli. It is interesting to note that at a sampling frequency of 96 kHz the ear is not sensitive to frequencies above 24 kHz. However, signals above 24 kHz are capable of producing large disturbance magnitudes in the filter. This section of work has been instrumental in the development of coefficient change strategies in mixing systems currently in development.

7.2 Future Work

Work has been carried out to develop coefficient calculation schemes for minimal filter response distortion. These distortions are of the form of high frequency magnitude and phase response errors with respect to the continuous time ideal response. The audible perception of these distortions has not been studied in this work. However, an investigation into auditory perception of high frequency magnitude and phase errors specifically in equalisers would be of interest. This work would involve extensive listening tests of the magnitude and phase differences produced by discrete-time equaliser. This would ideally produce objective criteria, detailing what response distortions are audible. Such work would contribute to the debate of quality assessment of equalisers operating at higher sampling frequencies.

Distortion, noise and disturbance artefacts caused in the implementation of discrete-time equalisers is the main topic of this project. The work has not used any performance benchmarks to assess these noise, distortion and disturbance artefacts. Such benchmarks should consider these equaliser artefacts within the context of an entire audio mixing system. A mixing system contains many equalisers, operating in parallel, each processing separate audio signals, which may or may not be correlated sources. Ultimately these signals are processed and mixed together to form audio outputs – which are amplified and auditioned in many different audio playback scenarios. An investigation to quantify auditory perception of these artefacts within the context of an active mixing system would be useful. For example, overall system disturbance effects could be examined through the implementation of a multiple equaliser coefficient update scenario. The work could examine noise and distortion products through the implementation of multiple equalisers using various multiple input stimuli. The effects of sample frequency, arithmetic type and wordlength and coefficient update, on the entire system, would be studied. The overall noise, distortion and disturbance behaviour of an entire mixing system could therefore be assessed. This would be useful in specification of distortion and disturbance benchmarks for system design.

7.3 Conclusions

The primary aim of the project was to investigate distortions associated with the efficient implementation of discrete-time audio equalisers. This involved the investigation of distortion and computational efficiency in coefficient calculation techniques; behaviour of filter topologies under finite wordlength arithmetic and the behaviour of filter topologies under coefficient change. Higher signal sampling frequencies facilitate lower frequency response distortion and more efficient coefficient calculation techniques. The effects of signal sampling frequency on quantisation noise, distortion and state change disturbances

are topology dependent. Some topologies are sensitive to the gain in the pole transfer function, producing poor noise performance and high sensitivity to sampling frequency. Some topologies have noise and disturbance characteristics that are insensitive to sampling frequency.

Floating point arithmetic facilitates the use of many low noise topologies, eliminating overflow and low level limit cycles. However floating point arithmetic, particularly using truncation, can produce linear and non-linear distortion artefacts not produced in fixed point arithmetic. Rounding is a preferred quantiser in floating point systems. Ultimately floating point extended precision is not superior to fixed point double precision. Signal disturbance due to coefficient update is heavily dependent on frequency response and choice of input excitation. Furthermore higher sampling frequencies can potentially increase signal disturbance effects.

References

Agarwal R. C., Burrus C. S., New recursive digital filter structures having very low sensitivity and roundoff noise. IEEE transactions on circuits and systems, vol. CAS-22, no. 12. December 1975.

Analog Devices, ADSP-2106x SHARC user's manual, second edition, 1997.

ANSI/IEEE 754, Standard for floating point arithmetic, New York, 1985.

ANSI/IEEE 854, Standard for radix-independent floating point arithmetic, New York, 1987.

Audio Engineering Society, AES Standard 3 - Recommended practice for digital audio engineering – serial transmission format for two-channel linearly represented digital audio data. New York, 1992.

Audio Engineering Society, AES standard 3 - amendment 3, New York, 1999.

Audio Engineering Society, AES standard 17- standard method for digital audio engineering – measurement of digital audio equipment, New York, 1998.

Barnes C. W., Tran B. N., Leung S. H., On the statistics of fixed-point roundoff error. IEEE transactions on acoustics, speech and signal processing, vol. ASSP-33, no. 3, June 1985.

Bohn D. A., Constant-Q Graphic Equalizers. Journal of the Audio Engineering Society, vol. 34, no. 9, September 1986.

Bristow-Johnson R., The equivalence of various methods of computing biquad coefficients for audio parametric equalisers". Preprint 3906, Audio Engineering Society Convention, 1994.

Cabot R. C., Hybrid digital filter – United States Patent Number 5,089,981. February 1992.

Cavanagh J., Digital Computer Arithmetic. M^{c} Graw-Hill, New York, 1984.

Claasen T. A. C. M., Mecklenbräuker M. F. G., Peek J. B. H., Second-order digital filter with only one magnitude-truncation quantiser and having practically no limit-cycles. Electronic Letters, vol. 9, pp. 531-532, November, 1973.

Clark R. J., Ifeachor E. C., Rogers G. M., The study of arithmetic and wordlength requirements for digital audio filtering hardware. Preprint 4100, 99th Audio Engineering Society Convention , 1995.

Clark R. J., Ifeachor E. C., Rogers G. M., Real-time equaliser coefficient realisation with minimised computational load and distortion. Preprint 4360, 101st Audio Engineering Society Convention , 1996.

Clark R. J., Ifeachor E. C., Rogers G. M., Van Eetvelt P. W. J., Techniques for generating digital equalizer coefficients. Journal of the Audio Engineering Society, vol. 48, no. 4, April 2000.

Cirrus Logic, Crystal CS8427, Digital audio interface transceiver. Data Sheet DS477PP1, November 1999.

Dattorro J., The implementation of recursive digital filters for high-fidelity audio. Journal of the Audio Engineering Society, vol. 36, no. 11, November 1988.

Ding Y., Rossum D., Filter morphing of parametric equalizers and shelving filters for audio signal processing. Journal of the Audio Engineering Society, vol. 43, no. 10, October 1995.

Everest, F. A., The master handbook of acoustics, third edition, Chapter 3. M^{c} Graw-Hill, New York, 1994.

Gold B., Rader C. M., Effects of parameter quantization on the poles of a digital filter. Proceedings IEEE, vol. 55, pp. 688-689, May 1967.

Greenfield R. G., Application of digital techniques to loudspeaker equalization. Ph.D. thesis, Department of Electronic Systems Engineering, University of Essex, September 1991.

Hanna C., Real-time control of DSP parametric equalizers. Proceedings of the Audio Engineering Society 13th International Conference, page 277, December 1994.

Harris F. J., On the use of windows for harmonic analysis with the discrete Fourier transform. Proceedings of the IEEE, vol. 66, no. 1, January 1978.

Hawksford M. O., Digital signal processing tools for loudspeaker evaluation and discrete-time crossover design. Journal of the Audio Engineering Society, vol. 45, no.1/2, January/February 1997.

Higgins W. E., Munson Jr. D. C., Optimal and suboptimal error spectrum shaping for cascade form digital filters. IEEE transactions on Circuits and Systems, vol. CAS-31, p. 429, May 1984.

Hirata Y., Digitalization of conventional analogue filters for recording use. Preprint 1621, 66th Audio Engineering Society Convention , 1980.

Jackson L. B., Kaiser J. F., M^{c} Donald H. S., An approach to the implementation of digital filters, IEEE transactions on Audio Electroacoustics, vol. AU-16, page 413, September 1968.

Jackson L. B., Digital filters and signal processing. Chapter 11, Kluwer Academic Publishers. 1986

Kamerling S., Janse. K., Van der Meulen F., A new way of feedback suppression. Preprint 4735, 104th Audio Engineering Society Convention, 1998.

Kaneko T., Limit-cycle oscillations in floating-point digital filters. IEEE transactions on Audio Electroacoustics, vol. AU-21, pp. 100-106, April 1973.

Kingsbury N. G., Second-order recursive digital filter element for poles near the unit circle and the real z-axis. Electronic Letters, pp. 155-156, March 1972.

Kraght P.H., A linear phase digital equalizer with cubic-spline frequency response. Journal of the Audio Engineering Society, vol. 40, no. 5, May 1992.

Lian Y., Lim Y. C., Linear-phase digital audio tone control using multiplication-free FIR filter. Journal of the Audio Engineering Society, vol. 41, no. 10, October 1993.

Lui B., Kaneko T., Error analysis of digital filters realized with floating point arithmetic. Proceedings of the IEEE, vol. 57, no. 10, October 1969.

Linkwitz S. H., Active crossover networks for non-coincident drivers. Journal of the Audio Engineering Society, vol. 24, pp. 367-373, January/February 1976.

MathSoft, Mathcad 8 user's guide, ISBN 1-57682-039-4, 1998.

Massie D. C., An engineering study of the four-multiply normalized ladder filter. Journal of the Audio Engineering Society, vol. 41, no. 7/8 July/August, 1993.

M^{c} Nally G.W., A computer-based mixing and filtering system for digital sound signals. BBC Research Department Report, RD 1979/4, March 1979.

Metzler B., Audio measurement handbook, Audio Precision, 1993.

Moorer J. A., The manifold joys of conformal mapping: Applications to digital filtering in the studio. Journal of the Audio Engineering Society, vol. 31, no. 11, November 1983.

Moorer J. A., 48-bit integer processing beats 32-bit floating point for professional audio applications. Preprint 5038, 107th Audio Engineering Society Convention , 1999.

Motorola, DSP56300 Digital signal processor family manual, revision two, 1999.

Mourjopoulos J. N., Kyriakis-Bitzaros E. D., Goutis C. E., Theory and real-time implementation of time varying digital audio filters. Journal of the Audio Engineering Society, vol. 38, no. 7/8, July/August 1990.

Mullis C. T., Roberts R. A., Roundoff noise in digital filters: Frequency transformations and invariants. IEEE transactions on Acoustics, Speech and Signal Processing, vol. ASSP-24, no. 6, December 1976.

Orfanidis S.J., Digital parametric equaliser design with prescribed Nyquist-frequency gain. Preprint 4361, 101st Audio Engineering Society Convention , 1996.

Rabiner L. R., Gold B., Theory and applications of digital signal processing. Englewood Cliffs NJ: Prentice-Hall, 1975.

Regalia P. A., Mitra S. K., Vaidyanathan P. P., The digital all-pass filter: A versatile signal processing building block. Proceedings of the IEEE, vol. 76, no. 1, January 1988.

Rimell A., Hawksford M. O., Audibility analysis of processing noise in digital filter morphing schema. 101st Audio Engineering Society Convention, 1996.

Rosenthal M. F., Concept and design of a reconfigurable parallel processing system for digital audio. Ph.D. thesis Swiss Federal Institute of Technology, Zürich, 1997.

Samueli H., Willson Jr. A. N., Non-periodic forced overflow oscillations in digital filters, IEEE transactions Circuits Systems, vol. CAS-30, p.709, October 1983.

Shpak D. J., Analytical design of biquadratic filter sections for parametric filters. Journal of the Audio Engineering Society, vol 40, no.11, November 1992.

Temes G. C., Mitra S. K., Modern filter theory and design. John Wiley & Sons, 1973.

Texas Instruments, TMS320C3x Family user's guide, revision F, 1992.

Tromans R., Back to basics, sound principles in EQ design. Audio Media, page 102, April 1995.

Vanderkooy J., Lipshitz S. P., Dither in Digital Audio. Journal of the Audio Engineering Society, vol. 35, no. 12, December 1987, pages 966-975.

Välimäki V., Discrete-Time Modeling of Acoustic Tubes Using Fractional Delay Filters. Ph.D. thesis. Report no. 37, Faculty of Electrical Engineering, Helsinki University of Technology, 1995.

Volder J. E., The CORDIC trigonometric computing technique. IRE transactions EC-8, 1959.

Weinstein C., Oppenheim A. V., A comparison of roundoff noise in floating point and fixed point digital filter realizations. Proceedings of the IEEE, vol. 57, pp. 1181-1183, June 1969.

White S. A., Design of a digital biquadratic peaking or notch filter for digital audio equalisation. Preprint 2230, 78th Audio Engineering Society Convention , 1985.

Wise D. K., A survey of biquad filter structures for application to digital parametric equalization. Preprint 4820, 105th Audio Engineering Society Convention, 1998.

Wilson, R., Filter topologies. Journal of the Audio Engineering Society, vol. 41, no. 9, September 1993.

Zölzer U., Roundoff error analysis of digital filters. Preprint 3142, 91st Audio Engineering Society Convention, 1991.

Zölzer U., Redmer B., Bucholtz J., Strategies for switching digital audio filters. Preprint 3714, 95th Audio Engineering Society Convention, 1993.

Appendix A Arithmetic Functions and Test Data

This appendix contains Mathcad functions, used for finite wordlength arithmetic emulation. The functions emulate the fractional quantisation that occurs in fixed and floating point arithmetic operations. The basic quantisers (truncation and rounding) for fixed and floating point are listed. Quantisers for sign magnitude and twos-complement binary coded data are given. Floating point addition and multiplication functions are included.

A.1 Arithmetic Functions

A-1 sign magnitude truncation
A-2 sign magnitude rounding
A-3 twos-complement truncation
A-4 twos-complement rounding
A-5 floating point normaliser
A-6 floating point sign magnitude truncation
A-7 floating point sign magnitude rounding
A-8 floating point twos-complement truncation
A-9 floating point twos-complement rounding
A-10 floating point addition sign magnitude truncation
A-11 floating point addition sign magnitude rounding
A-12 floating point addition twos-complement truncation
A-13 floating point addition twos-complement rounding
A-14 floating point multiplication sign magnitude truncation
A-15 floating point multiplication sign magnitude rounding
A-16 floating point multiplication twos-complement truncation
A-17 floating point multiplication twos-complement rounding

Notes,
'n' denotes the number of fractional unsigned bits in the wordlength for all functions listed
'floatextprec' denotes the number of fractional unsigned bits in the floating point arithmetic unit (product and accumulator registers).

$$\text{smtrc}(x,n) := \begin{cases} \left(\text{floor}\left(x\cdot 2^{n}\right)\cdot 2^{-n}\right) & \text{if } x\geq 0 \\ \left(\text{ceil}\left(x\cdot 2^{n}\right)\cdot 2^{-n}\right) & \text{otherwise} \end{cases}$$

(A-1)

$$\text{smrnd}(x,n) := \begin{cases} \left[\text{floor}\left[\left[x+2^{(-(n+1))}\right]\cdot 2^{n}\right]\cdot 2^{-n}\right] & \text{if } x\geq 0 \\ \left[\text{ceil}\left[\left[x-2^{(-(n+1))}\right]\cdot 2^{n}\right]\cdot 2^{-n}\right] & \text{otherwise} \end{cases}$$

(A-2)

$$\text{twosCtrc}(x,n) := \text{floor}\left(x\cdot 2^{n}\right)\cdot 2^{-n}$$

(A-3)

$$\text{twosCrnd}(x,n) := \text{floor}\left[\left[x+2^{-(n+1)}\right]\cdot 2^{n}\right]\cdot 2^{-n}$$

(A-4)

```
flnorm(x) :=
    a ← |x|
    return 0 if a = 0
    i ← 0
    while a < 1
        a ← a·2
        i ← i − 1
        return i if a ≥ 1
    while a ≥ 2
        a ← a·0.5
        i ← i + 1
        return i if a < 2
    return i
```

(A-5)

```
flsmtrc(x, n) :=
    a ← |x|
    i ← 0
    return x if x = 0
    while a < 1
        a ← a·2
        i ← i − 1
    while a ≥ 2
        a ← a·0.5
        i ← i + 1
    out ← smtrc(x, n − i)
    return out
```

(A-6)

```
flsmrnd(x, n) := | a ← |x|
                 | i ← 0
                 | return x if x = 0
                 | while a < 1
                 |     | a ← a·2
                 |     | i ← i − 1
                 | while a ≥ 2
                 |     | a ← a·0.5
                 |     | i ← i + 1
                 | out ← smrnd(x, n − i)
                 | return out
```

(A-7)

```
fl2ctrc(x, n) := | a ← |x|
                 | i ← 0
                 | return x if x = 0
                 | while a < 1
                 |     | a ← a·2
                 |     | i ← i − 1
                 | while a ≥ 2
                 |     | a ← a·0.5
                 |     | i ← i + 1
                 | return twosCtrc(x, n − i)
```

(A-8)

```
fl2crnd(x, n) := | a ← |x|
                 | i ← 0
                 | return x if x = 0
                 | while a < 1
                 |     | a ← a·2
                 |     | i ← i − 1
                 | while a ≥ 2
                 |     | a ← a·0.5
                 |     | i ← i + 1
                 | return twosCrnd(x, n − i)
```

(A-9)

$$
\begin{aligned}
&\text{Addsmtrc}(a,b) := \\
&\quad aexp \leftarrow \text{flnorm}(a) \\
&\quad bexp \leftarrow \text{flnorm}(b) \\
&\quad cexp \leftarrow aexp \quad \text{if } aexp > bexp \\
&\quad cexp \leftarrow bexp \quad \text{otherwise} \\
&\quad amant \leftarrow \text{flsmtrc}(a, \text{floatextprec}) \cdot 2^{-aexp} \\
&\quad bmant \leftarrow \text{flsmtrc}(b, \text{floatextprec}) \cdot 2^{-bexp} \\
&\quad result \leftarrow \left[bmant + amant \cdot 2^{-(bexp - aexp)} \right] \quad \text{if } aexp < bexp \\
&\quad result \leftarrow \left[amant + bmant \cdot 2^{-(aexp - bexp)} \right] \quad \text{otherwise} \\
&\quad out \leftarrow \text{flsmtrc}(result, \text{floatextprec}) \\
&\quad \text{return } out \cdot 2^{cexp}
\end{aligned}
$$

(A-10)

$$
\begin{aligned}
&\text{Addsmrnd}(a,b) := \\
&\quad aexp \leftarrow \text{flnorm}(a) \\
&\quad bexp \leftarrow \text{flnorm}(b) \\
&\quad cexp \leftarrow aexp \quad \text{if } aexp > bexp \\
&\quad cexp \leftarrow bexp \quad \text{otherwise} \\
&\quad amant \leftarrow \text{flsmrnd}\left(a \cdot 2^{-aexp}, \text{floatextprec}\right) \\
&\quad bmant \leftarrow \text{flsmrnd}\left(b \cdot 2^{-bexp}, \text{floatextprec}\right) \\
&\quad result \leftarrow \left[bmant + amant \cdot 2^{-(bexp - aexp)} \right] \quad \text{if } aexp < bexp \\
&\quad result \leftarrow \left[amant + bmant \cdot 2^{-(aexp - bexp)} \right] \quad \text{otherwise} \\
&\quad out \leftarrow \text{flsmrnd}(result, \text{floatextprec}) \\
&\quad \text{return } out \cdot 2^{cexp}
\end{aligned}
$$

(A-11)

$$
\begin{aligned}
&\text{Add2ctrc}(a,b) := \\
&\quad aexp \leftarrow \text{flnorm}(a) \\
&\quad bexp \leftarrow \text{flnorm}(b) \\
&\quad cexp \leftarrow aexp \quad \text{if } aexp > bexp \\
&\quad cexp \leftarrow bexp \quad \text{otherwise} \\
&\quad amant \leftarrow \text{fl2ctrc}(a, \text{floatextprec}) \cdot 2^{-aexp} \\
&\quad bmant \leftarrow \text{fl2ctrc}(b, \text{floatextprec}) \cdot 2^{-bexp} \\
&\quad result \leftarrow \left[bmant + amant \cdot 2^{-(bexp - aexp)} \right] \quad \text{if } aexp < bexp \\
&\quad result \leftarrow \left[amant + bmant \cdot 2^{-(aexp - bexp)} \right] \quad \text{otherwise} \\
&\quad out \leftarrow \text{fl2ctrc}(result, \text{floatextprec}) \\
&\quad \text{return } out \cdot 2^{cexp}
\end{aligned}
$$

(A-12)

$$\begin{aligned}
&\text{Add2crnd}(a,b) := \\
&\quad aexp \leftarrow \text{flnorm}(a) \\
&\quad bexp \leftarrow \text{flnorm}(b) \\
&\quad cexp \leftarrow aexp \quad \text{if } aexp > bexp \\
&\quad cexp \leftarrow bexp \quad \text{otherwise} \\
&\quad amant \leftarrow \text{fl2crnd}(a, \text{floatextprec}) \cdot 2^{-aexp} \\
&\quad bmant \leftarrow \text{fl2crnd}(b, \text{floatextprec}) \cdot 2^{-bexp} \\
&\quad result \leftarrow \left[bmant + amant \cdot 2^{-(bexp - aexp)} \right] \quad \text{if } aexp < bexp \\
&\quad result \leftarrow \left[amant + bmant \cdot 2^{-(aexp - bexp)} \right] \quad \text{otherwise} \\
&\quad out \leftarrow \text{fl2crnd}(result, \text{floatextprec}) \\
&\quad \text{return } out \cdot 2^{cexp}
\end{aligned}$$

(A-13)

$$\begin{aligned}
&\text{Multsmtrc}(a,b) := \\
&\quad aexp \leftarrow \text{flnorm}(a) \\
&\quad bexp \leftarrow \text{flnorm}(b) \\
&\quad cexp \leftarrow aexp + bexp \\
&\quad amant \leftarrow \text{flsmtrc}\left(a \cdot 2^{-aexp}, \text{floatextprec}\right) \\
&\quad bmant \leftarrow \text{flsmtrc}\left(b \cdot 2^{-bexp}, \text{floatextprec}\right) \\
&\quad cmant \leftarrow amant \cdot bmant \\
&\quad \text{return } \text{flsmtrc}(cmant, \text{floatextprec}) \cdot 2^{cexp}
\end{aligned}$$

(A-14)

$$\begin{aligned}
&\text{Multsmrnd}(a,b) := \\
&\quad aexp \leftarrow \text{flnorm}(a) \\
&\quad bexp \leftarrow \text{flnorm}(b) \\
&\quad cexp \leftarrow aexp + bexp \\
&\quad amant \leftarrow \text{flsmrnd}\left(a \cdot 2^{-aexp}, \text{floatextprec}\right) \\
&\quad bmant \leftarrow \text{flsmrnd}\left(b \cdot 2^{-bexp}, \text{floatextprec}\right) \\
&\quad cmant \leftarrow amant \cdot bmant \\
&\quad \text{return } \text{flsmrnd}(cmant, \text{floatextprec}) \cdot 2^{cexp}
\end{aligned}$$

(A-15)

$$\begin{aligned}
&\text{Mult2ctrc}(a,b) := \\
&\quad aexp \leftarrow \text{flnorm}(a) \\
&\quad bexp \leftarrow \text{flnorm}(b) \\
&\quad cexp \leftarrow aexp + bexp \\
&\quad amant \leftarrow \text{fl2ctrc}\left(a \cdot 2^{-aexp}, \text{floatextprec}\right) \\
&\quad bmant \leftarrow \text{fl2ctrc}\left(b \cdot 2^{-bexp}, \text{floatextprec}\right) \\
&\quad cmant \leftarrow amant \cdot bmant \\
&\quad \text{return } \text{fl2ctrc}(cmant, \text{floatextprec}) \cdot 2^{cexp}
\end{aligned}$$

(A-16)

$$\begin{aligned}
\text{Mult2crnd}(a,b) := \;& aexp \leftarrow \text{flnorm}(a) \\
& bexp \leftarrow \text{flnorm}(b) \\
& cexp \leftarrow aexp + bexp \\
& amant \leftarrow \text{fl2crnd}\left(a \cdot 2^{-aexp}, \text{floatextprec}\right) \\
& bmant \leftarrow \text{fl2crnd}\left(b \cdot 2^{-bexp}, \text{floatextprec}\right) \\
& cmant \leftarrow amant \cdot bmant \\
& \text{return } \text{fl2crnd}(cmant, \text{floatextprec}) \cdot 2^{cexp}
\end{aligned} \tag{A-17}$$

A.2 Comparison of DSP platform quantised data with Mathcad emulation

The following assembly code generates 32 bit floating point truncation or rounding (23 fractional bits) - .ROUND_ZERO or .ROUND_NEAREST select truncation or rounding. The code was executed on the ADSP-2106x DSP platform simulator. The quantised data can be inspected in the disassembly window of the Visual DSP simulator, overleaf.

```
/* Quant23.ASM       ADSP-2106x */
.PRECISION=32;                                  /* 23 fractional bits */
.ROUND_ZERO;            /* .ROUND_NEAREST;   generates rounded data at 23 rd fractional bit */
start:          F1 =  0.0;
                F2 =  2.123456789E-36;
                F2 = -2.123456789E-36;
                F3 =  1.192092895507813E-7;
                F3 = -1.192092895507813E-7;
                F4 =  5.960464477539063E-8;
                F4 = -5.960464477539063E-8;
                F5 =  0.12345432123456;
                F5 = -0.12345432123456;
                F6 =  0.49999988079071;
                F6 = -0.49999988079071;
                F7 =  0.499999940395355;
                F7 = -0.499999940395355;
                F8 =  0.99999988079071;
                F8 = -0.99999988079071;
                F1 =  0.999999940395355;
                F1 = -0.999999940395355;
                F2 =  1.00000011920929;
                F2 = -1.00000011920929;
                F3 =  1.00000005960464;
                F3 = -1.00000005960464;
                F4 =  1.23400011920929;
                F4 = -1.23400011920929;
                F5 =  1.23400005960464;
                F5 = -1.23400005960464;
                F6 =  1.99999988079071;
                F6 = -1.99999988079071;
                F7 = 2.00000005960464;
                F7 =-2.00000005960464;
                F8 = 8.50000011920929;
                F8 =-8.50000011920929;
                F1 = 16384.0000001788;
                F1 =-16384.0000001788;
                F2 = 31.6227766E+6;
                F2 =-31.6227766E+6;
                F3 = 3.16227766000001E+7;
                F3 = -3.16227766000001E+7;
end:        IDLE;
```

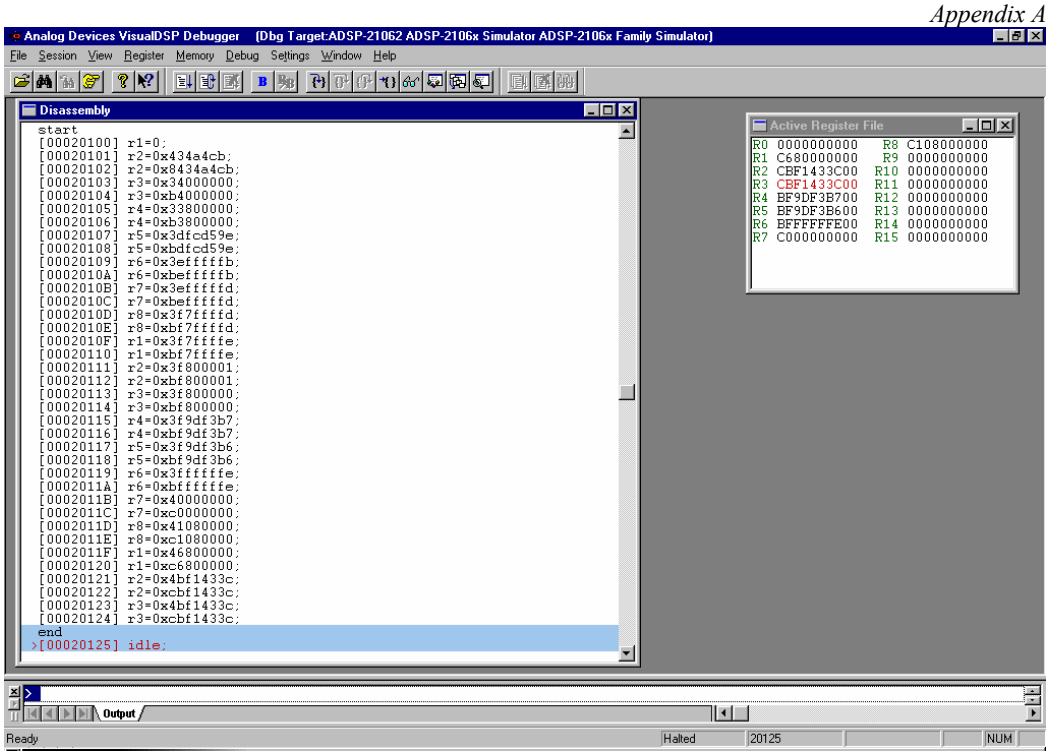

Figure A-1 Quantised data (32 bit floating point truncation) generated in the Visual DSP simulator.

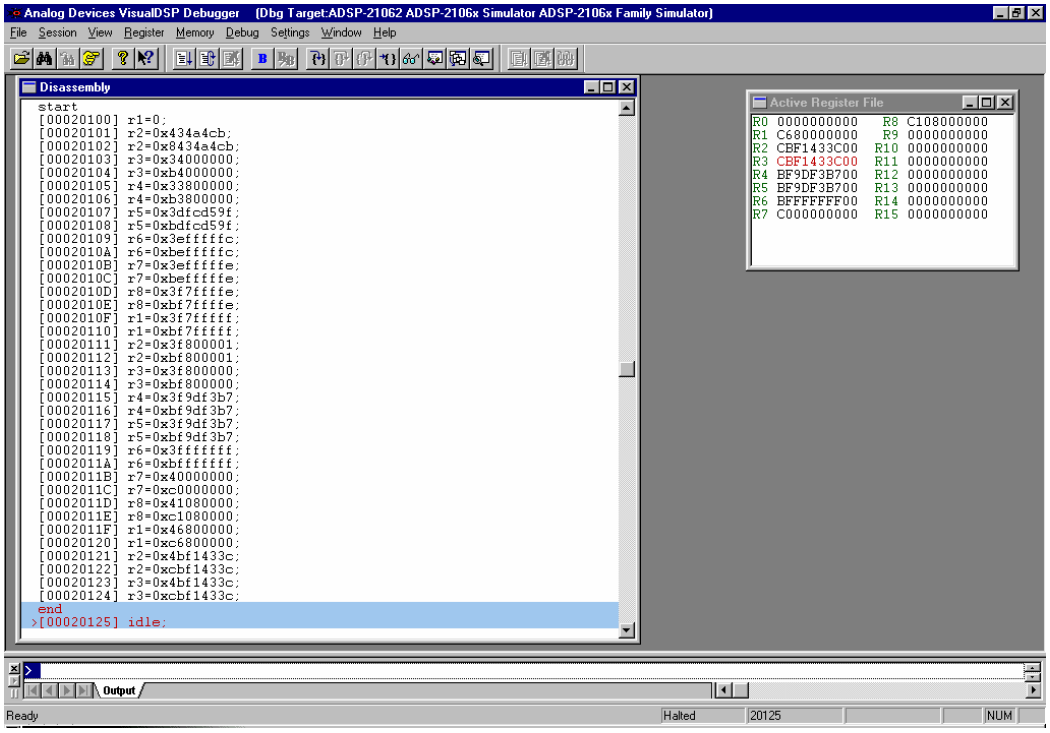

Figure A-2 Quantised data (32 bit floating point rounding) generated in the Visual DSP simulator.

Quantised data using the Mathcad emulated quantisation functions for 32 bit floating point (23 fractional bits) using sign magnitude truncation and rounding are shown overleaf. The quantised emulated data is identical (bit exact) to the quantised data in the DSP platform simulator.

$v := 0.0$
Qdata := flsmtrc(0.0, 23) IEEEExpo(Qdata) = 3fh IEEEFract(Qdata) = 0h

$v := 2.123456789 \cdot 10^{-36}$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 4h IEEEFract(Qdata) = 34a4cbh
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 84h IEEEFract(Qdata) = 34a4cbh

$v := 1.192092895507813 \cdot 10^{-7}$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 34h IEEEFract(Qdata) = 0h
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0b4h IEEEFract(Qdata) = 0h

$v := 5.960464477539063 \cdot 10^{-8}$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 33h IEEEFract(Qdata) = 800000h
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0b3h IEEEFract(Qdata) = 800000h

$v := 0.12345432123456$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 3dh IEEEFract(Qdata) = 0fcd59eh
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0bdh IEEEFract(Qdata) = 0fcd59eh

$v := 0.49999988079071$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 3eh IEEEFract(Qdata) = 0fffffbh
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0beh IEEEFract(Qdata) = 0fffffbh

$v := 0.499999940395355$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 3eh IEEEFract(Qdata) = 0fffffdh
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0beh IEEEFract(Qdata) = 0fffffdh

$v := 0.99999988079071$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 3fh IEEEFract(Qdata) = 7ffffdh
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0bfh IEEEFract(Qdata) = 7ffffdh

$v := 0.999999940395355$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 3fh IEEEFract(Qdata) = 7ffffeh
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0bfh IEEEFract(Qdata) = 7ffffeh

$v := 1.00000011920929$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 3fh IEEEFract(Qdata) = 800001h
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0bfh IEEEFract(Qdata) = 800001h

$v := 1.00000005960464$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 3fh IEEEFract(Qdata) = 800000h
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0bfh IEEEFract(Qdata) = 800000h

$v := 1.23400011920929$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 3fh IEEEFract(Qdata) = 9df3b7h
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0bfh IEEEFract(Qdata) = 9df3b7h

$v := 1.23400005960464$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 3fh IEEEFract(Qdata) = 9df3b6h
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0bfh IEEEFract(Qdata) = 9df3b6h

$v := 1.99999988079071$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 3fh IEEEFract(Qdata) = 0fffffeh
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0bfh IEEEFract(Qdata) = 0fffffeh

$v := 2.00000005960464$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 40h IEEEFract(Qdata) = 0h
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0c0h IEEEFract(Qdata) = 0h

$v := 8.50000011920929$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 41h IEEEFract(Qdata) = 80000h
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0c1h IEEEFract(Qdata) = 80000h

$v := 16384.0000001788$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 46h IEEEFract(Qdata) = 800000h
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0c6h IEEEFract(Qdata) = 800000h

$v := 31.6227766 \cdot 10^{6}$
Qdata := flsmtrc(v, 23) IEEEExpo(Qdata) = 4bh IEEEFract(Qdata) = 0f1433ch
Qdata := flsmtrc(- v, 23) IEEEExpo(Qdata) = 0cbh IEEEFract(Qdata) = 0f1433ch

$v := 0.0$

$Qdata := flsmrnd(0.0, 23)$	$IEEEExpo(Qdata) = 3fh$	$IEEEFract(Qdata) = 0h$

$v := 2.123456789 \cdot 10^{-36}$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 4h$	$IEEEFract(Qdata) = 34a4cbh$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 84h$	$IEEEFract(Qdata) = 34a4cbh$

$v := 1.19209289550781310^{-7}$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 34h$	$IEEEFract(Qdata) = 0h$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0b4h$	$IEEEFract(Qdata) = 0h$

$v := 5.96046447753906310^{-8}$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 33h$	$IEEEFract(Qdata) = 800000h$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0b3h$	$IEEEFract(Qdata) = 800000h$

$v := 0.12345432123456$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 3dh$	$IEEEFract(Qdata) = 0fcd59fh$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0bdh$	$IEEEFract(Qdata) = 0fcd59fh$

$v := 0.49999988079071$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 3eh$	$IEEEFract(Qdata) = 0fffffch$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0beh$	$IEEEFract(Qdata) = 0fffffch$

$v := 0.499999940395355$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 3eh$	$IEEEFract(Qdata) = 0fffffeh$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0beh$	$IEEEFract(Qdata) = 0fffffeh$

$v := 0.99999988079071$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 3fh$	$IEEEFract(Qdata) = 7ffffeh$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0bfh$	$IEEEFract(Qdata) = 7ffffeh$

$v := 0.999999940395355$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 3fh$	$IEEEFract(Qdata) = 7ffffh$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0bfh$	$IEEEFract(Qdata) = 7ffffh$

$v := 1.00000011920929$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 3fh$	$IEEEFract(Qdata) = 800001h$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0bfh$	$IEEEFract(Qdata) = 800001h$

$v := 1.00000005960464$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 3fh$	$IEEEFract(Qdata) = 800000h$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0bfh$	$IEEEFract(Qdata) = 800000h$

$v := 1.23400011920929$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 3fh$	$IEEEFract(Qdata) = 9df3b7h$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0bfh$	$IEEEFract(Qdata) = 9df3b7h$

$v := 1.23400005960464$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 3fh$	$IEEEFract(Qdata) = 9df3b7h$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0bfh$	$IEEEFract(Qdata) = 9df3b7h$

$v := 1.99999988079071$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 3fh$	$IEEEFract(Qdata) = 0ffffffh$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0bfh$	$IEEEFract(Qdata) = 0ffffffh$

$v := 2.00000005960464$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 40h$	$IEEEFract(Qdata) = 0h$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0c0h$	$IEEEFract(Qdata) = 0h$

$v := 8.50000011920929$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 41h$	$IEEEFract(Qdata) = 80000h$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0c1h$	$IEEEFract(Qdata) = 80000h$

$v := 16384.0000001788$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 46h$	$IEEEFract(Qdata) = 800000h$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0c6h$	$IEEEFract(Qdata) = 800000h$

$v := 31.6227766 \cdot 10^{6}$

$Qdata := flsmrnd(v, 23)$	$IEEEExpo(Qdata) = 4bh$	$IEEEFract(Qdata) = 0f1433ch$
$Qdata := flsmrnd(-v, 23)$	$IEEEExpo(Qdata) = 0cbh$	$IEEEFract(Qdata) = 0f1433ch$

A.3 Comparison of DSP platform floating point addition with the Mathcad emulation

The following assembly code performs various additions using 32 bit (23 fractional bits) floating point arithmetic, truncated or rounded data - .ROUND_ZERO or .ROUND_NEAREST selects the truncation or rounding for the constant data. Arithmetic truncation or rounding is selected in the MODE1 register. The code was executed on the ADSP-2106x DSP platform simulator.

```
/*    AddTest.ASM        ADSP-2106x          */

#include "def21060.h"                /*  Memory Mapped IOP register definitions  */
.PRECISION=32;
.ROUND_ZERO;                                        /* truncate constant data */
/* .ROUND_NEAREST;                                   round constant data */
start:
/*          Mode 1 Register
0001 0000          32 bit round
0001 8000          32 bit truncate
0000 0000          40 bit round
0000 8000          40 bit truncate
*/
BIT SET MODE1 0x00018000;                /* truncate, 32 bit arithmetic */
/* BIT SET MODE1 0x00010000;              round, 32 bit arithmetic */

          F8  =  31.6227766E+6;
          F9  =  0.0;
          F0  =  F8 + F9;

          F8  =  31.6227766E+6;
          F9  =  2.123456789E-36;
          F1  =  F8 + F9;

          F8  =  1.192092895507813E-7;
          F9  =  5.960464477539063E-8;
          F2  =  F8 + F9;

          F8  =  1.192092895507813E-7;
          F9  =  5.960464477539063E-8;
          F3  =  F8 - F9;

          F8  =  0.49999988079071;
          F9  =  0.499999940395355;
          F4  =  F8 - F9;

          F8  =  0.49999988079071;
          F9  =  1.99999988079071;
          F5  =  F8 + F9;

          F8  =  1.23400005960464;
          F9  =  16384.0000001788;
          F6  =  F8 + F9;

          F8  =  8.50000011920929;
          F9  =  31.6227766E+6;
          F7  =  F8 - F9;

end:        IDLE;
```

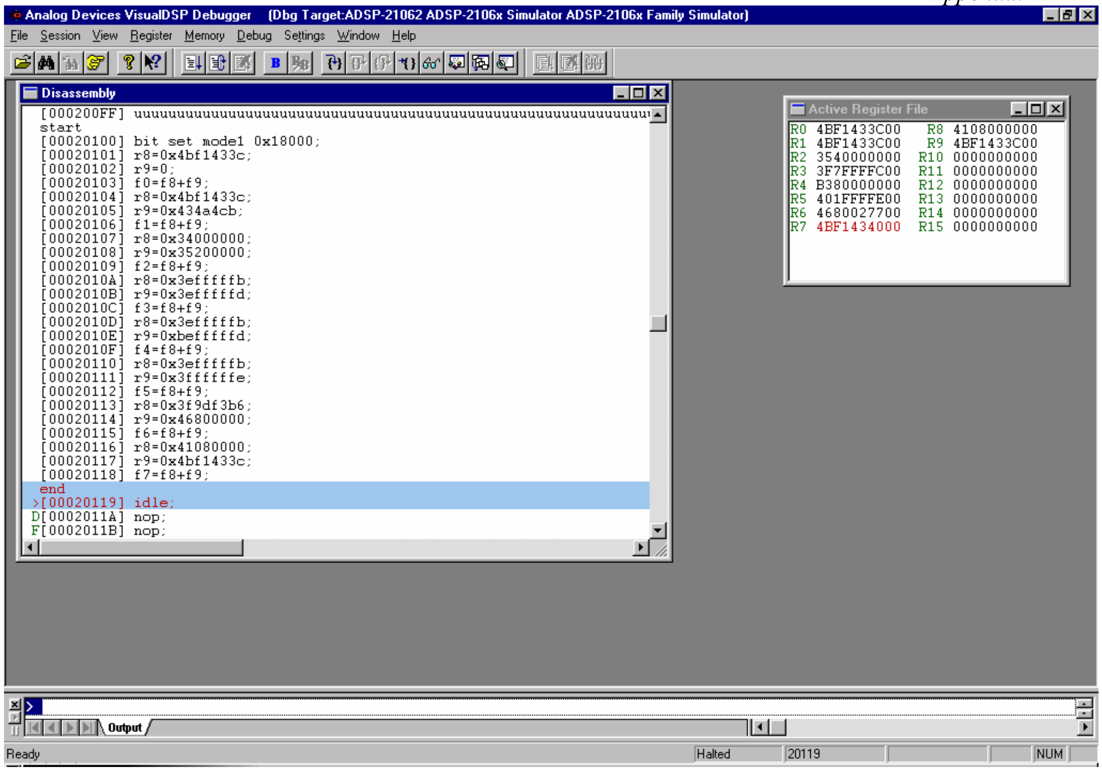

Figure A-3 Floating point addition test data using 32 bit truncation, generated in the Visual DSP simulator.

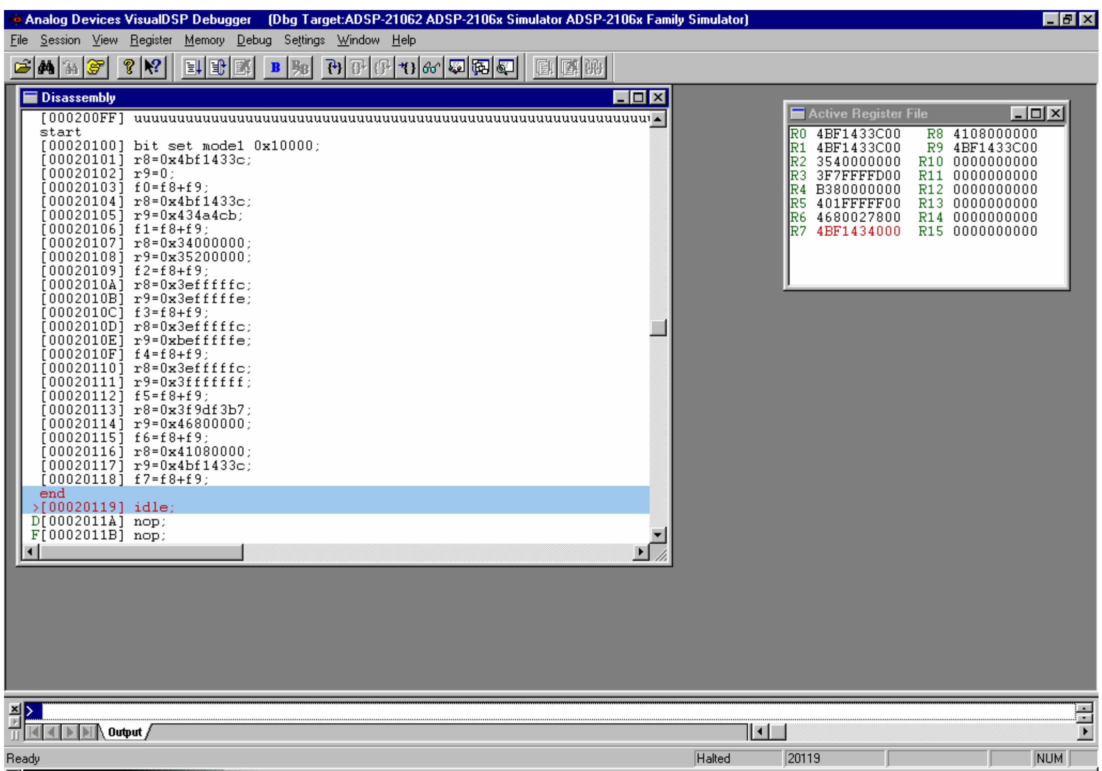

Figure A-4 Floating point addition test data using 32 bit rounding, generated in the Visual DSP simulator.

$a := 31.62277661 0^{6}$ $b := 0.0$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Addsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 4bh$ $IEEEFract(Qdata) = 0f1433ch$

$a := 31.62277661 0^{6}$ $b := 2.1234567891 0^{-36}$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Addsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 4bh$ $IEEEFract(Qdata) = 0f1433ch$

$a := 1.1920928955078131 0^{-7}$ $b := 5.960464477539063 10^{-7}$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Addsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 35h$ $IEEEFract(Qdata) = 400000h$

$a := 0.49999988079071$ $b := 0.499999940395355$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Addsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 3fh$ $IEEEFract(Qdata) = 7ffffch$

$a := 0.49999988079071$ $b := -0.499999940395355$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Addsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 0b3h$ $IEEEFract(Qdata) = 800000h$

$a := 0.49999988079071$ $b := 1.99999988079071$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Addsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 40h$ $IEEEFract(Qdata) = 1ffffeh$

$a := 1.23400005960464$ $b := 16384.0000001788$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Addsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 46h$ $IEEEFract(Qdata) = 800277h$

$a := 8.50000011920929$ $b := 31.62277661 0^{6}$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Addsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 4bh$ $IEEEFract(Qdata) = 0f14340h$

$a := 31.622776610^{6}$ $b := 0.0$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Addsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 4bh$ $IEEEFract(Qdata) = 0f1433ch$

$a := 31.622776610^{6}$ $b := 2.12345678910^{-36}$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Addsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 4bh$ $IEEEFract(Qdata) = 0f1433ch$

$a := 1.192092895507813 10^{-7}$ $b := 5.96046447753906310^{-7}$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Addsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 35h$ $IEEEFract(Qdata) = 400000h$

$a := 0.49999988079071$ $b := 0.499999940395355$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Addsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 3fh$ $IEEEFract(Qdata) = 7ffffdh$

$a := 0.49999988079071$ $b := -0.499999940395355$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Addsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 0b3h$ $IEEEFract(Qdata) = 800000h$

$a := 0.49999988079071$ $b := 1.99999988079071$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Addsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 40h$ $IEEEFract(Qdata) = 1fffffh$

$a := 1.23400005960464$ $b := 16384.0000001788$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Addsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 46h$ $IEEEFract(Qdata) = 800278h$

$a := 8.50000011920929$ $b := 31.622776610^{6}$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Addsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 4bh$ $IEEEFract(Qdata) = 0f14340h$

A.4 Comparison of DSP platform floating point multiplication with the Mathcad emulation

The following assembly code performs various multiplications using 32 bit (23 fractional bits) floating point arithmetic, truncated or rounded data - .ROUND_ZERO or .ROUND_NEAREST selects the truncation or rounding for the constant data. Arithmetic truncation or rounding is selected in the MODE1 register. The code was executed on the ADSP-2106x DSP platform simulator.

```
/*          MultTest.ASM        ADSP-2106x                                          */
#include "def21060.h"               /*  Memory Mapped IOP register definitions      */
.PRECISION=32;
.ROUND_ZERO;                                    /* truncate constant data */
/* .ROUND_NEAREST;                              round constant data */

start:
/*          Mode 1 Register
0001 0000           32 bit round
0001 8000           32 bit truncate
0000 0000           40 bit round
0000 8000           40 bit truncate
*/
BIT SET MODE1 0x00018000;               /* truncate, 32 bit arithmetic */
/* BIT SET MODE1 0x00010000;            round, 32 bit arithmetic */

        F8  =  31.6227766E+6;
        F9  =  0.0;
        F0  =  F8 * F9;

        F8  =  31.6227766E+6;
        F9  =  2.123456789E-36;
        F1  =  F8 * F9;

        F8  =  1.192092895507813E-7;
        F9  =  5.960464477539063E-7;
        F2  =  F8 * F9;

        F8  =  0.49999988079071;
        F9  =  0.499999940395355;
        F3  =  F8 * F9;

        F8  =  0.49999988079071;
        F9  =  -0.499999940395355;
        F4  =  F8 * F9;

        F8  =  0.49999988079071;
        F9  =  1.99999988079071;
        F5  =  F8 * F9;

        F8  =  1.23400005960464;
        F9  =  16384.0000001788;
        F6  =  F8 * F9;

        F8  =  8.50000011920929;
        F9  =  31.6227766E+6;
        F7  =  F8 * F9;

end:        IDLE;
```

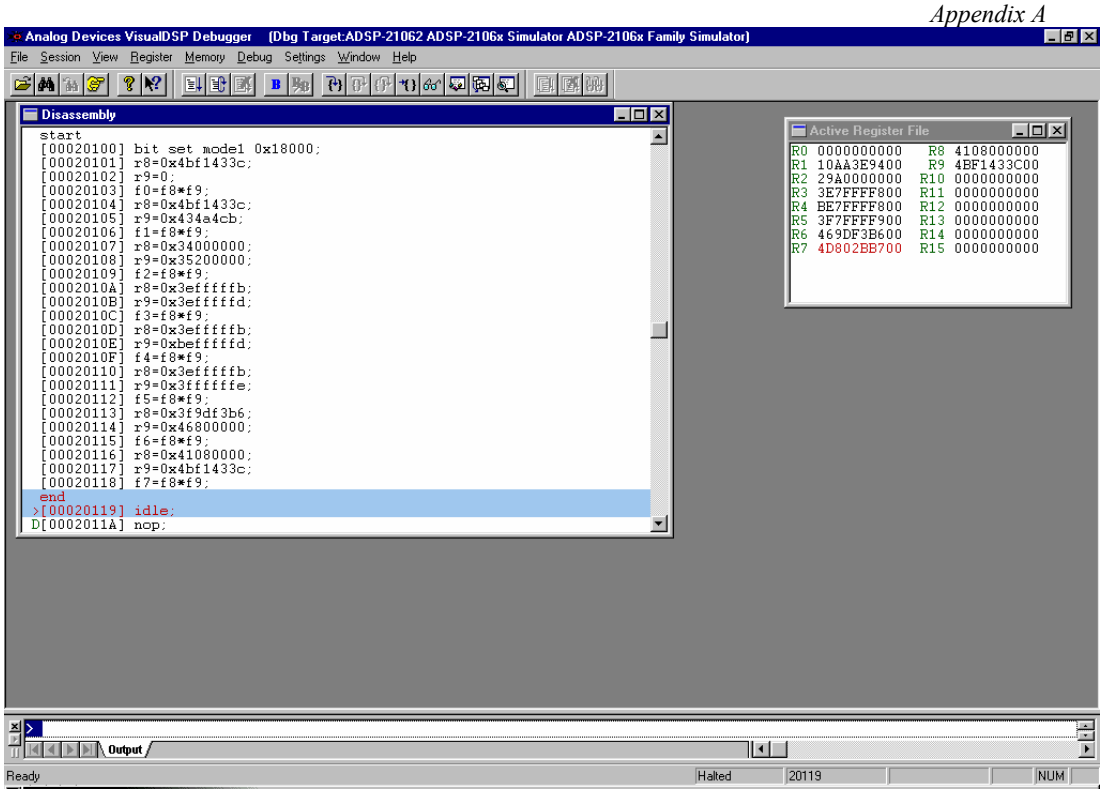

Figure A-5 Floating point multiplication test data using 32 bit truncation, generated in the Visual DSP simulator.

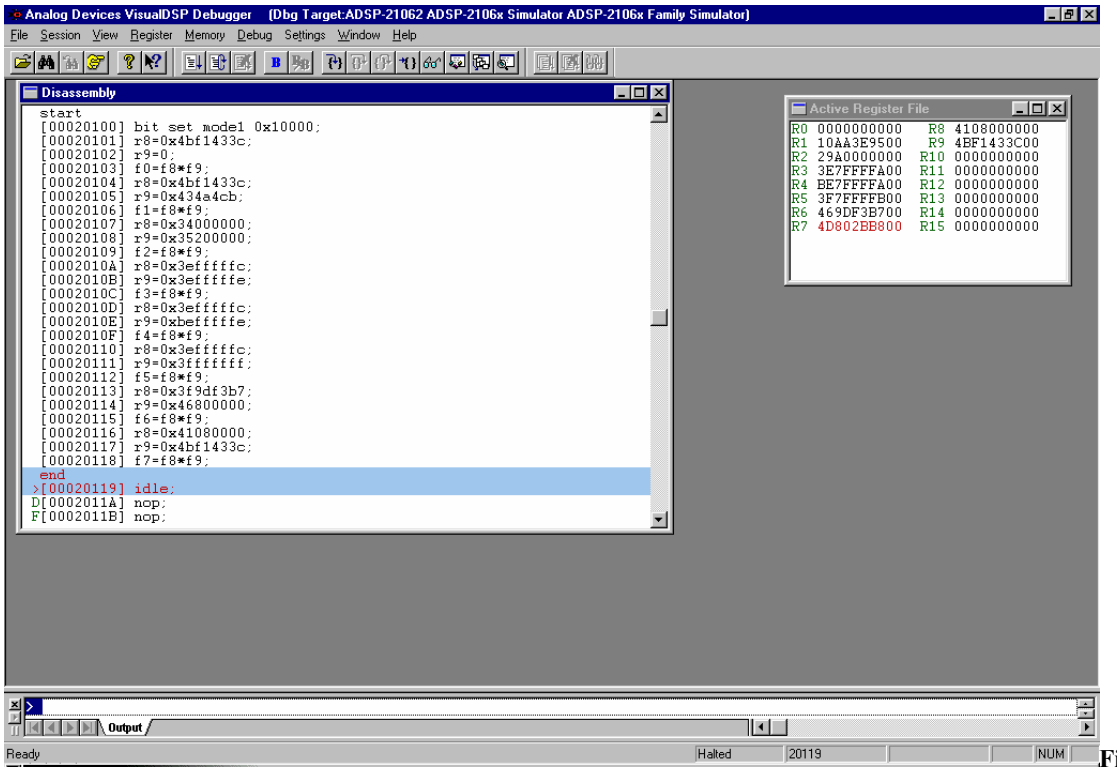

Figure A-6 Floating point multiplication test data using 32 bit rounding, generated in the Visual DSP simulator.

$a := 31.622776610^{6}$ $b := 0.0$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Multsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 3fh$ $IEEEFract(Qdata) = 0h$

$a := 31.622776610^{6}$ $b := 2.12345678910^{-36}$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Multsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 10h$ $IEEEFract(Qdata) = 0aa3e94h$

$a := 1.19209289550781310^{-7}$ $b := 5.96046447753906310^{-7}$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Multsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 29h$ $IEEEFract(Qdata) = 0a00000h$

$a := 0.49999988079071$ $b := 0.499999940395355$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Multsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 3eh$ $IEEEFract(Qdata) = 7ffff8h$

$a := 0.49999988079071$ $b := -0.499999940395355$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Multsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 0beh$ $IEEEFract(Qdata) = 7ffff8h$

$a := 0.49999988079071$ $b := 1.99999988079071$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Multsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 3fh$ $IEEEFract(Qdata) = 7ffff9h$

$a := 1.23400005960464$ $b := 16384.0000001788$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Multsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 46h$ $IEEEFract(Qdata) = 9df3b6h$

$a := 8.50000011920929$ $b := 31.622776610^{6}$

$Qa := flsmtrc(a, 23)$ $Qb := flsmtrc(b, 23)$

$Qdata := Multsmtrc(Qa, Qb)$ $IEEEExpo(Qdata) = 4dh$ $IEEEFract(Qdata) = 802bb7h$

$a := 31.6227766 10^{6}$ $b := 0.0$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Multsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 3fh$ $IEEEFract(Qdata) = 0h$

$a := 31.6227766 10^{6}$ $b := 2.12345678 10^{-36}$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Multsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 10h$ $IEEEFract(Qdata) = 0aa3e95h$

$a := 1.19209289550781 10^{-7}$ $b := 5.96046447753906 10^{-7}$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Multsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 29h$ $IEEEFract(Qdata) = 0a00000h$

$a := 0.49999988079071$ $b := 0.499999940395355$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Multsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 3eh$ $IEEEFract(Qdata) = 7ffffah$

$a := 0.49999988079071$ $b := -0.499999940395355$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Multsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 0beh$ $IEEEFract(Qdata) = 7ffffah$

$a := 0.49999988079071$ $b := 1.99999988079071$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Multsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 3fh$ $IEEEFract(Qdata) = 7ffffbh$

$a := 1.23400005960464$ $b := 16384.0000001788$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Multsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 46h$ $IEEEFract(Qdata) = 9df3b7h$

$a := 8.50000011920929$ $b := 31.6227766 10^{6}$

$Qa := flsmrnd(a, 23)$ $Qb := flsmrnd(b, 23)$

$Qdata := Multsmrnd(Qa, Qb)$ $IEEEExpo(Qdata) = 4dh$ $IEEEFract(Qdata) = 802bb8h$

Appendix B Finite wordlength topology functions

Appendix B lists the Mathcad functions implementing filter topologies using various forms of emulated finite wordlength arithmetic. The various topologies are listed below. The arithmetic type and precision used in each implementation is also listed.

B-1 DF1, fixed point single precision, example uses a coefficient scaling of 2
B-2 DF1, first order error feedback (inserting a zero at dc in the error transfer function)
B-3 Floating point quantisation of input stimulus to n fractional bits
B-4 DF1, floating point single precision
B-5 DF2, floating point single precision
B-6 DF2T, floating point single precision
B-7 DF2T, fixed point single precision, example uses a coefficient scaling of 2
B-8 DF1T, floating point single precision
B-9 Gold-Rader, coupled form floating point single precision
B-10 Kingsbury, coupled form floating point single precision
B-11 Zölzer, coupled form floating point single precision
B-12 Cabot (state-space hybrid), floating point single precision
B-13 State-space structure, floating point single precision
B-14 Ladder allpass, floating point single precision
B-15 Ladder Moorer, floating point single precision
B-16 Ladder Massie, floating point single precision
B-17 Lattice allpass, floating point single precision
B-18 Lattice Massie, floating point single precision

where,

S(x) performs single precision quantisation on data x.
D(x) performs double precision quantisation on data x.
$Q_m(x,n)$ performs floating point quantisation on data x, producing an n bit fractional wordlength in the mantissa.

$$y_i := S\left[2\cdot\left(D\left(a0\cdot x_i\right) + D\left(a1\cdot x_{i-1}\right) + D\left(a2\cdot x_{i-2}\right) + D\left(b1\cdot y_{i-1}\right) + D\left(b2\cdot y_{i-2}\right)\right)\right] \tag{B-1}$$

$$w_i := \left(D\left(a0\cdot x_i\right) + D\left(a1\cdot x_{i-1}\right) + D\left(a2\cdot x_{i-2}\right) + D\left(b1\cdot S\left(w_{i-1}\right)\right) + D\left(b2\cdot S\left(w_{i-2}\right)\right) + D\left(w_{i-1} - S\left(w_{i-1}\right)\right)\right)$$

$$y_i := S\left(w_i\right) \tag{B-2}$$

$$x_i := Qm\left(input_i, n\right) \tag{B-3}$$

$$y_i := Qm\left(Add\left(Add\left(Add\left(Add\left(Mlt\left(a0, x_i\right), Mlt\left(a1, x_{i-1}\right)\right), Mlt\left(a2, x_{i-2}\right)\right), Mlt\left(b1, y_{i-1}\right)\right), Mlt\left(b2, y_{i-2}\right)\right), n\right) \tag{B-4}$$

```
DF2 := | k ← 1
       | w_0 ← 0
       | w_1 ← 0
       | while k < MaximumSample
       |     k ← k + 1
       |     w_k ← Qm(Add(Add(x_k, Mlt(b1, w_{k-1})), Mlt(b2, w_{k-2})), n)
       |     y_k ← Qm(Add(Add(Mlt(a0, w_k), Mlt(a1, w_{k-1})), Mlt(a2, w_{k-2})), n)
       | return y
```

(B-5)

```
DF2T := | k ← 1
        | p_1 ← 0
        | w_1 ← 0
        | while k < MaximumSample
        |     k ← k + 1
        |     y_k ← Qm(Add(Mlt(x_k, a0), p_{k-1}), n)
        |     p_k ← Qm(Add(Add(Mlt(x_k, a1), Mlt(y_k, b1)), w_{k-1}), n)
        |     w_k ← Qm(Add(Mlt(x_k, a2), Mlt(y_k, b2)), n)
        | return y
```

(B-6)

```
DF2T :=  k ← 1
         p_1 ← 0
         w_1 ← 0
         while k < MaximumSample
             k ← k + 1
             y_k ← S[2·(D(x_k·a0) + p_(k−1))]
             p_k ← S(D(x_k·a1) + D(y_k·b1) + w_(k−1))
             w_k ← S(D(x_k·a2) + D(y_k·b2))
         return y
```

(B-7)

```
DF1T :=  k ← 1
         w_1 ← 0
         p_1 ← 0
         q_1 ← 0
         r_1 ← 0
         s_1 ← 0
         while k < MaximumSample
             k ← k + 1
             w_k ← Qm(Add(x_k, p_(k−1)), n)
             y_k ← Qm(Add(Mlt(a0, w_k), r_(k−1)), n)
             p_k ← Qm(Add(Mlt(b1, w_k), q_(k−1)), n)
             q_k ← Qm(Mlt(b2, w_k), n)
             r_k ← Qm(Add(Mlt(a1, w_k), s_(k−1)), n)
             s_k ← Qm(Mlt(a2, w_k), n)
         return y
```

(B-8)

GoldRader :=

$k \leftarrow 1$

$radius \leftarrow \sqrt{|-b2|}$

$GR\theta \leftarrow \operatorname{acos}\left(\frac{-b1}{-2\cdot\sqrt{|-b2|}}\right)$

$rcos \leftarrow radius \cdot \cos(GR\theta)$

$rsin \leftarrow radius \cdot \sin(GR\theta)$

$y1_1 \leftarrow 0$

$y2_1 \leftarrow 0$

$y_1 \leftarrow 0$

while $k < MaximumSample$

$k \leftarrow k + 1$

$zero_k \leftarrow Add\left(Mlt\left(x_k, a0\right), Add\left(Mlt\left(x_{k-1}, a1\right), Mlt\left(x_{k-2}, a2\right)\right)\right)$

$y1_k \leftarrow Qm\left(Add\left(Add\left(zero_k, Mlt\left(y1_{k-1}, rcos\right)\right), Mlt\left(y2_{k-1}, -rsin\right)\right), n\right)$

$y2_k \leftarrow Qm\left(Add\left(Mlt\left(y1_{k-1}, rsin\right), Mlt\left(y2_{k-1}, rcos\right)\right), n\right)$

$y_k \leftarrow Qm\left(Mlt\left(y2_k, \frac{1}{rsin}\right), n\right)$

return y

(B-9)

$$
\begin{aligned}
&\text{Kingsbury} := \\
&\quad k \leftarrow 1 \\
&\quad k1 \leftarrow \sqrt{1 - b1 - b2} \\
&\quad k2 \leftarrow \frac{1 + b2}{k1} \\
&\quad y1_1 \leftarrow 0 \\
&\quad y2_1 \leftarrow 0 \\
&\quad y3_1 \leftarrow 0 \\
&\quad y4_1 \leftarrow 0 \\
&\quad y_1 \leftarrow 0 \\
&\quad \text{while } k < \text{MaximumSample} \\
&\qquad k \leftarrow k + 1 \\
&\qquad zero_k \leftarrow \text{Add}\left(\text{Mlt}\left(x_k, a0\right), \text{Add}\left(\text{Mlt}\left(x_{k-1}, a1\right), \text{Mlt}\left(x_{k-2}, a2\right)\right)\right) \\
&\qquad y1_k \leftarrow \text{Add}\left(zero_k, \text{Mlt}\left(y4_{k-1}, k1\right)\right) \\
&\qquad y2_k \leftarrow \text{Add}\left(y1_k, \text{Qm}\left(y2_{k-1}, n\right)\right) \\
&\qquad y3_k \leftarrow \text{Qm}\left(\text{Add}\left(y2_k, \text{Mlt}\left(y4_{k-1}, k2\right)\right), n\right) \\
&\qquad y4_k \leftarrow \text{Qm}\left(\text{Add}\left(\text{Mlt}\left(y3_k, -k1\right), y4_{k-1}\right), n\right) \\
&\qquad y_k \leftarrow \text{Qm}\left(\text{Mlt}\left(y4_{k-1}, \frac{-1}{k1}\right), n\right) \\
&\quad \text{return } y
\end{aligned}
$$

(B-10)

$$
\begin{aligned}
\text{Zolzer} := \;& k \leftarrow 1 \\
& z1 \leftarrow \sqrt[3]{1 - b1 - b2} \\
& z2 \leftarrow \frac{1 + b2}{z1} \\
& y1_1 \leftarrow 0 \\
& y2_1 \leftarrow 0 \\
& y3_1 \leftarrow 0 \\
& y4_1 \leftarrow 0 \\
& y_1 \leftarrow 0 \\
& \text{while } k < \text{MaximumSample} \\
& \quad k \leftarrow k + 1 \\
& \quad zero_k \leftarrow \text{Add}\left(\text{Mlt}\left(x_k, a0\right), \text{Add}\left(\text{Mlt}\left(x_{k-1}, a1\right), \text{Mlt}\left(x_{k-2}, a2\right)\right)\right) \\
& \quad y1_k \leftarrow \text{Add}\left(zero_k, \text{Mlt}\left(y4_{k-1}, z1\right)\right) \\
& \quad y2_k \leftarrow \text{Add}\left(y1_k, \text{Qm}\left(y2_{k-1}, n\right)\right) \\
& \quad y3_k \leftarrow \text{Qm}\left(\text{Add}\left(\text{Mlt}\left(y2_k, z1\right), \text{Mlt}\left(y4_{k-1}, z2\right)\right), n\right) \\
& \quad y4_k \leftarrow \text{Qm}\left(\text{Add}\left(\text{Mlt}\left(y3_k, -z1\right), y4_{k-1}\right), n\right) \\
& \quad y_k \leftarrow \text{Qm}\left(\text{Mlt}\left(y4_{k-1}, \frac{-1}{z1^2}\right), n\right) \\
& \text{return } y
\end{aligned}
\tag{B-11}
$$

Cabot :=

$k \leftarrow 1$

$q11 \leftarrow \frac{b1}{2}$

$q22 \leftarrow \frac{b1}{2} \cdot 1$

$q12 \leftarrow \left(\frac{b1}{2} + \sqrt{-b2} \right)$

$q21 \leftarrow \left(\frac{b1}{2} - \sqrt{-b2} \right)$

$y1_1 \leftarrow 0$

$y2_1 \leftarrow 0$

$y3_1 \leftarrow 0$

$y_1 \leftarrow 0$

while $k < MaximumSample$

$k \leftarrow k + 1$

$zero_k \leftarrow Add\left(Add\left(Mlt\left(input_k, \frac{a0}{2} \right), Mlt\left(input_{k-1}, \frac{a1}{2} \right) \right), Mlt\left(input_{k-2}, \frac{a2}{2} \right) \right)$

$y1_k \leftarrow Qm\left(Add\left(zero_k, Add\left(Mlt\left(y2_{k-1}, q12 \right), Mlt\left(y1_{k-1}, q11 \right) \right) \right), n \right)$

$y2_k \leftarrow Qm\left(Add\left(zero_k, Add\left(Mlt\left(y1_{k-1}, q21 \right), Mlt\left(y2_{k-1}, q22 \right) \right) \right), n \right)$

$y_k \leftarrow Qm\left(Add\left(y1_k, y2_k \right), n \right)$

return y

(B-12)

Statespace :=

$k \leftarrow 1$

$q11 \leftarrow \frac{b1}{2}$

$q22 \leftarrow q11$

$q12 \leftarrow \frac{x1 \cdot (1 + k2)}{k1^2}$

$q21 \leftarrow \frac{x2}{1 + k2}$

$c1 \leftarrow \frac{1 + k2}{2}$

$c2 \leftarrow \frac{k1}{2}$

$s1 \leftarrow \frac{k1}{1 + k2}$

$s2 \leftarrow 1$

$y1_1 \leftarrow 0$

$y2_1 \leftarrow 0$

$y3_1 \leftarrow 0$

$y_1 \leftarrow 0$

while $k < MaximumSample$

$k \leftarrow k + 1$

$y1_k \leftarrow Qm\left(Add\left(Mlt\left(y1_{k-1}, q11\right), Add\left(Mlt\left(x_k, c1\right), Mlt\left(y2_{k-1}, q12\right)\right)\right), n\right)$

$y2_k \leftarrow Qm\left(Add\left(Mlt\left(y1_{k-1}, q21\right), Add\left(Mlt\left(x_k, c2\right), Mlt\left(y2_{k-1}, q22\right)\right)\right), n\right)$

$y_k \leftarrow Qm\left(Add\left(Add\left(Mlt\left(y1_{k-1}, s1\right), Mlt\left(y2_{k-1}, s2\right)\right), Mlt\left(x_k, a0\right)\right), n\right)$

return y

(B-13)

LadderAllpass :=

$i \leftarrow 1$

$k1 \leftarrow \dfrac{-b1}{1+-b2}$

$k2 \leftarrow -b2$

$c1 \leftarrow \sqrt{1-(k1)^2}$

$c2 \leftarrow \sqrt{1-(k2)^2}$

$R_1 \leftarrow 0$

$Q_1 \leftarrow 0$

while $i < MaximumSample$

$i \leftarrow i+1$

$P_i \leftarrow Qm\left(Add\left(Mlt\left(x_i, c2\right), Mlt\left(R_{i-1}, -k2\right)\right), n\right)$

$Q_i \leftarrow Qm\left(Add\left(Mlt\left(P_i, c1\right), Mlt\left(Q_{i-1}, -k1\right)\right), n\right)$

$R_i \leftarrow Qm\left(Add\left(Mlt\left(P_i, k1\right), Mlt\left(Q_{i-1}, c1\right)\right), n\right)$

$y_i \leftarrow Qm\left(Add\left(Mlt\left(R_{i-1}, c2\right), Mlt\left(x_i, k2\right)\right), n\right)$

return y

(B-14)

LadderMoorer :=

$i \leftarrow 1$

$k1 \leftarrow \dfrac{-b1}{1+-b2}$

$k2 \leftarrow -b2$

$c1 \leftarrow \sqrt{1-(k1)^2}$

$c2 \leftarrow \sqrt{1-(k2)^2}$

$v0 \leftarrow \dfrac{1}{c1 \cdot c2} \cdot (a0 - c2 \cdot k1 \cdot v1 - k2 \cdot v2)$

$v1 \leftarrow \dfrac{1}{c2} \cdot (a1 - a2 \cdot -b1)$

$v2 \leftarrow a2$

$R_1 \leftarrow 0$

$Q_1 \leftarrow 0$

while $i < MaximumSample$

$i \leftarrow i+1$

$P_i \leftarrow Qm\left(Add\left(Mlt\left(x_i, c2\right), Mlt\left(R_{i-1}, -k2\right)\right), n\right)$

$Q_i \leftarrow Qm\left(Add\left(Mlt\left(P_i, c1\right), Mlt\left(Q_{i-1}, -k1\right)\right), n\right)$

$R_i \leftarrow Qm\left(Add\left(Mlt\left(P_i, k1\right), Mlt\left(Q_{i-1}, c1\right)\right), n\right)$

$Y_i \leftarrow Qm\left(Add\left(Mlt\left(R_{i-1}, c2\right), Mlt\left(x_i, k2\right)\right), n\right)$

$y_i \leftarrow Qm\left(Add\left(Add\left(Mlt\left(Y_i, v2\right), Mlt\left(R_i, v1\right)\right), Mlt\left(Q_i, v0\right)\right), n\right)$

return y

(B-15)

LadderMassie :=

$i \leftarrow 1$

$k1 \leftarrow \frac{-b1}{1+-b2}$

$k2 \leftarrow -b2$

$c1 \leftarrow \sqrt{1-(k1)^2}$

$c2 \leftarrow \sqrt{1-(k2)^2}$

$MG1 \leftarrow \left(1-10^{\frac{G}{20}}\right)\cdot 0.5$

$MG2 \leftarrow \left(1+10^{\frac{G}{20}}\right)\cdot 0.5$

$R_1 \leftarrow 0$

$Q_1 \leftarrow 0$

while $i < MaximumSample$

- $i \leftarrow i+1$
- $P_i \leftarrow Qm\left(Add\left(Mlt\left(x_i, c2\right), Mlt\left(R_{i-1}, -k2\right)\right), n\right)$
- $Q_i \leftarrow Qm\left(Add\left(Mlt\left(P_i, c1\right), Mlt\left(Q_{i-1}, -k1\right)\right), n\right)$
- $R_i \leftarrow Qm\left(Add\left(Mlt\left(P_i, k1\right), Mlt\left(Q_{i-1}, c1\right)\right), n\right)$
- $Y_i \leftarrow Qm\left(Add\left(Mlt\left(R_{i-1}, c2\right), Mlt\left(x_i, k2\right)\right), n\right)$
- $y_i \leftarrow Qm\left(Add\left(Mlt\left(Y_i, MG1\right), Mlt\left(x_i, MG2\right)\right), n\right)$

return y

(B-16)

LatticeAllpass :=

$i \leftarrow 1$

$k1 \leftarrow \frac{-b1}{1+-b2}$

$k2 \leftarrow -b2$

$R_1 \leftarrow 0$

$Q_1 \leftarrow 0$

while $i < MaximumSample$

- $i \leftarrow i+1$
- $P_i \leftarrow Add\left(x_i, Mlt\left(R_{i-1}, -k2\right)\right)$
- $Q_i \leftarrow Qm\left(Add\left(P_i, Mlt\left(Q_{i-1}, -k1\right)\right), n\right)$
- $R_i \leftarrow Qm\left(Add\left(Q_{i-1}, Mlt\left(Q_i, k1\right)\right), n\right)$
- $y_i \leftarrow Qm\left(Add\left(R_{i-1}, Mlt\left(P_i, k2\right)\right), n\right)$

return y

(B-17)

LatticeMassie :=

$i \leftarrow 1$

$k1 \leftarrow \dfrac{-b1}{1+-b2}$

$k2 \leftarrow -b2$

$MG1 \leftarrow \left(1 - 10^{\frac{G}{20}}\right) \cdot 0.5$

$MG2 \leftarrow \left(1 + 10^{\frac{G}{20}}\right) \cdot 0.5$

$R_1 \leftarrow 0$

$Q_1 \leftarrow 0$

while $i < MaximumSample$

- $i \leftarrow i + 1$
- $P_i \leftarrow Add\left(x_i, Mlt\left(R_{i-1}, -k2\right)\right)$
- $Q_i \leftarrow Qm\left(Add\left(P_i, Mlt\left(Q_{i-1}, -k1\right)\right), n\right)$
- $R_i \leftarrow Qm\left(Add\left(Q_{i-1}, Mlt\left(Q_i, k1\right)\right), n\right)$
- $Y_i \leftarrow Qm\left(Add\left(R_{i-1}, Mlt\left(P_i, k2\right)\right), n\right)$
- $y_i \leftarrow Qm\left(Add\left(Mlt\left(Y_i, MG1\right), Mlt\left(x_i, MG2\right)\right), n\right)$

return y

(B-18)

Appendix C DSP platform filter implementations

Appendix C contains the Analog Devices SHARC 2106x family assembly code for the DF1 topology implementations in 32 bit fixed and floating point arithmetic. These algorithms were executed on a ADSP-21061 DSP platform. Noise and distortion measurements of the DF1 topology implementations were taken, via an AES/EBU interface, by 'Audio Precision Audio Measurement System Two Cascade'. These measurements were compared to the emulated DF1 topologies in the Mathcad environment. The magnitude frequency response of the bell filter used in the test is shown in Figure C-1.

C.1 DF1 floating point implementation in SHARC assembly code

```
/************************************************************************/
/* SHARC 21061 evm filter test project                                  */
/* file: df1rnd32.asm                                                   */
/* date : july2000 RobC                                                 */
/************************************************************************/

#include "def21060.h"

/************************************************************************/
.segment /dm    seg_dmda;
//preassembler constant data formats and quantisation defaults
//.PRECISION=32;
//.ROUND_NEAREST;

.var       TapData[4] =      0x00000000,
                             0x00000000,
                             0x00000000,
                             0x00000000;

.var       Coeffcients[5] =  -0.9975987673,
                             1.9975919724,
                             0.9986481667,
                             -1.9975919724,
                             0.9989504814;
.endseg;

/************************************************************************/
.segment/pm seg_pmco;
```

```
start:  r0=0x000231F3;
        dm(STCTL1)=r0;
        r0=0x000231F3;
        dm(SRCTL1)=r0;
        nop;
        nop;
        r0=0x12345678;
        dm(TX1)=r0;
        nop;
        nop;

BIT SET MODE1 0x00010000;                       /* rounding, 32 bit arithmetic */

        b4=TapData;
        i4=TapData;                             /* tap data pointer - circular */
        l4=4;                                    /* length 4 */

        bit set imask       SPR1I;              /* enable sport1 rx interrupt */
        bit set mode1       IRPTEN;             /* global interrupt enable   */

        r0=0x12345678;
        dm(TX1)=r0;
        bit set astat FLG2 | FLG3;

wait:   nop;
        jump    wait;

/*************************************************************************/
/* SPort1 RX interrupt                                                   */
/*************************************************************************/

s1rx:

        r0=dm(RX1);

        i0=Coeffcients;             /* initialise to start of coefficient list */

        f0=FLOAT r0;

        f4=dm(i4,0);                            /* fetch y(n-2)    */
        f8=dm(i0,1);                            /* fetch coef b2 */
        dm(i4,1)=f0;                            /* overwrite y(n-2) with x(n) */
        f12=f4*f8,   f4=dm(i4,1);               /* y(n-2) * b2, fetch y(n-1) */
        f8=dm(i0,1);                            /* fetch coef b1 */
        f7=f4*f8,    f4=dm(i4,1);               /* y(n-1) * b1 , fetch x(n-2) */
        f12=f12+f7, f8=dm(i0,1);                /* fetch coef a2 */
        f7=f4*f8,   f4=dm(i4,-1);               /* x(n-2) * a2 , fetch x(n-1) */
        f12=f12+f7, f8=dm(i0,1);                /* fetch coef a1 */
        f7=f4*f8;                               /* x(n-1) * a1 */
        f12=f12+f7, f8=dm(i0,1);                /* fetch coef a0 */
        f7=f0*f8;                               /* x(n) * a0 */
        f12=f12+f7;                             /*    */
        dm(i4,-1)=f12;                          /*    */

        r0 = FIX    f12;                    /* float to fixed conversion */
        dm(TX1)=r0;
        rti;
```

C.2 DF1 fixed point implementation in SHARC assembly code

```
/************************************************************************/
//requires the coefficient variables declaration as follows

.var    Coeffcients[5] =  0xc0275780,
                          0x7fd88c00,
                          0x3fe9da00,
                          0x80277400,
                          0x3feece00;

/************************************************************************/

/************************************************************************/
/*      Fixed point DF1                                                 */
/*      rounding quantiser                                              */
/*      coefficient scaling of 2                                        */
/*      insert into the serial receive interrupt code space             */
/************************************************************************/
s1rx:

    r0=dm(RX1);

    i0=Coeffcients;                 /* initialise to start of coefficient list */

    r4=dm(i4,0);                    /* fetch y(n-2)                    */
    r8=dm(i0,1);                    /* fetch coef b2                   */
    dm(i4,1)=r0;                    /* overwrite y(n-2) with x(n)      */
    mrf=r4*r8,          r4=dm(i4,1);/* y(n-2) * b2, fetch y(n-1)       */
    r8=dm(i0,1);                    /* fetch coef b1                   */
    mrf=mrf + r4*r8,    r4=dm(i4,1);/* y(n-1) * b1 , fetch x(n-2)      */
    r8=dm(i0,1);                    /* fetch coef a2                   */
    mrf=mrf + r4*r8,    r4=dm(i4,-1);/* x(n-2) * a2 , fetch x(n-1)      */
    r8=dm(i0,1);                    /* fetch coef a1                   */
    mrf=mrf + r4*r8;                /* x(n-1) * a1                     */
    r8=dm(i0,1);                    /* fetch coef a0                   */
    mrf=mrf + r0*r8;                /* x(n) * a0                       */
    nop;
    r2 = RND mrf;
    r1 = ASHIFT r2 BY 1;            /* arithmetic shift left       */
    dm(i4,-1)=r1;                   /*          */
    dm(TX1)=r1;
    rti;

/************************************************************************/

/************************************************************************/
/*      Fixed point DF1                                                 */
/*      truncation quantiser                                            */
/*      coefficient scaling of 2                                        */
/*      insert into the serial receive interrupt code space             */
/************************************************************************/
s1rx:

    r0=dm(RX1);
    i0=Coeffcients;                 /* initialise to start of coefficient list */
    r4=dm(i4,0);                    /* fetch y(n-2)                    */
    r8=dm(i0,1);                    /* fetch coef b2                   */
    dm(i4,1)=r0;                    /* overwrite y(n-2) with x(n)      */
    mrf=r4*r8,          r4=dm(i4,1);/* y(n-2) * b2, fetch y(n-1)       */
    r8=dm(i0,1);                    /* fetch coef b1                   */
    mrf=mrf + r4*r8,    r4=dm(i4,1);/* y(n-1) * b1 , fetch x(n-2)      */
```

```
r8=dm(i0,1);                          /* fetch coef a2                    */
mrf=mrf + r4*r8,     r4=dm(i4,-1);    /* x(n-2) * a2 , fetch x(n-1)       */
r8=dm(i0,1);                          /* fetch coef a1                    */
mrf=mrf + r4*r8;                      /* x(n-1) * a1                      */
r8=dm(i0,1);                          /* fetch coef a0                    */
mrf=mrf + r0*r8;                      /* x(n) * a0                        */
nop;
r2 = mr1f;
r1 = ASHIFT r2 BY 1;                  /* arithmetic shift left            */
dm(i4,-1)=r1;                         /*                                  */
dm(TX1)=r1;
rti;

/*************************************************************************/
```

C.1 Measured magnitude frequency response of test filter

Figure C-1 shows the magnitude frequency response of the test filter used in comparative tests of the emulated DF1 topology with the DF1 topology implemented on the 32 bit DSP platform. The test filter used was a bell filter, tuned to a centre frequency, Fc equal to 20 Hz, Q factor of 8.65 with an attenuation of –18 dB.

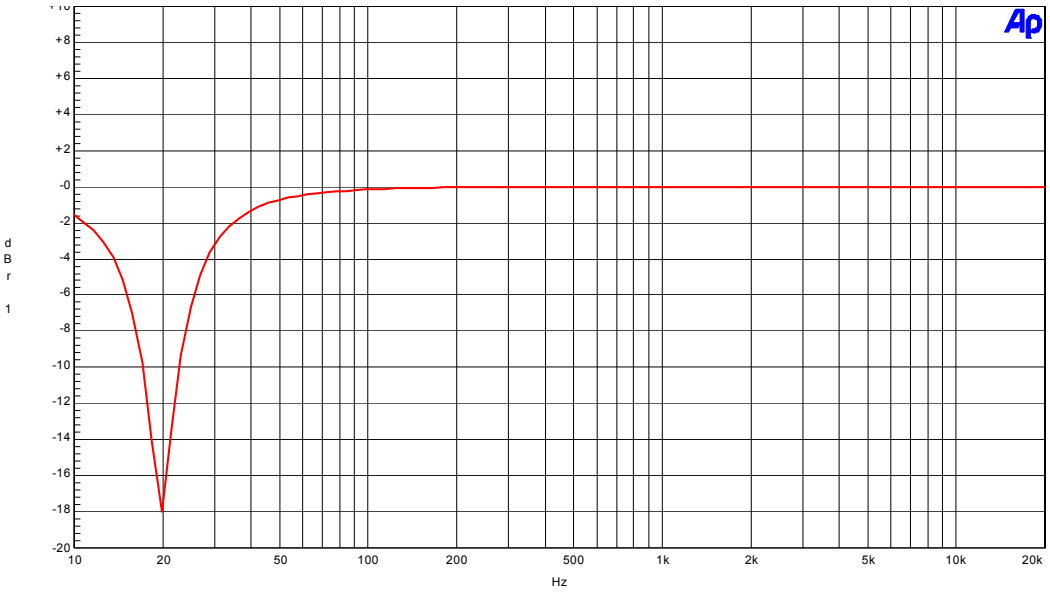

Figure C-1 Measured frequency response of bell filter implemented on SHARC 21061 DSP under floating point arithmetic (F=20Hz, Q=8.65, G=-18dB).

Appendix D Coefficient update topology functions

Appendix D contains Mathcad functions used in the investigation of filter topology behaviour during coefficient update. This includes the functions for coefficient calculation, functions implementing the discrete-time varying filter topologies and the implementation of the various coefficient and parameter interpolation schemes.

D.1 Coefficient calculation

This section contains coefficient calculation functions for the various filter types considered in this project. The coefficients calculated in this section are specifically for the direct form topology (five coefficients for a z-plane biquadratic). Coefficient mappings for each of the other topologies are given in Section D.2 with the respective filter topology implementation. The bilinear z-transform is used as the example mapping technique for the low and high pass filters, LF and HF shelving filters and the bell filter. The notch filter coefficient realisation uses the direct pole zero placement technique. Note, Ts is the sampling interval (1/Fs). Fs is the sampling frequency

Equations (D-1) and (D-2) are the tuned frequency and Q prewarping functions used in conjunction with the bilinear z-transform. If the tuned frequency, Fc, and the Q factor are to be pre-warped, then variables F and Q in the following coefficient calculation functions are effectively replaced by the prewarped parameters produced by (D-1) and (D-2).

$$\mathrm{Fwarp}(\mathrm{Fc}) := \tan\left(\frac{\pi \cdot \mathrm{Fc}}{\mathrm{Fs}}\right) \cdot \frac{\mathrm{Fs}}{\pi} \qquad \text{(D-1)}$$

$$\mathrm{Qwarp}(\mathrm{Fc}, \mathrm{Q}) := \frac{\mathrm{Fc}}{\tan\left(\frac{\pi \cdot \mathrm{Fc}}{\mathrm{Fs}}\right) \cdot \frac{\mathrm{Fs}}{\pi}} \cdot \mathrm{Q} \qquad \text{(D-2)}$$

Coefficient calculation for second order low pass filter is shown in (D-3), where F is the tuned frequency (-3 dB corner frequency) and Q (quality factor).

$$LPFa0(F,Q) := \frac{4\cdot\pi^2\cdot F^2\cdot Q}{k^2\cdot Q + 4\cdot\pi^2\cdot F^2\cdot Q + 2\cdot\pi\cdot F\cdot k} \qquad LPFa1(F,Q) := 2\cdot LPFa0(F,Q) \qquad LPFa2(F,Q) := LPFa0(F,Q)$$

$$LPFb1(F,Q) := \frac{-\left(-2\cdot k^2\cdot Q + 8\cdot\pi^2\cdot F^2\cdot Q\right)}{k^2\cdot Q + 4\cdot\pi^2\cdot F^2\cdot Q + 2\cdot\pi\cdot F\cdot k} \qquad LPFb2(F,Q) := \frac{-\left(-2\cdot\pi\cdot F\cdot k + k^2\cdot Q + 4\cdot\pi^2\cdot F^2\cdot Q\right)}{k^2\cdot Q + 4\cdot\pi^2\cdot F^2\cdot Q + 2\cdot\pi\cdot F\cdot k}$$

(D-3)

where,

$$k := \frac{2}{Ts}.$$

Coefficient calculation for second order high pass filter is shown in (D-4), where F is the tuned frequency (-3 dB corner frequency) and Q (quality factor).

$$HPFa0(F,Q) := \frac{k^2\cdot Q}{k^2\cdot Q + 4\cdot\pi^2\cdot F^2\cdot Q + 2\cdot\pi\cdot F\cdot k} \qquad HPFa1(F,Q) := -2\cdot HPFa0(F,Q) \qquad HPFa2(F,Q) := HPFa0(F,Q)$$

$$HPFb1(F,Q) := \frac{-\left(-2\cdot k^2\cdot Q + 8\cdot\pi^2\cdot F^2\cdot Q\right)}{k^2\cdot Q + 4\cdot\pi^2\cdot F^2\cdot Q + 2\cdot\pi\cdot F\cdot k} \qquad HPFb2(F,Q) := \frac{-\left(-2\cdot\pi\cdot F\cdot k + k^2\cdot Q + 4\cdot\pi^2\cdot F^2\cdot Q\right)}{k^2\cdot Q + 4\cdot\pi^2\cdot F^2\cdot Q + 2\cdot\pi\cdot F\cdot k}$$

(D-4)

Coefficient calculation for second order bell filter is shown in (D-5), where F is the tuned centre frequency and Q (quality factor). The variables A and B determine the gain G. Variables A and B also determine the relationship between Q and Gain. The expressions shown for A and B provide a constant Q symmetrical with cut and boost.

$$A(G) := \begin{cases} 10^{\frac{G}{20}} & \text{if } 10^{\frac{G}{20}} \geq 1 \\ 1 & \text{otherwise} \end{cases} \qquad B(G) := \begin{cases} 1 & \text{if } 10^{\frac{G}{20}} \geq 1 \\ \frac{1}{10^{\frac{G}{20}}} & \text{otherwise} \end{cases}$$

$$Bella0(F,G,Q) := \frac{2\cdot A(G)\cdot\pi\cdot\frac{F}{Q}\cdot k + k^2 + 4\cdot\pi^2\cdot F^2}{2\cdot B(G)\cdot\pi\cdot\frac{F}{Q}\cdot k + k^2 + 4\cdot\pi^2\cdot F^2} \qquad Bella1(F,G,Q) := \frac{8\cdot\pi^2\cdot F^2 - 2\cdot k^2}{2\cdot B(G)\cdot\pi\cdot\frac{F}{Q}\cdot k + k^2 + 4\cdot\pi^2\cdot F^2}$$

$$Bella2(F,G,Q) := \frac{k^2 + 4\cdot\pi^2\cdot F^2 - 2\cdot A(G)\cdot\pi\cdot\frac{F}{Q}\cdot k}{2\cdot B(G)\cdot\pi\cdot\frac{F}{Q}\cdot k + k^2 + 4\cdot\pi^2\cdot F^2}$$

$$Bellb1(F,G,Q) := \frac{-\left(8\cdot\pi^2\cdot F^2 - 2\cdot k^2\right)}{2\cdot B(G)\cdot\pi\cdot\frac{F}{Q}\cdot k + k^2 + 4\cdot\pi^2\cdot F^2} \qquad Bellb2(F,G,Q) := \frac{-\left(k^2 + 4\cdot\pi^2\cdot F^2 - 2\cdot B(G)\cdot\pi\cdot\frac{F}{Q}\cdot k\right)}{2\cdot B(G)\cdot\pi\cdot\frac{F}{Q}\cdot k + k^2 + 4\cdot\pi^2\cdot F^2}$$

(D-5)

Coefficient calculation schemes for second order LF and HF shelving filters are given in (D-5) and (D-6), where F is the tuned corner frequency and Q is the slope control. The variables A and B determine the gain G. Variables A and B also determine the relationship between Q and Gain. The example expressions given in (D-6) produce a non-constant Q relationship symmetrical response for cut and boost.

$$A(G) := \left[1 + \frac{\sqrt{10^{\frac{G}{20}}} - 1}{1 + \sqrt{10^{\frac{G}{20}}}}\right]\cdot\pi \qquad B(G) := \left[1 - \frac{\sqrt{10^{\frac{G}{20}}} - 1}{1 + \sqrt{10^{\frac{G}{20}}}}\right]\cdot\pi$$

(D-6)

$$\text{LFshelfa0}(G,F) := \frac{k^2\cdot Q + 2\cdot k\cdot A(G)\cdot 2\cdot\pi\cdot F + (A(G)\cdot 2\cdot\pi\cdot F)^2}{k^2\cdot Q + 2\cdot k\cdot(B(G)\cdot 2\cdot\pi\cdot F) + (B(G)\cdot 2\cdot\pi\cdot F)^2}$$

$$\text{LFshelfa1}(G,F) := \frac{-2\cdot k^2\cdot Q + 2\cdot(A(G)\cdot 2\cdot\pi\cdot F)^2}{k^2\cdot Q + 2\cdot k\cdot(B(G)\cdot 2\cdot\pi\cdot F) + (B(G)\cdot 2\cdot\pi\cdot F)^2}$$

$$\text{LFshelfa2}(G,F) := \frac{-2\cdot k\cdot(A(G)\cdot 2\cdot\pi\cdot F) + k^2\cdot Q + (A(G)\cdot 2\cdot\pi\cdot F)^2}{k^2\cdot Q + 2\cdot k\cdot(B(G)\cdot 2\cdot\pi\cdot F) + (B(G)\cdot 2\cdot\pi\cdot F)^2}$$

$$\text{LFshelfb1}(G,F) := \frac{-\left[-2\cdot k^2\cdot Q + 2\cdot(B(G)\cdot 2\cdot\pi\cdot F)^2\right]}{k^2\cdot Q + 2\cdot k\cdot(B(G)\cdot 2\cdot\pi\cdot F) + (B(G)\cdot 2\cdot\pi\cdot F)^2}$$

$$\text{LFshelfb2}(G,F) := -\frac{-2\cdot k\cdot(B(G)\cdot 2\cdot\pi\cdot F) + k^2\cdot Q + (B(G)\cdot 2\cdot\pi\cdot F)^2}{k^2\cdot Q + 2\cdot k\cdot(B(G)\cdot 2\cdot\pi\cdot F) + (B(G)\cdot 2\cdot\pi\cdot F)^2}$$

(D-7)

$$\text{HFshelfa0}(G,F) := \frac{A(G)^2\cdot k^2 + 4\cdot\pi^2\cdot F^2\cdot Q + 4\cdot A(G)\cdot k\cdot\pi\cdot F}{B(G)^2\cdot k^2 + 4\cdot\pi^2\cdot F^2\cdot Q + 4\cdot B(G)\cdot k\cdot\pi\cdot F}$$

$$\text{HFshelfa1}(G,F) := \frac{-2\cdot k^2\cdot Q + 2\cdot A(G)^2}{B(G)^2\cdot k^2 + 4\cdot\pi^2\cdot F^2\cdot Q + 4\cdot B(G)\cdot k\cdot\pi\cdot F}$$

$$\text{HFshelfa2}(G,F) := \frac{-2\cdot k\cdot A(G) + k^2\cdot Q + A(G)^2}{B(G)^2\cdot k^2 + 4\cdot\pi^2\cdot F^2\cdot Q + 4\cdot B(G)\cdot k\cdot\pi\cdot F}$$

$$\text{HFshelfb1}(G,F) := \frac{-\left(-2\cdot k^2\cdot Q + 2\cdot B(G)^2\right)}{B(G)^2\cdot k^2 + 4\cdot\pi^2\cdot F^2\cdot Q + 4\cdot B(G)\cdot k\cdot\pi\cdot F}$$

$$\text{HFshelfb2}(G,F) := -\frac{-2\cdot k\cdot B(G) + k^2\cdot Q + B(G)^2}{B(G)^2\cdot k^2 + 4\cdot\pi^2\cdot F^2\cdot Q + 4\cdot B(G)\cdot k\cdot\pi\cdot F}$$

(D-8)

Coefficient calculation schemes for a second order notch filter are given in (D-9) where F is the tuned notch frequency. Note r determines the radius of the complex conjugate poles on the z-plane. This radius r controls the bandwidth of the notch filter independent of tuned frequency, for example r = 0.9995 produces a –3dB bandwidth of 7.5 Hz at a sampling frequency of 48 kHz.

$$\text{notcha0} := 1 \qquad \text{notcha1}(F) := -2\cdot\cos\left(2\cdot\pi\cdot\frac{F}{Fs}\right) \qquad \text{notcha2} := 1$$

$$\text{notchb1}(F) := 2\cdot r\cdot\cos\left(2\cdot\pi\cdot\frac{F}{Fs}\right) \qquad \text{notchb2} := -r^2$$

(D-9)

D.2 Discrete-time varying filter topologies

This section lists the discrete-time varying filter topologies used in the investigation. The coefficients are implemented as arrays where each sample instance has a separate coefficient entry. For example the DF1 topology example in (D-10) uses a coefficient array set shown as '$IntEa0_i$, $IntEa1_i$, $IntEa2_i$, $IntEb1_i$, $IntEb2_i$'. These coefficients, for example, could be a pre-calculated coefficient set using exponential interpolation across the frequency response state change. The non-direct form topologies also include the relevant coefficient mappings required to map the biquadratic coefficients to the coefficients used in each of the topologies.

D-10 DF1, using coefficient set $IntEa0_i$, $IntEa1_i$, $IntEa2_i$, $IntEb1_i$, $IntEb2_i$.

D-11 DF2, using a generic coefficient set ($a0_k$, $a1_k$, $a2_k$, $b1_k$, $b2_k$).

D-12 DF1T, using a generic coefficient set.

D-13 DF2T, using a generic coefficient set.

D-14 DF2T (using coefficient compensation), generic coefficient set.

D-15 Gold-Rader, using a generic coefficient set.

D-16 Kingsbury, using a generic coefficient.

D-17 Zölzer, using a generic coefficient.

D-18 Cabot, using a generic coefficient.

D-19 State-space, using a generic coefficient.

D-20 Ladder and lattice allpass, using a generic coefficient.

D-21 Ladder Massie, using a generic coefficient.

D-22 Lattice Massie, using a generic coefficient.

D-23 Ladder Moorer, using a generic coefficient.

$$df1_i := \left[\left(IntEa0_i \cdot input_i\right) + \left(IntEa1_i \cdot input_{i-1}\right) + \left(IntEa2_i \cdot input_{i-2}\right) + \left(IntEb1_i \cdot df1_{i-1}\right) + \left(IntEb2_i \cdot df1_{i-2}\right) \right] \tag{D-10}$$

$$\begin{aligned} df2 := \; & k \leftarrow 1 \\ & w_0 \leftarrow 0 \\ & w_1 \leftarrow 0 \\ & \text{while } k < MaximumSample \\ & \quad k \leftarrow k + 1 \\ & \quad w_k \leftarrow input_k + b1_k \cdot w_{k-1} + b2_k \cdot w_{k-2} \\ & \quad y_k \leftarrow a0_k \cdot w_k + a1_k \cdot w_{k-1} + a2_k \cdot w_{k-2} \\ & \text{return } y \end{aligned} \tag{D-11}$$

```
df1t := | k ← 1
        | w_1 ← 0
        | p_1 ← 0
        | q_1 ← 0
        | r_1 ← 0
        | s_1 ← 0
        | while k < MaximumSample
        |     | k ← k + 1
        |     | w_k ← input_k + p_(k-1)
        |     | y_k ← w_k · a0_k + r_(k-1)
        |     | p_k ← q_(k-1) + w_k · b1_k
        |     | q_k ← w_k · b2_k
        |     | r_k ← s_(k-1) + w_k · a1_k
        |     | s_k ← w_k · a2_k
        | return y
```

(D-12)

```
df2t := | k ← 1
        | p_1 ← 0
        | w_1 ← 0
        | while k < MaximumSample
        |     | k ← k + 1
        |     | y_k ← input_k · a0_k + p_(k-1)
        |     | p_k ← input_k · a1_k + y_k · b1_k + w_(k-1)
        |     | w_k ← input_k · a2_k + y_k · b2_k
        | return y
```

(D-13)

```
df2tc := | k ← 1
         | p_1 ← 0
         | w_1 ← 0
         | while k < MaximumSample
         |     | k ← k + 1
         |     | y_k ← input_k · a0_(k-2) + p_(k-1)
         |     | p_k ← input_k · a1_(k-1) + y_k · b1_(k-1) + w_(k-1)
         |     | w_k ← input_k · a2_k + y_k · b2_k
         | return y
```

(D-14)

$$
\begin{aligned}
&\text{GoldRader} := \left|\begin{array}{l}
k \leftarrow 1 \\
rcos \leftarrow GRCos \\
rsin \leftarrow GRSin \\
y1_1 \leftarrow 0 \\
y2_1 \leftarrow 0 \\
youtput_1 \leftarrow 0 \\
\text{while } k < MaximumSample \\
\quad \left|\begin{array}{l}
k \leftarrow k + 1 \\
y1_k \leftarrow input_k \cdot a0_k + input_{k-1} \cdot a1_k + input_{k-2} \cdot a2_k + y1_{k-1} \cdot rcos_k + y2_{k-1} \cdot -rsin_k \\
y2_k \leftarrow y1_{k-1} \cdot rsin_k + y2_{k-1} \cdot rcos_k \\
youtput_k \leftarrow y2_k \cdot \dfrac{1}{rsin_k}
\end{array}\right. \\
\text{return } youtput
\end{array}\right.
\end{aligned}
$$

where

$$
GoldRader\theta_k := \operatorname{acos}\left(\frac{-b1_i}{-2\cdot\sqrt{|-b2_k|}}\right) \qquad GoldRaderR_k := \sqrt{|-b2_k|}
$$

$$
GRCos_k := GoldRaderR_k \cdot \cos\left(GoldRader\theta_k\right) \qquad GRSin_k := GoldRaderR_k \cdot \sin\left(GoldRader\theta_k\right)
$$

(D-15)

```
Kingsbury := | k ← 1
             | k1 ← Kingsburyk1
             | k2 ← Kingsburyk2
             | y1_1 ← 0
             | y2_1 ← 0
             | y3_1 ← 0
             | y4_1 ← 0
             | youtput_1 ← 0
             | while k < MaximumSample
             |     k ← k + 1
             |     y1_k ← input_k · a0_k + input_(k-1) · a1_k + input_(k-2) · a2_k + y4_(k-1) · k1_k
             |     y2_k ← y1_k + y2_(k-1)
             |     y3_k ← y2_k + y4_(k-1) · k2_k
             |     y4_k ← y3_k · -k1_k + y4_(k-1)
             |     youtput_k ← y4_(k-1) · (-1 / k1_k)
             | return youtput
```

where

$$\text{Kingsburyk1}_k := \sqrt{1 - b1_k - b2_k} \qquad \text{Kingsburyk2}_k := \frac{1 + b2_k}{\text{Kingsburyk1}_k}$$

(D-16)

$$
\begin{aligned}
&\text{Zolzer} := \left|\begin{array}{l}
k \leftarrow 1 \\
z1 \leftarrow \text{Zolzerz1} \\
z2 \leftarrow \text{Zolzerz2} \\
y1_1 \leftarrow 0 \\
y2_1 \leftarrow 0 \\
y3_1 \leftarrow 0 \\
y4_1 \leftarrow 0 \\
\text{youtput}_1 \leftarrow 0 \\
\text{while } k < \text{MaximumSample} \\
\quad \left|\begin{array}{l}
k \leftarrow k + 1 \\
y1_k \leftarrow \text{input}_k \cdot a0_k + \text{input}_{k-1} \cdot a1_k + \text{input}_{k-2} \cdot a2_k + y4_{k-1} \cdot z1_k \\
y2_k \leftarrow y1_k + y2_{k-1} \\
y3_k \leftarrow y2_k \cdot z1_k + y4_{k-1} \cdot z2_k \\
y4_k \leftarrow y3_k \cdot -z1_k + y4_{k-1} \\
\text{youtput}_k \leftarrow y4_{k-1} \cdot \dfrac{-1}{(z1_k)^2}
\end{array}\right. \\
\text{return youtput}
\end{array}\right.
\end{aligned}
$$

where

$$\text{Zolzerz1}_k := \sqrt[3]{1 - b1_k - b2_k} \qquad \text{Zolzerz2}_k := \frac{1 + b2_k}{\text{Zolzerz1}_k}$$

(D-17)

Cabot := | $k \leftarrow 1$
| $q11 \leftarrow Cabotq11$
| $q22 \leftarrow Cabotq22$
| $q12 \leftarrow Cabotq12$
| $q21 \leftarrow Cabotq21$
| $y1_1 \leftarrow 0$
| $y2_1 \leftarrow 0$
| $y3_1 \leftarrow 0$
| $youtput_1 \leftarrow 0$
| while $k < MaximumSample$
| | $k \leftarrow k + 1$
| | $zero_k \leftarrow \left(input_k \cdot a0_k + input_{k-1} \cdot a1_k + input_{k-2} \cdot a2_k\right) \cdot 0.5$
| | $y1_k \leftarrow zero_k + q12_k \cdot y2_{k-1} + q11_k \cdot y1_{k-1}$
| | $y2_k \leftarrow zero_k + q21_k \cdot y1_{k-1} + q22_k \cdot y2_{k-1}$
| | $y3_k \leftarrow y1_{k-1} + y2_{k-1}$
| | $youtput_k \leftarrow y3_k$
| return youtput

where

$$Cabotq11_k := \frac{b1_k}{2} \qquad Cabotq22_k := \frac{b1_k}{2} \qquad Cabotq12_k := \left(\frac{b1_k}{2} + \sqrt{-b2_k}\right) \qquad Cabotq21_k := \left(\frac{b1_k}{2} - \sqrt{-b2_k}\right)$$

(D-18)

$$
\begin{aligned}
&\text{Statespace} := \left|\begin{array}{l}
k \leftarrow 1 \\
y1_1 \leftarrow 0 \\
y2_1 \leftarrow 0 \\
y3_1 \leftarrow 0 \\
\text{youtput}_1 \leftarrow 0 \\
\text{while } k < \text{MaximumSample} \\
\quad \left|\begin{array}{l}
k \leftarrow k + 1 \\
y1_k \leftarrow \left(y1_{k-1} \cdot q11_k\right) + \left(\text{input}_k \cdot c1_k\right) + \left(y2_{k-1} \cdot q12_k\right) \\
y2_k \leftarrow \left(y1_{k-1} \cdot q21_k\right) + \left(\text{input}_k \cdot c2_k\right) + \left(y2_{k-1} \cdot q22_k\right) \\
\text{youtput}_k \leftarrow \left(y1_{k-1} \cdot s1_k\right) + \left(y2_{k-1} \cdot s2_k\right) + \left(\text{input}_k \cdot a0_k\right)
\end{array}\right. \\
\text{return youtput}
\end{array}\right.
\end{aligned}
$$

where

$$m1_k := -b1_k \qquad m2_k := -b2_k$$

$$k1_k := a1_k - \left(a0_k \cdot m1_k\right) \qquad k2_k := a2_k - a0_k \cdot m2_k$$

$$x1_k := \left(k2_k - m1_k \cdot \frac{k1_k}{2}\right) + \sqrt{\left[\left(k2_k\right)^2 - k1_k \cdot k2_k \cdot m1_k\right] + \left(k1_k\right)^2 \cdot m2_k}$$

$$x2_k := \left(k2_k - m1_k \cdot \frac{k1_k}{2}\right) - \sqrt{\left[\left(k2_k\right)^2 - k1_k \cdot k2_k \cdot m1_k\right] + \left(k1_k\right)^2 \cdot m2_k}$$

$$q11_k := \frac{-m1_k}{2} \qquad q22_k := q11_k \qquad c1_k := \frac{1 + k2_k}{2} \qquad c2_k := \frac{k1_k}{2}$$

$$q12_k := \frac{x1_k \cdot \left(1 + k2_k\right)}{\left(k1_k\right)^2} \qquad q21_k := \frac{x2_k}{1 + k2_k} \qquad s1_k := \frac{k1_k}{1 + k2_k} \qquad s2_k := 1$$

(D-19)

```
LadderAllpass :=
    i ← 1
    R_1 ← 0
    Q_1 ← 0
    while i < MaximumSample
        i ← i + 1
        P_i ← input_i · c2_i + R_(i-1) · -k2_(i-2)
        Q_i ← P_i · c1_i + Q_(i-1) · -k1_(i-1)
        R_i ← P_i · k1_i + Q_(i-1) · c1_i
        Y_i ← R_(i-1) · c2_i + input_i · k2_i
    return Y
```

```
LatticeAllpass :=
    i ← 1
    R_1 ← 0
    Q_1 ← 0
    while i < MaximumSample
        i ← i + 1
        P_i ← input_i + R_(i-1) · -k2_i
        Q_i ← P_i + Q_(i-1) · -k1_i
        R_i ← Q_i · k1_i + Q_(i-1)
        Y_i ← R_(i-1) + P_i · k2_i
    return Y
```

where

$$k2_i := -b2_i \qquad k1_i := \frac{-b1_i}{1 + -b2_i} \qquad c1_i := \sqrt{1 - (k1_i)^2} \qquad c2_i := \sqrt{1 - (k2_i)^2}$$

(D-20)

```
LadderMassie :=
    i ← 1
    R_1 ← 0
    Q_1 ← 0
    while i < MaximumSample
        i ← i + 1
        P_i ← input_i · c2_i + R_(i-1) · -k2_i
        Q_i ← P_i · c1_i + Q_(i-1) · -k1_i
        R_i ← P_i · k1_i + Q_(i-1) · c1_i
        Y_i ← R_(i-1) · c2_i + input_i · k2_i
        output_i ← [Y_i · (1 - MassieG) + input_i · (1 + MassieG)] · 0.5
    return output
```

where

$$MassieG := 10^{\frac{G}{20}}$$

(D-21)

```
LatticeMassie := | i ← 1
                 | R_1 ← 0
                 | Q_1 ← 0
                 | while i < MaximumSample
                 |     i ← i + 1
                 |     P_i ← input_i + R_(i-1) · -k2_i
                 |     Q_i ← P_i + Q_(i-1) · -k1_i
                 |     R_i ← Q_i · k1_i + Q_(i-1)
                 |     Y_i ← R_(i-1) + P_i · k2_i
                 |     output_i ← [Y_i · (1 - MassieG) + input_i · (1 + MassieG)] · 0.5
                 | return output
```

(D-22)

```
LadderMoorer := | i ← 1
                | R_1 ← 0
                | Q_1 ← 0
                | while i < MaximumSample
                |     i ← i + 1
                |     P_i ← input_i · c2_i + R_(i-1) · -k2_i
                |     Q_i ← P_i · c1_i + Q_(i-1) · -k1_i
                |     R_i ← P_i · k1_i + Q_(i-1) · c1_i
                |     Y_i ← R_(i-1) · c2_i + input_i · k2_i
                |     output_i ← Y_i · v2_i + R_i · v1_i + Q_i · v0_i
                | return output
```

where

$$v2_i := a2_i \qquad v1_i := \frac{1}{c2_i}\cdot\left(a1_i - a2_i \cdot b1_i\right) \qquad v0_i := \frac{1}{c1_i \cdot c2_i}\cdot\left(a0_i - c2_i \cdot k1_i \cdot v1_i - k2_i \cdot v2_i\right)$$

(D-23)

D.3 Interpolator functions

This section describes the coefficient and parameter interpolation algorithms employed to minimise signal disturbance under coefficient update. The linear interpolator algorithm is shown in (D-24). Finite wordlength arithmetic can be applied to the addition operation in this algorithm if required. The algorithm can also perform sub-sampled interpolation through the manipulation of constant 'interleave'. (D-25) shows the linear interpolator increment used in the algorithm. An alternative increment function is shown under finite wordlength constraints can also be quantised. (D-26) shows the application of the linear interpolator function between two coefficient states, sa0 and fa0.

$$\begin{aligned}
&\text{IntLinRamp}(\text{start},\text{target}) := \left|\begin{array}{l}
c_1 \leftarrow \text{start} \\
n \leftarrow 0 \\
c\delta \leftarrow \dfrac{(\text{target}-\text{start})\cdot\text{interleave}}{\text{RampSamples}} \\
i \leftarrow 1 \\
\text{while } i < \text{MaximumSample} \\
\quad \left|\begin{array}{l}
i \leftarrow i+1 \\
\delta \leftarrow 0 \quad \text{if } i \le \text{LinearStartPt} \\
\delta \leftarrow c\delta \quad \text{if } i > \text{LinearStartPt} \\
\delta \leftarrow 0 \quad \text{if } i > \text{LinearStartClampPt} \\
\delta \leftarrow -c\delta \quad \text{if } i > \text{LinearEndPt} \\
\delta \leftarrow 0 \quad \text{if } i > \text{LinearEndClampPt} \\
n \leftarrow n+1 \\
c_i \leftarrow c_{i-1}+\delta \quad \text{if } n = \text{interleave} \\
c_i \leftarrow c_{i-1} \quad \text{otherwise} \\
n \leftarrow 0 \quad \text{if } n = \text{interleave}
\end{array}\right. \\
c
\end{array}\right.
\end{aligned} \tag{D-24}$$

$$a0\delta := \frac{fa0-sa0}{\text{RampSamples}} \qquad a0\delta c := \text{fxstd}\left(\frac{fa0-sa0}{\text{RampSamples}}\right) \tag{D-25}$$

$$\text{Lina0} := \text{IntLinRamp}(sa0, fa0) \tag{D-26}$$

(D-27) shows a function implementing an exponential interpolator using fixed point single precision arithmetic emulation. The time constant is controlled through the variable, 'τcoef', (D-28) where, 'IFs' is the sampling frequency of the interpolator and 'torr' is the time constant in seconds. (D-29) shows the exponential interpolator facilitating sub-sampling frequencies through the variable interleave, note the variable 'τcoef' needs to be consider the sub-sampling frequency. (D-30) shows an example of the exponential interpolator implemented in single precision floating point arithmetic (using sign magnitude truncation). Note 'n' is the number of fractional bits in the single precision floating point mantissa format. Addsmtrc and Multsmtrc are arithmetic operations in extended precision floating point format. (D-31) shows the application of an exponential interpolator function between two coefficient states, sa0 and ta0.

```
IntExpFixed(start, target) :=
    c_1 ← start
    i ← 1
    while i < MaximumSample
        i ← i + 1
        c_i ← fxstd(c_(i-1) + fxstd(τcoef · fxext(target_i - c_(i-1))))
        c_i ← target_i  if c_i = c_(i-1)
    c
```

(D-27)

$$\tau coef := 1 - e^{-\frac{1}{IFs \cdot torr}}$$

(D-28)

```
IntExp(start, target) :=
    c_1 ← start
    n ← 0
    i ← 1
    while i < MaximumSample
        i ← i + 1
        n ← n + 1
        c_i ← c_(i-1) + τcoef · (target_i - c_(i-1))  if n = interleave
        c_i ← c_(i-1)  otherwise
        n ← 0  if n = interleave
    c
```

(D-29)

```
IntExpFloat(start, target) :=
    c_1 ← start
    i ← 1
    while i < MaximumSample
        i ← i + 1
        c_i ← flsmtrc(Addsmtrc(c_(i-1), Multsmtrc(Addsmtrc(target_i, -c_(i-1)), τcoef)), n)
        c_i ← target_i  if c_i = c_(i-1)
    c
```

(D-30)

$$IntEa0 := IntExpRamp(sa0, ta0)$$

(D-31)

(D-32) shows an example of a frequency parameter interpolation scheme. The frequency interpolation has a log law as shown. The interpolated frequency parameter is passed into a coefficient calculation process, shown in (D-33). The example coefficient calculation function used in (D-33) is the Bell filter a0 function. Note, in this example, the G and Q are constant.

$$freq_i := \begin{cases} sf & \text{if } i \le StartInterpolationPt \\ sf \cdot 10^{\log\left(\frac{ff}{sf}\right)\cdot\frac{i - StartInterpolationPt}{RampSamples}} & \text{if } i > StartInterpolationPt \\ ff & \text{if } i > StartInterpolationPt \\ ff \cdot 10^{\log\left(\frac{sf}{ff}\right)\cdot\frac{i - EndInterpolationPt}{RampSamples}} & \text{if } i > EndInterpolationPt \\ sf & \text{if } i > EndInterpolationPt \end{cases} \tag{D-32}$$

$$a0_i := Bella0\left(freq_i, sg, sq\right) \tag{D-33}$$

Appendix E Published papers

Clark R. J., Ifeachor E. C., Rogers G. M., The study of arithmetic and wordlength requirements for digital audio filtering hardware. Preprint 4100, 99th Audio Engineering Society Convention , 1995.

Clark R. J., Ifeachor E. C., Rogers G. M., Real-time equaliser coefficient realisation with minimised computational load and distortion. Preprint 4360, 101st Audio Engineering Society Convention , 1996.

Clark R. J., Ifeachor E. C., Rogers G. M., Van Eetvelt P. W. J., Techniques for generating digital equalizer coefficients. Journal of the Audio Engineering Society, vol. 48, no. 4, April 2000.