12-1-2017

Comparison of Simulation Methods of Single and Multi-Bit Continuous Time Sigma Delta Modulators

Benju Koirala
University of Nevada, Las Vegas, benzu.koirala@gmail.com

COMPARISON OF SIMULATION METHODS OF SINGLE AND MULTI-BIT CONTINUOUS TIME SIGMA DELTA MODULATORS

By

Benju Koirala

Bachelor of Engineering – Electronics and Telecommunications Engineering

Tribhuvan University

2013

A thesis submitted in partial fulfillment of

the requirements for the

Master of Science in Engineering - Electrical Engineering

Department of Electrical and Computer Engineering

Howard R. Hughes College of Engineering

The Graduate College

University of Nevada, Las Vegas

December 2017

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

November 14, 2017

This thesis prepared by

Benju Koirala

entitled

Comparison of Simulation Methods of Single and Multi-Bit Continuous Time Sigma Delta Modulators

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Engineering - Electrical Engineering
Department of Electrical and Computer Engineering

Dr. Peter Stubberud, Ph.D.
Examination Committee Chair

Dr. Sahjendra Singh, Ph.D.
Examination Committee Member

Dr. Ebrahim Saberinia, Ph.D.
Examination Committee Member

Dr. Ajoy K. Datta, Ph.D.
Graduate College Faculty Representative

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

ABSTRACT

Continuous time Sigma Delta Modulators (CT ∑ΔMs) are a type of analog to digital converter (ADC) that are used in mixed signal systems to convert analog signals into digital signals. ADCs typically require antialiasing filter; however antialiasing filters are inherent in CT ∑ΔMs, and therefore they require less circuitry and less power than other ADC architectures that require separate antialiasing filters. As a result, CT ∑ΔM ADC architectures are preferred in many mixed signal electronic applications.

Because of the mixed signal nature of CT ∑ΔMs, they can be difficult to simulate. In this thesis, various methods for simulating single-bit and multi-bit CT ∑ΔMs are developed and these simulations include the bilinear transform or trapezoidal integration, impulse invariance transform, midpoint integration, Simpson's rule, delta transform or Euler's forward integration rule and Simulink modeling. These methods are compared with respect to speed which is given by the total simulation time, accuracy which is given by the signal to noise ratio (SNR) value and the simplicity of the simulation method. The CT ∑ΔMs have been extended from first order up to fifth order with one, two and three bit quantizers. Also, the frequency domain analysis is done for all the orders of CT ∑ΔMs.

The results show that the numerical integration methods are more accurate and faster than Simulink. However, CT ∑ΔM simulations using Simulink are simpler because of the availability of the required blocks in Simulink. The overall comparison shows that the numerical integration methods can perform better than Simulink models. The frequency domain analysis proves the correctness of the use of numerical integration methods for CT ∑ΔM simulations.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincerest gratitude to my thesis advisor, Dr. Peter Stubberud for his continuous support, motivation and patience throughout my work. I would like to thank him for choosing me as his student and giving me the opportunity to work under his guidanc. His guidance and immense knowledge helped me complete this research work.

Besides my advisor, I would also like to thank my advisory committee members; Dr. Sahjendra Singh, Dr. Ebrahim Saberinia and Dr. Ajoy K. Datta for allocating their precious time in reviewing my work and providing valuable comments.

I am grateful to my husband Ashok Adhikari for all his love and support during my studies. I would like to specially thank my sister Lata Koirala, who is my guardian and inspiration. She has always encouraged me and supported me financially and emotionally during my course of studies till today. I would like to thank my mother Sushila Koirala who took a stand against my family to educate me in a nice school and make me what I am today. Also, I would like to thank my friends who helped me directly or indirectly during my thesis.

BENJU KOIRALA

University of Nevada, Las Vegas

December 2017

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

In today's rapidly growing market of portable electronics, low voltage and low power circuits are in great demand because they have a longer battery life. In any type of mixed signal electronic system, analog circuits including analog to digital converters (ADCs) typically consume the most power [1]**.** An ADC is the interface between the analog and digital electronics. In many mixed signal applications, accuracy improves the system's performance. Therefore, low power and highly accurate ADCs are fundamental requirements of many electronic systems. Sigma-delta modulators (∑ΔMs) are a type of ADC that can achieve high accuracy while consuming less power and using fewer critical analog components than other ADCs architectures such as flash ADCs, dual-slope ADCs, pipeline ADCs and successive approximation register (SAR) ADCs [2]. Sigma-delta (ΣΔ) ADCs are commonly used in modern high data-rate mobile wireless communications systems [3].

Analog to digital converters can be classified by their sampling rates as either Nyquist-rate converters or oversampling converters [7]. Nyquist-rate converters operate near the input signal's Nyquist rate which is twice the signal's maximum frequency whereas oversampling converters operate at rates much greater than the input signal's Nyquist rate. Flash ADCs, dual-slope ADCs, pipeline ADCs and successive approximation register (SAR) ADCs are examples of Nyquist-rate converters. ∑ΔM ADCs are oversampling converters. Unlike Nyquist-rate converters which are suitable for applications requiring moderate resolution conversion of wide bandwidth signals, oversampling converters typically provide high-resolution conversion of signals with moderate bandwidths. ∑ΔM ADCs can provide high resolution conversion of signals with moderate bandwidth using less power than other oversampling architectures because ∑ΔMs use fewer analog circuit components than most other architectures. As a result, ∑ΔMs are very popular in broadband telecommunication systems which use moderate signal bandwidths and require high resolution, high speed, and low power ADCs.

Accuracy and resolution of any ADC is typically measured using the ADC metrics, signal to noise ratio (SNR) and dynamic range (DR). Sigma-delta modulators (∑ΔM) can achieve high SNRs and large DRs. To achieve high SNRs and large DRs, ∑ΔMs use a feedback loop filter to shape the quantization noise and filter the input signal as it passes to the output. A ∑ΔM's loop filter is designed in such a way that it attenuates the quantizer's noise and passes the input signal to the ∑ΔM's output without attenuation in the frequency band of interest [2]. The loop filter's transfer function from the ∑ΔM's input to the ∑ΔM's output is called the signal transfer function (STF), and for a lowpass ∑ΔM, the ∑ΔM's STF is a lowpass filter. The loop filter's transfer function from the ∑ΔM's quantizer to the ∑ΔM's output is called the noise transfer function (NTF), and for a low pass ∑ΔM, the ∑ΔM's NTF is a high pass filter [7].

Depending on the circuit components used in the ∑ΔM's loop filter, ∑ΔMs can be classified as either discrete time (DT) ∑ΔMs or continuous time (CT) ∑ΔMs. DT ∑ΔMs have loop filters consisting of discrete time circuits such as switched current or switched-capacitor circuits whereas CT ∑ΔMs have loop filters consisting of continuous time circuits such as RC integrators [3]. For a DT ∑ΔM, the input signal is sampled prior to the ∑ΔM's loop filter; whereas in a CT ∑ΔM, the signal is sampled inside the modulator's loop filter. Unlike DT ∑ΔMs, CT ∑ΔMs do not use discrete time circuits and therefore do not have settling time requirements in their loop filters. As a result, CT ∑ΔMs can operate at higher frequencies than DT ∑ΔMs. Using the same technology, CT ∑ΔMs can be clocked up to an order of magnitude faster than DT ∑ΔMs without much performance penalty [5].

Another advantage that CT ∑ΔM ADCs have over DT ∑ΔM ADCs is that CT ∑ΔMs have inherent anti-aliasing filtering in their signal transfer functions which helps to reduce the number of analog circuit components in the overall system. Thus, CT ∑ΔMs can operate with less power than DT ∑ΔMs [5]. A disadvantage of CT ∑ΔMs is that they are more difficult to design and simulate than DT ∑ΔMs because of the mixed signal nature of CT ∑ΔMs which use both analog and digital circuits in their loop filters. On the other hand, DT ∑ΔMs can be accurately modeled using difference equations as they are simply made up of delays and gains [3]. There are various approaches for simulating CT ∑ΔMs such as using Simulink, SPICE modeling and solving differential equations**.** Each simulation method has a tradeoff between various measures such as speed, accuracy, and simplicity.

Depending on the number of bits that are used in a ∑ΔM's quantizer, a ∑ΔM can be classified as either a single-bit ∑ΔM or a multi-bit ∑ΔM. Single-bit CT ∑ΔMs have the advantage over multi-bit CT ∑ΔMs in that single-bit quantizers are inherently linear because they have only one quantization step. Thus, mismatches of quantization step sizes do not exist and highly linear data conversion is realizable with single-bit ∑ΔMs. On the other hand, multi-bit quantizers exhibit some nonlinearity in their transfer characteristics due to the mismatch of quantization step sizes. These nonlinearities negatively affect the performance of the multi-bit ∑ΔMs**.** Also, the added analog circuitry of a multi-bit quantizer increases the ∑ΔM's overall power consumption. A disadvantage of single-bit ∑ΔMs is that for a certain loop filter, a single bit ∑ΔM achieves less signal to noise ratio (SNR) than an equivalent multi-bit ∑ΔM [6]. Every bit added to a ∑ΔM's quantizer reduces the quantization noise by approximately 6dB. This 6 dB decrease in the quantization noise power increases the ∑ΔM's signal to noise ratio (SNR) by 6dB for every bit added to the quantizer [3].

1.1. Motivation and History

The need of continuous time sigma delta modulator converters arises from the need for current digital electronic circuits to operate with low power using low voltage processes and achieve higher signal to noise ratios (SNRs) and larger dynamic ranges (DRs) than the previous generation of circuits. Various other ADC architectures such as flash ADCs, dual-slope ADCs, pipeline ADCs and successive approximation register (SAR) ADCs have been researched and implemented over many years. Since these architectures are Nyquist-rate converters, they require an analog antialiasing filter with a sharp transition band which is difficult to obtain. CT ∑ΔMs are oversampling converters that have an anti-aliasing filter inherent in their architecture. As a result, ∑ΔMs have become popular for use in digital electronic circuits [13]. In oversampling converters like ∑ΔMs, the in-band quantization noise power is reduced by a factor of $\frac{1}{OSR}$ where OSR is the oversampling ratio. Also, ∑ΔM ADCs use fewer critical analog components and consume less power. Along with that, they are not very sensitive to circuit imperfections and do not need correction mechanisms like Nyquist-rate architectures [2]. Research shows that there are various advantages that continuous-time (CT)∑ΔMs have over discrete-time (DT) ∑ΔMs implementations. These advantages include requiring less power because of the inherent anti-aliasing filters in their STFs and because they require fewer analog components; operating at higher clock frequencies and relaxed requirements on sampling because sampling is performed inside the loop filter [13] [21]. CT ∑ΔM ADCs have been extensively used in wireless receivers to perform analog to digital conversion of signals that have bandwidths greater than 15 MHz and resolutions of 10-14 bits [14].

The sigma Delta Modulator was invented by Cutler in 1960 but it was only described in the published literature by Inosha and Yasuda in 1962 [15]. In [15], the authors report that sigma-delta modulators have excellent precision, linearity and noise rejection capability and are highly suited for implementation in integrated circuits (ICs). In his widely-cited paper [16] in 1985 on use of double integration in sigma delta modulation, Candy described how sigma delta modulation employs integration and feedback to shape quantization noise and how a modulator that employs double integration and has two-level quantization is simple to implement and tolerant of parameter variations. After this paper, several applications of sigma delta modulators appeared in audio and wide bandwidth communication applications [4].

In mid1990s, after research established various CT ∑ΔMs' benefits such as the simplicity of the required continuous-time circuits, and an inherent anti-aliasing, the research and implementation of continuous-time sigma-delta modulators became popular [4]. In recent years, much research has been done in the field of CT ∑ΔMs because of the high demand of high-speed and low-power ADCs in communications systems.

In their book on "Continuous Time Delta Sigma Modulators for High Speed Analog to Digital Conversion" [5], J. A. Cherry and W. M. Seagrove discuss the advantages of CT ∑ΔMs over DT ∑ΔMs, CT ∑ΔM practical design issues such as excess loop delay degrading the stability of CT ∑ΔMs, the theoretical treatment of clock jitter and quantizer metastability as well as various compensation approaches for feedback loop delay. The authors state that CT ∑ΔMs have faster clocking, better virtual grounds and inherent antialiasing filters, and thus, CT ∑ΔMs have fewer circuit requirements and longer battery life. The authors design CT ∑ΔMs by converting DT ∑ΔMs to CT ∑ΔMs using the impulse-invariant transformation and then use a SPICE based procedure to determine DAC feedback currents that are used to implement the CT ∑ΔM's NTF. The authors use a root locus method to show how excess loop delay degrades the feedback loop stability and present a method that minimizes excess loop delay problems by using RZ DAC pulses instead of

NRZ DAC pulses, by using additional feedback loops and by tuning the DAC feedback levels. To avoid quantizer metastability problems at high speed, the authors use a fully integrated modulator with a VCO operating at around GHz speed and another half latch in the quantizer for additional signal regeneration.

In their book on "Continuous Time Sigma Delta Modulation For Analog to Digital Conversion in Radio Receivers" [17], Lucien Breems and Johan H. Huijsing describe the design and implementation of CT ∑ΔMs for signal conversion in radio receivers. Their objective was to design a highly linear modulator with a large dynamic range and good image rejection capabilities both of which are important requirements for a radio receiver. They use single-bit CT ∑ΔMs for analog to digital conversion as it has benefits like high linearity, and low power capability which is very important for battery-powered receivers. They also describe various design issues of CT ∑ΔMs such as quantization noise, clock jitter, intersymbol interference (ISI), DC tones, and aliasing. They use inverse-Chebyshev and Butterworth filter characteristics for designing the NTFs of higher-order ∑ΔMs. The book also describes the design of a quadrature ∑ΔMs with a data-dependent dynamic element matching circuit. The book emphasizes how a CT ∑ΔM can be combined with a mixer in radio receivers for intermediate frequency to baseband analog to digital conversion with less power and higher performance than other ADC architectures.

In their book "Continuous Time Sigma Delta Analog to Digital Conversion" [4], M. Ortmanns and F. Gerfers discuss CT ∑ΔMs non-idealities, their classification and modeling. They also present a low power design strategy that is based on a Figure-of-Merit which uses optimal ∑ΔM topology for designing CT ∑ΔMs. Design examples of low pass single loop, ΣΔ modulators, with single-bit and multi-bit quantizaters are presented. The authors conclude that multi-bit modulators offer improved stability and reduce the in-band quantization noise by a factor of $(2^B -1)^2$ in comparison to single-bit modulators where B is the number of bits used in the quantizer. They also discuss how single-loop architectures are highly unstable and only through proper selection of the scaling coefficients, the architecture can be made stable. As an alternative to single loop architecture, they discuss cascaded-loop architectures which consist of connections of low-order modulators. Therefore, these topologies require almost no scaling, and can achieve good stability.

In [3] and [18] K. Kang and P. Stubberud use the delta transformation to model CT ∑ΔMs. The delta transform is based on Euler's forward integration method. The authors compared simulation methods such as MATLAB/Simulink, delta transform, CT/DT transformation, SPICE modeling and solving differential equations for modeling second, third, fourth and fifth single-bit CT ∑ΔMs. The comparison is based on SQNR accuracy, speed and simplicity of the simulation method. Also, the authors discuss overloading associated with the quantizer and have given conditions on how overloading can be prevented. An analytical root locus method is discussed for determining the stability criteria for CT ∑ΔMs having exponential functions in the characteristics equations. The analytical root locus method describes the range of the quantizer gains where the modulator can function without being unstable. The gains values can also help to determine the internal and input signal powers to prevent the CT ∑ΔMs from being unstable.

1.2. Intention of this work

Most research work to date has mainly focused on uses of CT ∑ΔMs in various disciplines from communications to biomedical applications, advantages of CT ∑ΔMs over DT ∑ΔMs, using a single transformation method for converting z-domain to s-domain transfer functions, only one type of simulation environment and minimizing nonidealities associated with the modulators. No research has been published on how types of numerical integration methods and types of simulation methods can be used to predict the performance of CT ∑ΔMs. Only [3] and [18] present simulations using various methods such as such as MATLAB/Simulink, delta transform, CT/DT transformation, SPICE modeling and solving differential equations. These simulation methods are also compared in [3] and [18]. However, the authors only simulated single-bit CT ∑ΔMs and did not simulate multi-bit CT ∑ΔMs. They also did not include the first order CT ∑ΔMs. Also, they did not include simulation methods like Simpsons rule, bilinear or trapezoidal integration and midpoint integration. They have not done a comparison of various numerical integration methods. Also, they have not done the frequency domain analysis of the simulation methods.

In this thesis, methods for simulating single-bit and multi-bit CT ∑ΔMs are developed. These methods are compared with respect to simulated signal to noise ratio, dynamic range, total elapsed time, frequency response and performance which includes accuracy, simplicity, and speed of the simulation method. The various methods of simulations include the bilinear transform or trapezoidal integration, impulse invariance transform, midpoint integration, Simpson's rule, delta transform or Euler's forward integration rule, Simulink and SPICE modeling. The CT ∑ΔMs have been extended from first order up to fifth order with one, two and three bit quantizers. Also, the frequency domain analysis is done for all the orders of CT ∑ΔMs.

1.3. Organization of the thesis

In this thesis, Chapter 2 reviews various components and metrics of analog to digital converters; operating principles of ∑ΔMs; several types of ∑ΔMs; various numerical integration methods such as the bilinear transform or trapezoidal integration, impulse invariance transform, midpoint integration, Simpson's rule and delta transform or Euler's forward integration rule for modeling CT ∑ΔMs and the frequency response comparison of the numerical integration methods.

Chapter 3 is about literature review on various numerical integration methods and simulation methods that are used for simulating CT ∑ΔMs.

In Chapter 4, the first order, second order, third order, fourth order and fifth order CT ∑ΔMs are represented in block diagrams and their STFs and NTFs are determined. Also, the STFs and NTFs are designed using Chebyshev Type 2 filter and coefficients are calculated by comparing the designed STFs and NTFs with the determined STFs and NTFs from the block diagrams.

Chapter 5 describes the simulation of the first order, second order, third order, fourth order and fifth order CT ∑ΔMs with one, two and three bit quantizers using the numerical integration methods and Simulink. In this chapter, comparison is done on all the simulation methods which is based on total computation time, SNR, dynamic range and simplicity of the simulation method. Also, the frequency domain analysis is done for these CT ∑ΔMs in order to prove the correctness of the proposed numerical integration *s*-domain to *z*-domain transformation formulas.

Chapter 6 summarizes all the work done in this thesis along with the future work.

Chapter 2

BACKGROUND

Sigma-delta modulation is a method that has been applied to both analog to digital converters (ADCs) and digital to analog converters (DACs). Both Sigma (∑) and delta (Δ) are Greek letters and with respect to sigma delta modulation, the ∑ represents accumulation or integration operations and the Δ represents the difference operation. Thus, sigma-delta (∑Δ) modulation usually refers to the operation of accumulating the differences of two signals in a feedback loop.

2.1 Analog to Digital Conversion:

In a conventional ADC operation, an analog signal is sampled at a certain sampling frequency and subsequently quantized into a digital signal. The general ADC process can be modeled by three subsystems, an anti-aliasing filter (AAF), a sampler, and a quantizer [3] as shown in Fig. 2.1. If the input to an ADC is an analog or continuous time signal $x(t)$, then the output, $y(n)$ is a discrete time signal with an amplitude that is quantized. After these three basic components of an ADC operation, an encoder converts $y(n)$ into the desired number representation such as sign+ magnitude or 2's compliment.

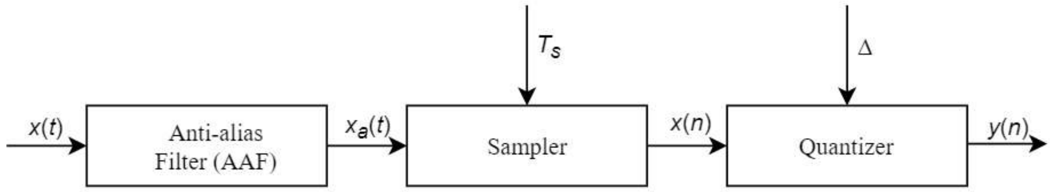


Figure 2.1: Components of an analog to digital converter (ADC)

2.1.1 Anti-alias filter (AAF):

The first component of an ADC operation is an anti-aliasing filter (AAF) which bandlimits the analog input signal, $x(t)$. Anti-alias filters remove frequency components above half of the sampling frequency which can fold into the signal's band of interest during the subsequent sampling process [3]. Ideally, the AAF is an lowpass filter (LPF) with a cut-off frequency of f_c which equals half the input signal's sampling frequency.

2.1.2 Sampler:

The second component is a sampler, which converts the filtered continuous-time-signal $x_a(t)$ into a discrete-time sequence $x(n)$ by extracting the amplitudes of the signal $x_a(t)$ at integer multiples of the sampling period, T_s, such that

$$x(n) = x_a(nT_s). \tag{2.1}$$

The filtered continuous-time signal, $x_a(t)$, must be sampled at a minimum sampling rate f_s which is twice the signal's maximum or highest frequency i.e. $f_s \geq 2f_m$. Sampling at or above f_s prevents signal loss due to aliasing which is an effect that causes higher frequency components to become indistinguishable from lower frequency components when sampled. If $f_s \geq 2f_m$, then the sampling period T_s will be

$$T_s \leq \frac{1}{2f_m} \tag{2.3}$$

and the input signal can be recovered from its samples $x_a(nT_s)$. The highest or maximum signal frequency, f_m, is the Nyquist frequency and twice the Nyquist frequency, $2f_m$, is the Nyquist rate which is also the minimum sampling rate required to prevent the signal from aliasing when sampled.

Based on sampling frequency, ADCs can be broadly classified as Nyquist-rate converters or oversampling converters.

2.1.2.1 Nyquist-rate Converters:

Nyquist-rate converters operate at or near the Nyquist rate, $2f_m$, which is twice the signal's maximum frequency, f_m. In practice, Nyquist-rate converters are difficult to design because they have zero transition band to cut off unwanted high frequency signals for their filter. Also, because Nyquist-rate converters require various operations such as amplification, comparison, etc. that must be performed with high accuracy and precision, the intrinsic precision of the integrated circuits (ICs) components can limit a Nyquist rate ADC to 12-bits of accuracy [13]. To achieve more than 12 bits of accuracy, Nyquist-rate converters mostly depend on correction techniques such as DEM and self-calibration. As a result, Nyquist-rate converters are suitable for applications requiring moderate resolution conversion of wide bandwidth signals.

Various Nyquist rate ADC architectures include flash, dual-slope, pipeline, and successive approximation register (SAR) converters.

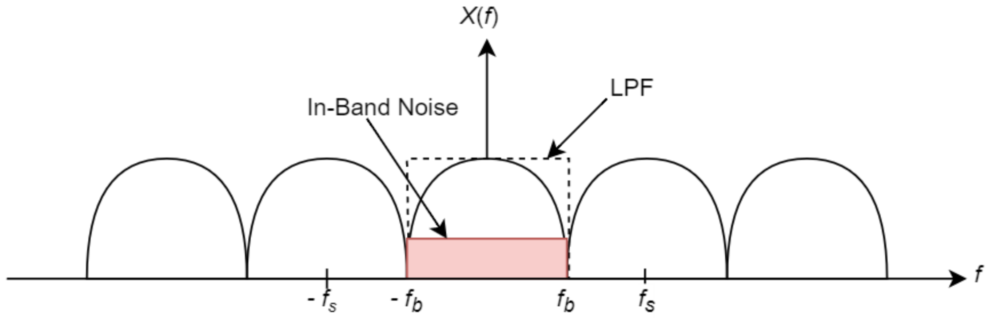
2.1.2.2 Oversampling Converters:

Oversampling converters operate at rates much greater than the signal's Nyquist rate. An ADC's oversampling ratio (OSR) is defined as

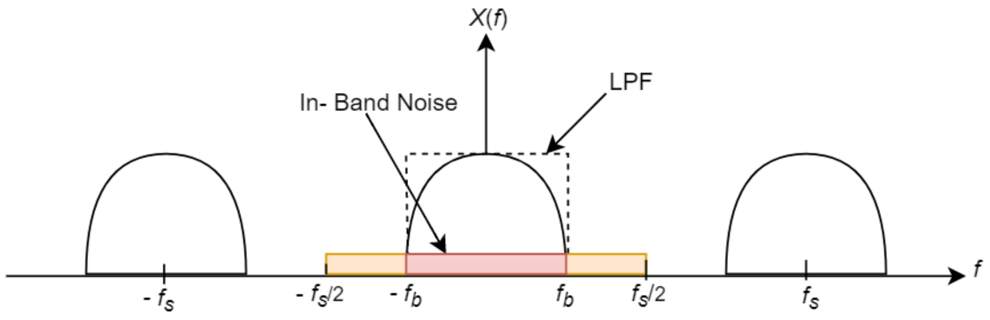
$$OSR = \frac{f_s}{2f_m} \tag{2.4}$$

where f_s is the sampling frequency and f_m is the maximum signal frequency. With oversampling, anti-aliasing filters do not have a zero-transition band width but instead have a gentle roll off in their transition band which will make them less costly and easier to design. Thus, oversampling converters can require less power, and use less chip area. Also, the resolution of a Nyquist-rate ADC can be increased by increasing the oversampling rate of the converter. Thus, oversampling converters can achieve higher resolution than the resolution obtained by Nyquist-rate converters. Sigma-delta modulators are popular oversampling ADCs [4].

To illustrate how oversampling can be used to increase a converter's resolution, consider a B-bit Nyquist-rate converter and a B-bit oversampling converter. A B-bit quantizer's noise power is the same for both the Nyquist-rate and oversampling converters. However, for an oversampling converter the out of band noise from the quantizer can be filtered out. The output of an oversampling ADC can be lowpass filtered from - $\frac{f_s}{2}$ to $\frac{f_s}{2}$. This implies that out of the total quantization noise, it keeps only $\frac{1}{OSR}$ of the noise while $\frac{OSR-1}{OSR}$ out of band noise from the quantizer can be filtered out from the signal's frequency band of interest.



(a)



(b)

Figure 2.2: Spectral effect of oversampling. (a) Total quantization noise and in-band quantization noise for OSR = 1; (b) Total quantization noise and in-band quantization noise for OSR = 4

Fig.2.2 illustrates how oversampling converters reduce the quantization noise. As shown in Fig. 2.2, oversampling and filtering the total quantization noise reduces the in-band quantization noise power by a factor of $\frac{1}{OSR}$. Because the quantization noise power of an oversampled ADC is inversely proportional to the ADC's OSR, an oversampling ADC's signal to noise ratio (SNR) can be increased by increasing its OSR. The only drawback of oversampling ADCs is that oversampling increases the quantizer's performance requirements. Nevertheless, oversampling converters can achieve higher data rates, higher speeds and higher resolutions than Nyquist-rate converters [6]. In many situations, oversampling ∑ΔM ADCs can obtain higher resolutions than Nyquist-rate converters without the need of component matching. Oversampling converters are normally used for moderate or narrow bandwidth operations such as audio and instrumentation.

2.1.3 Quantizer:

The third component of an ADC operation is the quantizer. Quantization is the process of converting continuous amplitudes to discrete amplitudes. A quantizer transforms a discrete time, continuous amplitude signal into digital signal which has a finite number of amplitude levels. In Fig. 2.1, the quantizer block is quantizing the sequence $x(n)$ into a B-bit number where B is the number of bits used by the quantizer. The quantizer maps the continuous amplitude of $x(n)$ into a discrete set of amplitudes and its operation can be represented mathematically by the transformation

$$y(n) = Q[x(n)] \tag{2.2}$$

where $x(n)$ is a discrete sampled signal, $y(n)$ is a B-bit digital signal and Q is the non-linear transformation representing the quantization operation. Quantization is a noninvertible process because an infinite number of continuous input amplitude values are converted into a finite number of discrete output amplitudes, and hence even the ideal quantization process inherently introduces quantization errors into the output signal. Because a quantizer is a non-linear device, it introduces nonlinearities into the output signal.

Quantizers can have either uniformly and non-uniformly spaced quantization levels. If all the levels of a quantizer's output are equally spaced in a quantizer, it is a uniform quantizer. For a uniform quantizer, the quantization process is defined by the number, B, of bits and the quantization interval, Δ, where Δ is often referred to as the quantization step-size. For a B bit quantizer, the number of equally spaced quantization levels, L, is

$$L = 2^B. \tag{2.5}$$

If quantizer that has 2^B discrete amplitudes, the quantizer is said to have B-bits of resolution. The range, R, of the quantizer is

$$R = (2^B - 1)\,\Delta. \tag{2.6}$$

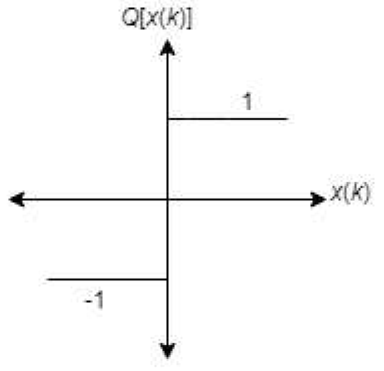
Therefore, if a B bit quantized input signal, $x(k)$ is bounded such that $Xmin \le x(k) \le Xmax$, the quantizer's range, $Xmax - Xmin$,, can be covered by a uniform step size, Δ, of

$$\Delta = \frac{Xmax-Xmin}{2^B-1}. \tag{2.7}$$

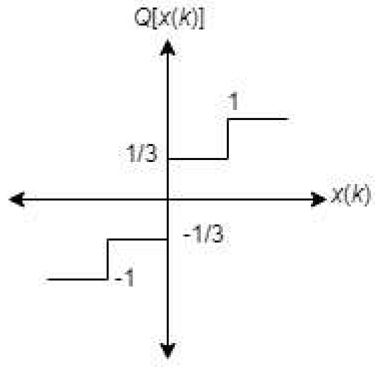
For example, if a 2-bit quantized input signal, $x(k)$ is bounded such that

$$-1 \le x(k) \le 1,$$

then the range R, of quantizer is 2 because $Xmax - Xmin = 2$. The number of quantization level is $2^B = 2^2 = 4$, and therefore, the quantization step size is, $\Delta = \frac{2}{2^2-1} = \frac{2}{3}$. Similarly, for a single-bit quantized signal $x(k)$ that has the same bounds, the range is, $R = 2$. The number of quantization levels is $2^B = 2^1 = 2$, and therefore, the quantization step size is $\Delta = \frac{2}{1-1} = 2$. These examples are illustrated in Fig. 2.3.



(a)



(b)

Figure 2.3: (a) Single-Bit Quantization; (b) 2-Bit Quantization

Generally, a quantizer with higher number of bits *B* and lower Δ will have higher resolution. Digital signals with higher resolution has less quantization noise than a digital signal with lower resolution.

Because the quantization process in non-linear, it is often modeled linearly as shown in Fig. 2.4 to simplify its analysis.

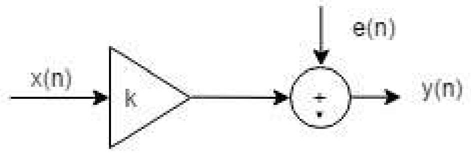


Figure 2.4 Equivalent linear model of a quantizer

In Fig. 2.4, the quantizer is modeled as a linear gain *k* with an additive random error signal $e(n)$ so that the quantizer's output can be written as

$$y(n) = Q[x(n)] = Kx(n) + e(n) \tag{2.8}$$

where $x(n)$ is the quantizer's input and *K* is the quantizer's gain. This model assumes that

a) The error sequence, $e(n)$ is a stationary random process.
b) The probability density of the error sequence is uniform over the range of values of the quantization error.

c) The error sequence is uncorrelated with the input sequence.

d) The error sequence is a white noise process; i.e. it is a sequence of uncorrelated random variables.

Equation (2.8) implies that the quantization error can be written as $e(n) = Q[x(n)] - Kx(n)$. This quantization error depends on the quantization method (truncation or rounding) and the number of equally spaced quantization levels. Quantization in fixed-point architectures is almost always performed by rounding instead of truncation, and rounding errors have a range of $[-\Delta/2, \Delta/2]$. As mentioned above, the error sequence $e(n)$ can be modeled as a uniformly distributed random process over the errors range which implies that $e(n)$ is uniformly distributed over, $[-\Delta/2, \Delta/2]$. Therefore, the amplitude of the quantization noise's probability density function, *P*(*e*), has an amplitude of $\frac{1}{\Delta}$ for $-\frac{\Delta}{2} \leq e(n) \leq \frac{\Delta}{2}$ as shown in Fig 2.5.

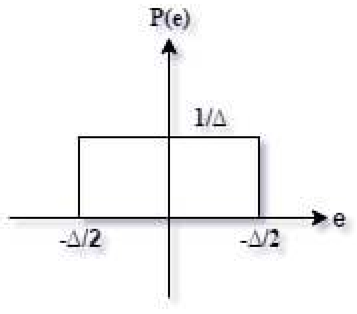


Figure 2.5 Probability density function of error sequence $e(n)$

The expected value, E, or mean of the error $e(n)$ is

$$E[e(n)] = m_{e(n)} = \int_{-\Delta/2}^{+\Delta/2} \frac{1}{\Delta}\, de(n) = 0.$$

and the quantization error power, P_e, is

$$\begin{aligned} P_e &= E(e^2) \\ &= \sigma_{e(n)}^2 - m_{e(n)}^2 \end{aligned}$$

$$= \sigma^2_{e(n)}$$

$$= \int_{-\Delta/2}^{+\Delta/2} e^2 \frac{1}{\Delta} de$$

$$= \frac{1}{\Delta}(\frac{\Delta^3}{24} + \frac{\Delta^3}{24})$$

$$= \frac{\Delta^2}{12} \tag{2.9}$$

where $\sigma^2_{e(n)}$ is the variance of $e(n)$. If the quantizer rounds to $B + 1$ bits and if the range of Δ is from $-1 + \frac{\Delta}{2} \leq \Delta \leq 1 - \frac{\Delta}{2}$ then $\Delta = 2^{-B}$ and equation (2.9) can be written as

$$\sigma^2_{e(n)} = \frac{2^{-2B}}{12}. \tag{2.10}$$

2.2 Performance Metrics:

The performance of any ADC is measured by metrics such as signal to noise ratio (SNR) and dynamic range (DR) which compare the output signal power with the output noise power.

2.2.1 Signal to Noise Ratio (SNR):

The SNR of an ADC is the ratio of the output signal power to the output noise power, i.e.

$$SNR = \frac{P_s}{P_e} \tag{2.11}$$

where P_s is the ADC's output signal power and P_e is the ADC's output quantization noise power. In decibels (dB),

$$SNR(\text{dB}) = 10\log_{10}\left(\frac{P_y}{P_e}\right) \tag{2.12}$$

If the ADC's output and quantization noise both have zero means, then $P_s = \sigma_s^2$ and $P_e = \sigma_e^2$. Assuming $Xmax - Xmin = 2 - \Delta$, then $\sigma^2_{e(n)} = \frac{2^{-2B}}{12}$, and

$$SNR\ (\text{dB}) = 10\log_{10}\left(\frac{\sigma_s^2}{\sigma_e^2}\right)$$

$$= 10\log_{10}\left(\frac{12\sigma_s^2}{2^{-2B}}\right)$$

$$= 20B\log_{10}(2) + 10\log_{10}(12) + 10\log_{10}(\sigma_s^2)$$

$$= 6.02B + 10.8 + 10\log_{10}(\sigma_s^2) \qquad (2.13)$$

If the input is a full-scale sinewave, then

$$\sigma_s^2 = \frac{1}{2}$$

and

$$SNR(\text{dB}) = 6.02B + 7.8\text{dB}. \qquad (2.14)$$

For the total number of bits, B', where $B' = B + 1$,

$$SNR\ (\text{dB}) = 6.02B' + 1.78\text{dB}. \qquad (2.15)$$

The above equation can be used to calculate the ADC's effective number of bits (ENOB) which is defined as

$$ENOB = \frac{SNR(dB) - 1.76}{6.02}. \qquad (2.16)$$

ENOB determines the resolution of an ADC. Equation (2.16) implies that an ADC with 6dB more SNR has one additional bit of ENOB [3].

The SNR of an oversampled ADC can be calculated as

$$SNR\ (dB)_{oversampled} = 6.02B' + 1.78\text{dB} + 10\log_{10}(OSR). \qquad (2.17)$$

Equation (2.17) shows that doubling an ADC's OSR will increase its SNR by 3dB.

2.2.2 Dynamic Range (DR)

Dynamic Range (DR) is the ratio of the power of the maximum detectable input signal that can be applied to an ADC without significantly degrading its performance to the power of the minimum detectable signal. DR is typically expressed in *decibels* (dBs). The smallest detectable signal can be determined by the power spectral density of the ADC's noise floor. For an ADC with a constant noise spectrum such as white noise, DR is equivalent to SNR. But, when the noise floor does not have same values and has peaks, the ADC's dynamic range is less than its SNR. In the non-uniform power spectral density noise case, the smallest detectable signal is determined where the noise floor's power spectral density is largest.

2.3 Operating Principles of Sigma Delta Modulators (∑ΔMs):

∑ΔMs achieve high resolution signal conversion by using a loop filter and a clocked quantizer. Fig. 2.6 shows the three main components of a ∑ΔM. The components are:

a) A loop filter

b) A Clocked Quantizer, or ADC

c) A Feedback digital to analog converter (DAC)

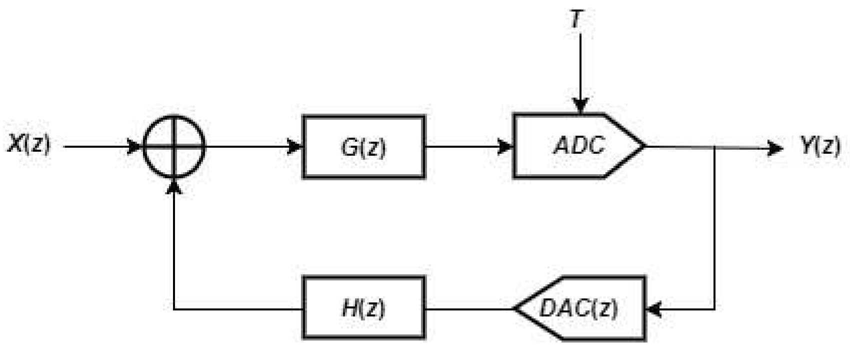


Figure 2.6: General discrete-time ∑ΔM

Quantizers and ADCs are non-linear devices which make the behavior of the modulator difficult to analyze [5]. However, the analysis can be simplified by replacing the quantizer with the linear additive noise model in Fig. 2.4. Using this quantizer model, the ∑ΔM can be represented by the linear model shown in Fig. 2.7.

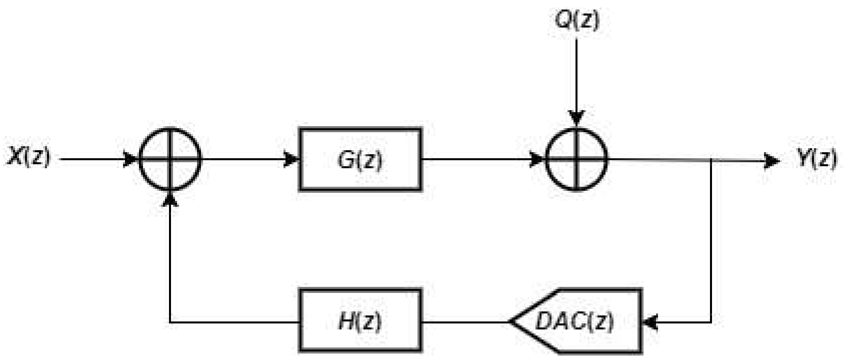


Figure 2.7: A discrete-time ∑ΔM linear model

Using the linear model in Fig. 2.7, the signal transfer function (STF) and noise transfer function (NTF) of the discrete-time ∑ΔM can be written as

$$STF(z) = \frac{Y(z)}{X(z)} = \frac{G(z)}{1+H(z)G(z)DAC(z)} \tag{2.18}$$

and
$$NTF(z) = \frac{Y(z)}{Q(z)} = \frac{1}{1+H(z)G(z)DAC(z)}, \tag{2.19}$$

respectively. The modulator's output, $Y(z)$, can now be written as

$$Y(z) = NTF(z)Q(z) + STF(z)X(z)$$

$$Y(z) = \frac{1}{1+H(z)G(z)DAC(z)}Q(z) + \frac{G(z)}{1+H(z)G(z)DAC(z)}X(z) \tag{2.20}$$

Similarly, Fig. 2.8 shows the block diagram of a continuous-time ∑ΔM.

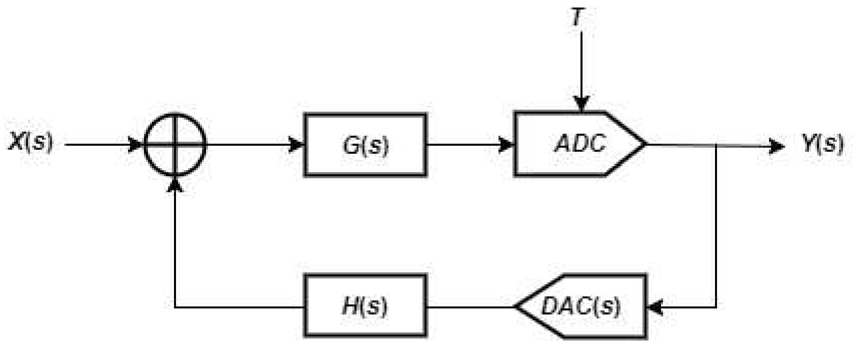


Figure 2.8: General continuous-time ∑ΔM

The continuous-time ∑ΔM in Fig. 2.8 can be modeled by the linear model shown in Fig. 2.9.

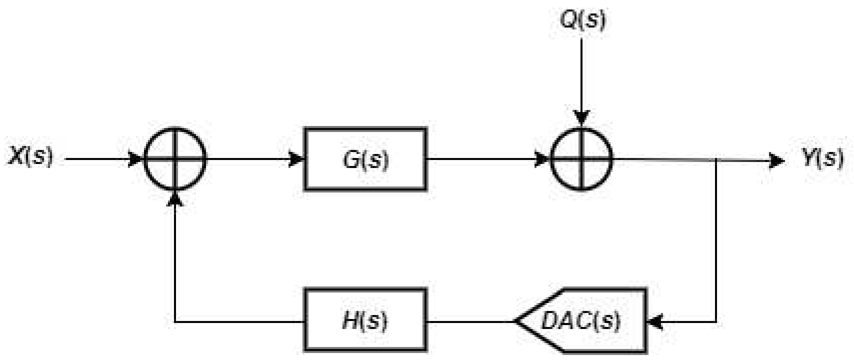


Figure 2.9: A continuous-time ∑ΔM linear model

Using the linear model in Fig. 2.9, the signal transfer function (STF) and noise transfer function (NTF) of the continuous-time ∑ΔM can be written as

$$STF(s) = \frac{Y(s)}{X(s)} = \frac{G(s)}{1+H(s)G(s)DAC(s)} \tag{2.21}$$

and
$$NTF(s) = \frac{Y(s)}{Q(s)} = \frac{1}{1+H(s)G(s)DAC(s)} , \tag{2.22}$$

respectively and modulator's output, $Y(s)$, can now be written as

$$Y(s) = NTF(s)Q(s) + STF(s)X(s)$$

$$Y(s) = \frac{1}{1+G(s)H(s)DAC(s)}Q(s) + \frac{G(s)}{1+G(s)H(s)DAC(s)}X(s) \tag{2.23}$$

For both discrete-time and continuous-time ∑ΔMs, a ∑ΔM's loop filter is designed in such a way that it attenuates the quantizer's noise in the required frequency band of interest and passes the input signal to the ∑ΔM's output without attenuation [2]. The STF is designed in such a way that the loop filter's gain is approximately unity in the passband. For a lowpass ∑ΔM, the ∑ΔM's STF is a lowpass filter and the NTF is a high pass filter so that it can suppress the quantization noise in the ∑ΔM's STF's passband [7].

2.4 Classification of Sigma Delta Modulators (∑ΔMs):

∑ΔMs are typically classified based on number of bits in the quantizer, number of quantizers, order of loop filter, the STF and NTF characteristics and the type of circuit components used in the loop filter circuitry.

2.4.1 Number of bits in a quantizer:

Depending on the number of bits that are used in a ∑ΔM's quantizer, ∑ΔMs can be classified as either single-bit ∑ΔMs or multi-bit ∑ΔMs. Single-bit CT ∑ΔMs are intrinsically linear because their quantizers have only two levels of quantization and thus one quantization step-size. Thus, mismatches of quantization step sizes do not exist and highly linear data conversion is realizable with single-bit ∑ΔMs. Multi-bit ∑ΔMs have multiple quantization levels, and thus they have mismatched quantization step sizes. As a result, they exhibit nonlinearities in their transfer characteristics which can negatively influence the performance of the ∑ΔM**.** Also, the additional analog circuitry required for multi-bit ∑ΔMs increases the design complexity and the overall cost of the design. An advantage of ∑ΔMs with multi-bit quantizers is that they generate approximately 6dB less quantization noise for every additional bit, and therefore, the signal to noise ratio (SNR) of multi-bit ∑ΔMs increases 6dB for every bit added to the quantizer. In this thesis, simulations are done for both single-bit and multi-bit CT ∑ΔMs.

2.4.2 Number of quantizers employed:

∑ΔMs that have only one quantizer are called single-loop ∑ΔMs, whereas ∑ΔMs that have more than one quantizer are often termed cascaded-loop ∑ΔMs. Cascaded topologies are relatively more stable and can achieve more performance than single loop ∑ΔMs but cascaded-loop ∑ΔMs require tighter constraints on circuit specifications and mismatch than single-loop ∑ΔMs [3].

2.4.3 Order of the loop filter:

∑ΔMs can be classified by the order of their loop filters. Orders of loop filters range from first order to higher order. As the order of the modulator increases, the quantization noise can be suppressed more over the frequencies of interest and a significant improvement in the ∑ΔM's performance can be achieved. However, modulators with higher order loop filters are less stable, have increased design complexity, increased costs and increased power consumption.

2.4.4 STF and NTF characteristics:

Depending on the characteristics of the ∑ΔM's STF and NTF characteristics, ∑ΔMs can be classified as either a lowpass (LP) ∑ΔMs or bandpass (BP) ∑ΔMs. Lowpass ∑ΔMs sample signals of interest from DC to a specific frequency. Therefore, low pass ∑ΔMs have NTFs with highpass shapes and STFs with lowpass shapes. On the other hand, bandpass ∑ΔMs sample signals from one specific frequency to another frequency and therefore they have NTFs with bandstop shapes and STFs with bandpass shapes.

2.4.5 Loop filter Circuitry:

Based on the circuit components used in the loop filter, ∑ΔMs can be classified as either discrete time (DT) ∑ΔMs or continuous time (CT) ∑ΔMs. DT ∑ΔMs use discrete time circuits such as switched current or switched capacitor circuits in their loop filters whereas CT ∑ΔMs use continuous time circuits such as RC or G_mC integrators in their loop filters. In (DT) ∑ΔMs, sampling is done outside of the loop filter whereas in (CT) ∑ΔMs, sampling is done inside the loop filter.

The classification of ∑ΔMs based on number of bits in the quantizer, number of quantizers, order of loop filter, the STF and NTF characteristics and the type of circuit components used in the loop filter circuitry have been summarized in Table 2.1.

Criteria	Classification
The number of bits in a quantizer	• Single-bit ∑ΔM • Multi-bit ∑ΔM
The number of quantizers employed	• Single-loop ∑ΔM • Cascaded ∑ΔM
The order of loop filter	• First-order ∑ΔM • Higher-order ∑ΔM
Signal Transfer Function (STF) Characteristic	• Lowpass ∑ΔM • Bandpass ∑ΔM
Loop filter circuitry	• Discrete time ∑ΔM • Continuous-time ∑ΔM

Table 2.1: Classification of ∑ΔMs

2.5 Discrete models of CT ∑ΔM:

Because CT ∑ΔMs are mixed signals systems, discrete models of the analog circuitry are required to simulate modulators digitally. Various transformation techniques can be used to model and design the modulators. Continuous time ∑ΔMs are often modeled in Laplace transform's *s* domain and discrete time ∑ΔMs are often modeled in *z*-transform's *z* domain. To simulate a CT ∑ΔM, the Laplace transform's *s*-domain variable needs to be converted to the *z*-transform's *z*-domain variable. If *H*(*s*) is the transfer function of a continuous-time filter, an equivalent digital transfer function *H*(*z*) can be obtained simply by replacing *s* by some function

$$s = f(z) \tag{2.27}$$

in *H*(*s*). In this case, the equivalent discrete transfer function *H*(*z*) would be

$$H(z) = H(s)|_{s=f(z)} = H[f(z)]. \tag{2.28}$$

If *H*(*s*) and *f*(*z*) are rational functions of *s* and *z,* respectively, then *H*(*z*) is a rational function of *z*.

For a causal analog system to be stable, the poles of *H*(*s*) must be in the left-half of the *s*-plane and for a causal digital system to be stable, the poles of *H*(*z*) (stable) must be inside the *z*-plane's unit circle. When converting a continuous time transfer function, *H*(*s*), to an equivalent discrete time transfer function, *H*(*z*), the mapping *f*(*z*) must transform the left-half of the *s*-plane inside the *z*-plane's unit circle to preserve stability in *H*(*z*) [12].

Numerical integration methods can transform from *s*-domain transfer functions to *z*-domain transfer functions. The numerical integration methods used in this thesis for modeling CT ∑ΔMs in the discrete time domain are the bilinear transform, or trapezoidal integration; impulse invariance transform; midpoint integration; Simpson's rule and the delta transform, or Euler's forward integration.

2.5.1 Impulse Invariance Transformation:

The impulse invariance transformation method maps the transfer function in *s*-domain into *z*-domain transfer functions so that both models have similar impulse responses. The impulse invariance transform generates a discrete model by sampling the impulse response of the analog system. Therefore, if an analog system has the continuous-time impulse response $h_a(t)$, and is sampled at a period of T, then the impulse invariance method selects the discrete time impulse response $h(n)$ as

$$h(n) = h_a(nT). \tag{2.29}$$

The sampling process of $h_a(t)$ can be modelled by the system in Fig. 2.9.

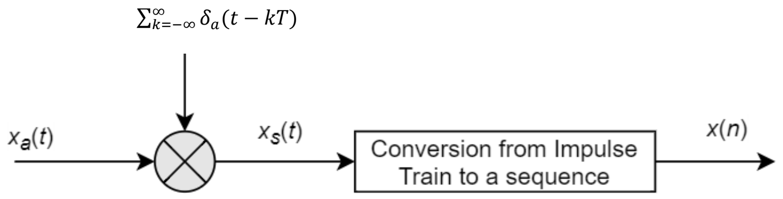

$\sum_{k=-\infty}^{\infty} \delta_a(t-kT)$
$x_a(t)$
$x_s(t)$
Conversion from Impulse Train to a sequence
$x(n)$

Figure 2.10: Model of the sampling process

As shown in Fig. 2.10,

$$x_s(t) = x_a(t) \sum_{k=-\infty}^{\infty} \delta_a(t-kT) = \sum_{k=-\infty}^{\infty} x_a(kt)\, \delta_a(t-kT) \tag{2.30}$$

where $\delta_a(t)$ is the Dirac Delta function. The Laplace transform of $X(s)$ is

$$X(s) = \int_{-\infty}^{\infty} x_s(t) e^{-st} dt. \tag{2.31}$$

Substituting (2.30) into (2.31),

$$\begin{aligned} X(s) &= \int_{-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x_a(kt)\, \delta_a(t-kT) e^{-st} dt \\ &= \sum_{k=-\infty}^{\infty} x_a(kt) \int_{-\infty}^{\infty} \delta_a(t-kT) e^{-st} dt \\ &= \sum_{k=-\infty}^{\infty} x_a(kt)\, e^{-skT} \\ &= X(z)\big|_{z = e^{sT}} \end{aligned} \tag{2.32}$$

which implies

$$H(e^{jw}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} H_a(j\frac{w}{T} - j\frac{2\pi k}{T}) \tag{2.33}$$

where $H(e^{jw})$ is the Fourier transform of $h(n)$ and $H_a(jw)$ is the Fourier transform of $h_a(t)$. Equation (2.33) shows that $H_a(j\Omega) = 0$ for $\Omega \geq \frac{\pi}{T}$ to prevent aliasing.

To apply the impulse invariance transformation, the *s*-domain transfer function $H(s)$ is expanded into partial fractions. The pole of each partial fraction is transformed to a digital pole in the *z*-domain. The transfer function $H(z)$ is found by combining the partial fractions using the *z*-domain poles. For example, using partial fraction expansion, an analog transfer function can be written as

$$H(s) = \sum \frac{b_k}{(s-a_k)} \tag{2.34}$$

Using impulse invariance transformation, the equivalent discrete transfer function, $H(z)$, can now be written as

$$H(z) = \sum \frac{b_k}{(1- e^{-a_k T} z^{-1})} \tag{2.35}$$

The impulse invariance transformation method maps the transfer functions in *s*-domain into *z*-domain transfer functions so that both models have similar impulse responses. The transformation $z = e^{sT}$ maps the analog frequencies $\frac{-\pi}{T} \leq \Omega \leq \frac{\pi}{T}$ into $\pi \leq w \leq \pi$. It also maps $\frac{-\pi}{T} \leq \Omega \leq \frac{3\pi}{T}$ into $\pi \leq w \leq \pi$. This causes aliasing. One advantage of impulse invariance mapping is that it preserves stability of the system. A disadvantage is that because the transform samples in the time domain aliasing can occur in the frequency domain. In Fig. 2.11, only the left-half of the *s*-plane maps inside the *z*-plane's unit circle. The right-half of *s*-plane maps outside of the *z*-plane's unit circle. The impulse invariance transformation process can be shown in Fig. 2.11.

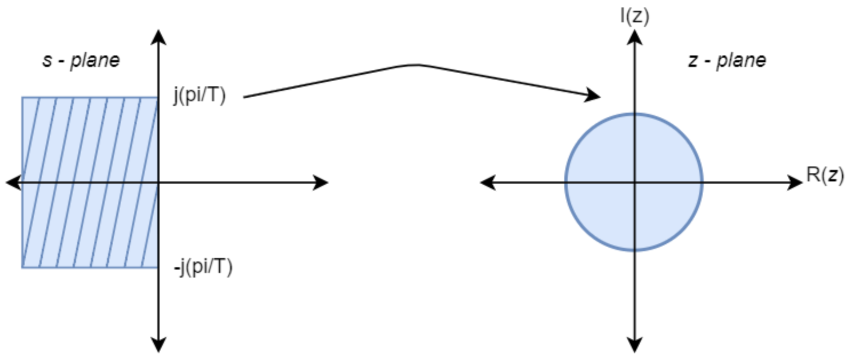


Figure 2.11: *s*-domain to *z*-domain transformation using Impulse Invariance

The impulse invariance transformation only maps poles. Since, all the poles in the *s*-plane map inside the *z*-plane's unit circle, it preserves the stability of the system.

2.5.2 Matched z-transform:

The matched *z*-transform method uses the same pole mapping process as in the impulse invariance method, but the zeros are handled in a different way. The matched *z*-transform method uses impulse invariance transformation method to map zeros as well as poles [9]. Fig. 2.12 shows the matched *z*-transformation process.

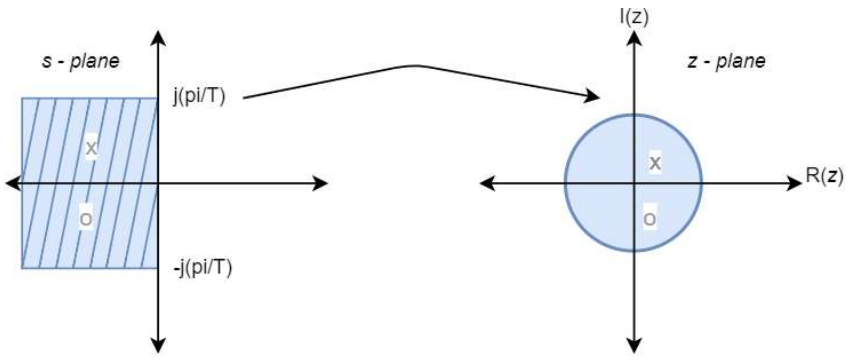


Figure 2.12: *s*-domain to *z*-domain transformation of poles and zeros using Matched *z*-transform

To illustrate consider the analog transfer function,

$$H(s) \ = \frac{\prod_{k=1}^{Q}(s-b_k)}{\prod_{k=1}^{P}(s-a_k)} \tag{2.36}$$

Using the matched *z*-transformation method, its equivalent discrete time transfer function is

$$H(z) = \frac{\prod_{k=1}^{Q}(1- e^{-b_k T} z^{-1})}{\prod_{k=1}^{P}(1- e^{-a_k T} z^{-1})}. \tag{2.37}$$

Thus, the matched *z* transformation has same digital poles as that of impulse invariance method but normally has different discrete domain zeros. Since both poles and zeros are mapped separately, this transformation method is more general and applicable to all kinds of analog filters. Since all the left-half plane poles are mapped inside the *z*-plane's unit-circle, the transformation preserves the stability of the system. The matched *z*-transform method has the same disadvantage as impulse invariance transformation method in that the signal suffers from aliasing if the sampling frequency is not fast enough. The impulse invariance transformation method is more popular than matched *z*-transformation [10].

2.5.3 Bilinear Transformation (or Trapezoidal Integration):

The bilinear Transformation is a very commonly used mapping method and is based on the trapezoid rule. For numerical integration, the trapezoidal rule is a numerical integration method that approximates the area under a curve by using trapezoids [3].

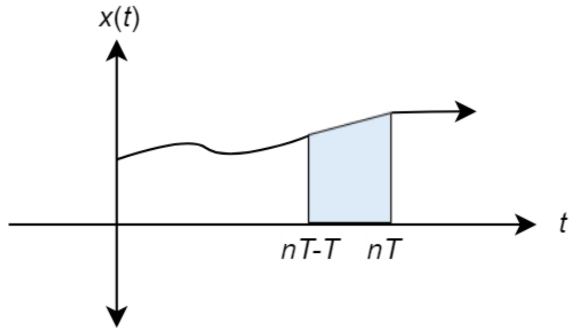


Figure 2.13: Numerical Integration using Trapezoidal Integration

For example, the area under the curve in Fig. 2.13. for the interval $nT\text{-}T \le t \le nT$ would approximated by the shaded trapezoid. The approximating formula is

$$\int_{nT-T}^{nT} x(t)dt \cong \frac{T}{2}[x(nT - T) + x(nT)] \tag{2.38}$$

Using the trapezoidal integration rule, a definite integral over the interval $0 \le t \le nT$ where n is a positive integer can be calculated by first-order difference equation

$$\mathrm{y(nT)} \cong \int_0^{nT-T} x(t)dt + \int_{nT-T}^{nT} x(t)dt = y(nT - T) + \int_{nT-T}^{nT} x(t)dt \tag{2.39}$$

Substituting (2.38) into (2.39),

$$y(nT) \cong y(nT - T) + \frac{T}{2}[x(nT - T) + x(nT)] \tag{2.40}$$

Taking the z-transform of (2.40),

$$Y(z) = Y(z)z^{-1} + \frac{T}{2}[X(z)z^{-1} + X(z)]$$

which implies that

$$\frac{Y(z)}{X(z)} = \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}} \tag{2.41}$$

To relate the trapezoidal integration transfer function in (2.41) to the s-domain, consider

$$y(t) = \int_0^t x(\tau)d\tau \tag{2.42}$$

which has the Laplace transform

$$Y(s) = \mathcal{L}\{\int_0^t x(\tau)d\tau\} = \frac{1}{s}X(s) \tag{2.43}$$

which implies that

$$H(\mathrm{s}) = \frac{Y(s)}{X(s)} = \frac{1}{s}. \tag{2.44}$$

Comparing (2.44) and (2.41),

$$\frac{1}{s} \longrightarrow \frac{T(1+z^{-1})}{2(1-z^{-1})}. \tag{2.45}$$

which described the bilinear transformation.

The bilinear transformation maps the entire $j\Omega$ axis from the s-plane into the unit circle from the z-plane i.e. $H_a(j\Omega)$ for $\infty \le \Omega \le \infty$ maps into $H(e^{jw})$ for $-\pi \le w \le \pi$. The bilinear transformation can be viewed as a two-step mapping where the first step maps the entire s-plane into a strip between $\frac{-\pi}{T} \leq j\Omega \leq \frac{\pi}{T}$ on the *s*-plane and the second step uses impulse invariance transformation $z = e^{sT}$ to map the *s*-plane to *z*-plane. Fig. 2.15. graphically illustrates this 2-step transformation process.

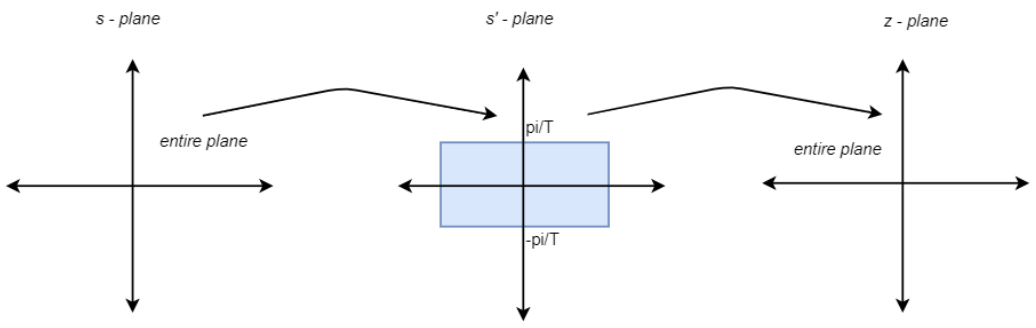


Figure 2.14: *s*-plane to *z*-plane transformation using Bilinear Transformation

The first mapping in Fig.2.14 is

$$s = \frac{2}{T} tanh(\frac{s'T}{2}). \tag{2.46}$$

The second step is impulse invariance $z = e^{sT}$ in (2.32) which implies that

$$s' = \frac{1}{T}\ln(z). \tag{2.47}$$

Substituting (2.47) into (2.46),

$$\text{s} = \frac{2}{T} tanh(\frac{1}{2}\ln(z)). \tag{2.48}$$

Because,

$$\begin{aligned} tanh(x) &= \frac{sinh(x)}{cosh(x)} \\ &= \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \\ &= \frac{1 - e^{-2x}}{1 + e^{-2x}} \end{aligned} \tag{2.49}$$

Using (2.49), (2.48) can be written as

$$s = \frac{2}{T}\frac{1 - e^{-ln(z)}}{1 + e^{-ln(z)}} = \frac{2}{T}\frac{1 - e^{-\ln(z^{-1})}}{1 + e^{-\ln(z^{-1})}} \tag{2.50}$$

(2.50) implies that

$$s \rightarrow \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}. \quad (2.51)$$

which is the bilinear transformation.

2.4.4 Delta transform (or Forward Euler Integration):

The delta transform is based on Forward Euler integration method and has the special property that as the delta transform sample time approaches zero, the delta transform converges towards its continuous-time counterpart, the Laplace transform [3].

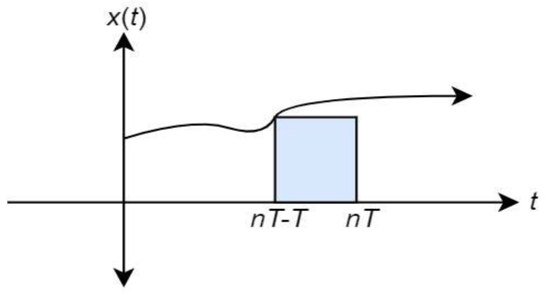


Figure 2.15: Numerical Integration using Forward Euler Integration

Fig. 2.15 illustrates forward Euler Integration method. As shown in Fig. 2.15, the area under a curve for the interval $nT - T \leq t \leq nT$ can be approximated by

$$\int_{nT-T}^{nT} x(t)dt \cong Tx(nT-T). \quad (2.52)$$

Using forward Euler Integration, a definite integral over the interval $0 \leq t \leq nT$, where *n* is a positive integer can be calculated by first-order difference equation,

$$y(nT) = \int_{0}^{nT-T} x(t)dt + \int_{nT-T}^{nT} x(t)dt = y(nT-T) + \int_{nT-T}^{nT} x(t)dt \quad (2.53)$$

Substituting (2.52) into (2.53),

$$y(nT) = y(nT-T) + Tx(nT-T) \quad (2.54)$$

Taking the *z*- transform of (2.54),

$$Y(z) = Y(z)z^{-1} + T\,[X(z)z^{-1}]$$

which implies that

$$\frac{Y(z)}{X(z)} = \frac{Tz^{-1}}{1-z^{-1}}. \tag{2.55}$$

Comparing (2.44) and (2.55),

$$\frac{1}{s} \longrightarrow \frac{Tz^{-1}}{1- z^{-1}} \tag{2.56}$$

Therefore, the delta transform relates the transfer function in *s*-domain, *H*(*s*), to transfer function in *z*-domain, *H*(*z*), by the relation

$$s \longrightarrow \frac{1-z^{-1}}{Tz^{-1}}. \tag{2.57}$$

For the delta transform,

$$\delta \rightarrow \frac{1-z^{-1}}{Tz^{-1}}. \tag{2.58}$$

which implies that

$$\delta T z^{-1} \longrightarrow 1 - z^{-1} \tag{2.59}$$

For stability in the *z*-transform, all the poles should lie inside the unit circle i.e. $|z| < 1$. Therefore, for the stability in the δ-transform, all the system's poles must lie inside the region $|1 + \delta T\,| < 1$. This region of stability is defined by a unit circle of radius, $\frac{1}{T}$ centered at $-\frac{1}{T}$. Therefore, as the sampling time, *T* approaches zero, the stability region of the delta transform becomes the left half plane which is equivalent to that of the Laplace transform. Thus, the delta-transform has superior performances at high sample rates compared to other CT-DT transformations because the discrete time models approach the equivalence CT models when the delta transform has a small sampling time.

2.4.5 Midpoint Integration Rule

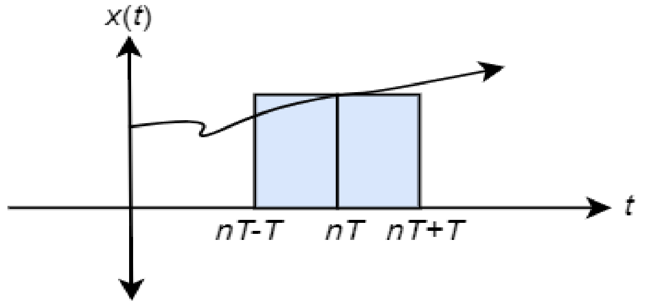


Figure 2.16: Numerical Integration using Midpoint Integration

The midpoint integration rule approximates the area under a curve using a rectangle. As shown in Fig. 2.16, the area under a curve for the interval range $nT - T \leq t \leq nT + T$ can be approximated by a rectangle of length $2T$ and height $x(nT - T)$. The approximating formula is

$$\int_{nT-T}^{nT+T} x(t)dt \cong \frac{x(nT-T)+x(nT+T)}{2} . 2\mathrm{T}$$

$$\cong 2Tx(nT) \qquad (2.60)$$

Therefore, a definite integral over the interval $0 \leq t \leq nT$, where n is a positive integer can be calculated by first-order difference equation

$$y(nT) = \int_{0}^{nT-2T} x(t)dt + \int_{nT-2T}^{nT} x(t)dt$$

$$= y(nT - 2T) + \int_{nT-2T}^{nT} x(t)dt \qquad (2.61)$$

Substituting (2.60) into (2.61),

$$y(nT) = y(nT - 2T) + 2Tx(nT - T) \qquad (2.62)$$

Taking the z- transform of (2.62),

$$Y(z) = Y(z)z^{-2} + 2T\,[X(z)z^{-1}]$$

which implies that

$$\frac{Y(z)}{X(z)} = \frac{2Tz^{-1}}{1-z^{-2}}. \qquad (2.63)$$

Comparing (2.44) and (2.60),

$$\frac{1}{s} \longrightarrow \frac{2Tz^{-1}}{1-z^{-2}} \qquad (2.64)$$

Therefore, we can relate $H(s)$ and $H(z)$ by the relation

$$s \longrightarrow \frac{1-z^{-2}}{2T\,z^{-1}}. \qquad (2.65)$$

Because, midpoint integration approximates the area under a curve using rectangles instead of using more accurate geometrics such as trapezoids as in the trapezoidal integration method, this method is not most accurate method of numerical integration.

2.4.6 Simpsons Rule

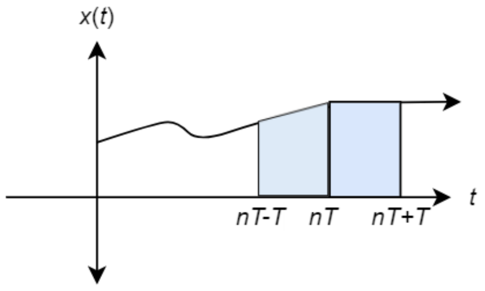


Figure 2.17: Numerical Integration using Simpsons Rule

Simpsons rule is a numerical integration method that is derived from a parabolic integration method [12]. Fig. 2.17 illustrates Simpson's rule's integration method. As shown in Fig. 2.17, the area under the curve for the interval range $nT - T \le t \le nT + T$ is approximated by

$$\int_{nT-T}^{nT+T} x(t)dt \cong \frac{T}{3}\left[x(nT+T) + 4x(nT) + x(nT-T)\right] \qquad (2.66)$$

Therefore, a definite integral over the interval $0 \le t \le nT$, where n is a positive integer can be calculated by first-order difference equation

$$y(nT) = \int_{0}^{nT-2T} x(t)dt + \int_{nT-2T}^{nT} x(t)dt = y(nT-2T) + \int_{nT-2T}^{nT} x(t)dt \qquad (2.67)$$

Substituting (2.66) into (2.67),

$$y(nT) = y(nT-2T) + \frac{T}{3}\left[x(nT-2T) + 4x(nT-T) + x(nT)\right] \qquad (2.68)$$

Applying the z-transformation to (2.68),

$$Y(z) - Y(z)\,z^{-2} = \frac{T}{3}(\,Xz^{-2} + 4\,X(z)z^{-1} + X(z))$$

$$Y(z)(1 - z^{-2}) = X(z)\,(z^{-2} + 4z^{-1} + 1)$$

which implies that

$$\frac{Y(z)}{X(z)} = \frac{T}{3}\frac{1+4z^{-1}+z^{-2}}{1-z^{-2}}\,. \tag{2.69}$$

Comparing (2.44) and (2.69),

$$\frac{1}{s} \longrightarrow \frac{T}{3}\frac{1+4z^{-1}+z^{-2}}{1-z^{-2}} \tag{2.70}$$

Therefore, Simpson's rule relates $H(s)$ to $H(z)$ by the relation

$$s \longrightarrow \frac{3}{T}\frac{1-z^{-2}}{1+4z^{-1}+z^{-2}} \tag{2.71}$$

Because Simpson's rule uses a sequence of parabolic segments instead of straight lines, it is typically more accurate than midpoint and trapezoidal integration rule. The disadvantage of Simpson's rule is that it is a more complex integration method than trapezoidal, midpoint or Euler's integration rules. To apply Simpson's rule for definite integrals using difference equations, the integral of the first interval is approximated using the trapezoidal integration rule.

2.4.7 Summary

The numerical integration methods used in this thesis along with their s-domain to z-domain transformation functions are summarized in Table 2.2.

S.N.	Numerical Integration Methods	Transformation
1.	Impulse Invariance Transformation	$s = \frac{1-z^{-1}}{T}$
2.	Bilinear Transformation (Trapezoidal Integration)	$s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$.
3.	Delta Transformation (Euler Forward Integration)	$s = \frac{1-z^{-1}}{T\,z^{-1}}$
4.	Midpoint Integration	$s = \frac{1-z^{-2}}{2T\,z^{-1}}$
5.	Simpsons Rule	$s = \frac{3(1-z^{-2})}{T(1+4z^{-1}+z^{-2})}$

Table 2.2: Numerical Integration Methods along with their *s*-*z* transformation functions

2.6 Frequency Response Comparison of Numerical Integration Methods:

Fig. 2.18 shows the ratio of the magnitude response of all the numerical integration methods such as bilinear transform or trapezoidal integration, impulse invariance transformation, midpoint integration, Simpson's rule and delta transform or Euler's forward integration to the ideal integrator's magnitude response, $|\frac{1}{jw}|$. This can be calculated by letting $z = exp(jw)$ in (2.45), (2.51), (2.56), (2.64) and (2.70).

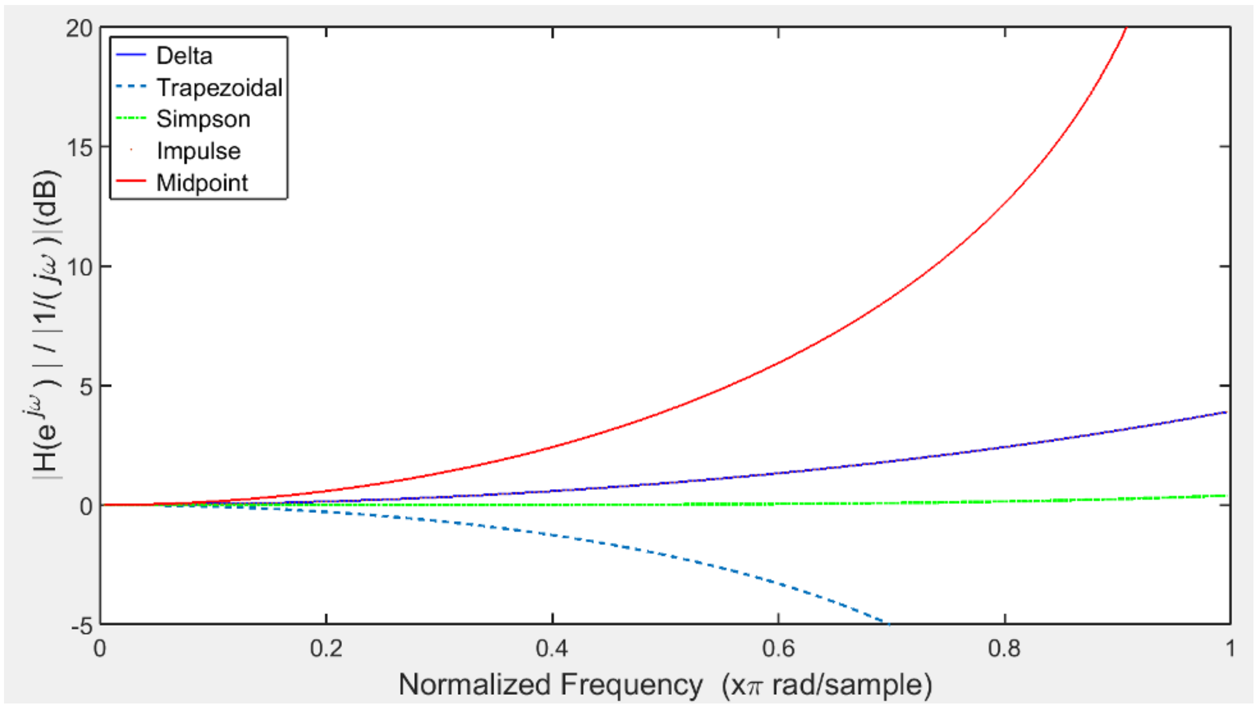


Figure 2.18: Ratio of each of the numerical integration magnitude response to an integrator's magnitude response, $|\frac{1}{jw}|$

The sampling period for numerical integration, T, is chosen so that the ∑ΔM's sampling period, T_s, is an integer multiple of T, that is, $T_s = kT$ where k is any positive integer. Fig 2.18 can be used to select the appropriate value of k to preserve frequency. It can be depicted from the plot that, Simpson's Rule provide an accurate approximation for $k \geq 1$ while the Delta require $k \geq 5$, the Bilinear and Impulse Invariance transformation require $k \geq 10$ and the Midpoint Integration require $k \geq 15$ for accurate frequency approximations. This shows Simpson's rule is closer to the ideal integration and Midpoint integration is most deviated from the ideal integration result.

Chapter 3

LITERATURE REVIEW

Although there has been a lot of research going on CT ∑ΔMs in recent years, there has been few research on the numerical integration methods and simulation methods used for modeling CT ∑ΔMs. Since integrator is one of the major circuit element of the CT ∑ΔM, proper research should be done while choosing what type of integration method works best for modeling CT ∑ΔM. Similarly, the choice of the simulation method is also very critical for improving the performance requirements of the CT ∑ΔM. Therefore, in this chapter, the research works on various numerical integration methods and simulation methods used for simulating and modeling CT ∑ΔM has been discussed.

3.1 Numerical Integration Methods used in CT ∑ΔM:

Numerical integration methods can be used to transform *s*-domain transfer functions to *z*-domain transfer functions. As a result, CT ∑ΔM can be modeled in the discrete time domain using various numerical integration methods such as the bilinear transform or trapezoidal integration, the impulse invariance transform, the matched-z transform, midpoint integration, Simpson's rule and the delta transform, or Euler's forward integration. However, there has not been a lot of research on using numerical integration methods to model CT ∑ΔMs. Some of the important research work on numerical integration methods in CT ∑ΔMs is discussed in this chapter.

In [3] and [18], K. Kang and P. Stubberud model CT ∑ΔMs using the delta transformation, which is based on Euler's forward integration method, to convert from the *s*-domain to *z*-domain. As the integration sampling period is reduced, the delta transform approaches the Laplace transformation, and thus, the discrete system's zeros and poles approach the continuous system's zeros and poles, respectively. Thus, by increasing the transform's sampling rate, a delta transform's discrete model can better represent an equivalent continuous model.

For the delta transform, the relation between a transfer function in s-domain, $H(s)$, and a z-domain transfer function $H(z)$ is related by

$$\frac{1}{s} \rightarrow \frac{T_d z^{-1}}{1-z^{-1}} \tag{3.1}$$

where T_d is the numerical integration sampling rate. To apply the transformation in (3.1), the system's integrators are replaced by the z-transform in (3.1) and converted into difference equations. The resulting delta transform model is simulated using MATLAB. In [3] and [8], second, third, fourth and fifth order single-bit CT ∑ΔMs were simulated. All the modulators have a sampling frequency of 1 GHz (T_s=1e-9) and a bandwidth of 20 MHz. The NTFs are Chebyshev Type 2 highpass filters with cut-off frequencies near the ∑ΔM's bandwidth and the STFs approximate Chebyshev Type 2 lowpass filters. The authors accomplished this by using the NTF's denominator and a Chebyshev Type 2 filter numerator. The authors implement their CT ∑ΔMs using cascade of integrator feedback (CIFB) architecture. Simulations of CIFB implementation assume the use of both RC and $G_m C$ integrators. The authors compare the simulation methods which include MATLAB/Simulink, delta transform, CT/DT transformation, SPICE modeling and solving differential equations. The comparison is with respect to speed which is based on total elapsed time taken for the simulation, accuracy which is based on the value of SQNR and simplicity. For the comparison, six second order, third order, fourth order and fifth order single-bit CT ∑ΔMs were simulated. A numerical integration oversampling ratio of 10 is used for all simulations. The simulation parameters include excess loop delays that are multiples of the ∑ΔM's sampling rate i.e. 2.5T, 2T, 1.5T, 1T, 0.5T and 0T. The authors include separate tables for comparing SQNR, SQNR difference between the simulation methods, the percentage of SQNR difference, elapsed time and performance of simulation in terms of simplicity, accuracy and speed. The tables show that the delta transform method results in accurate simulations compared to other simulation methods such as Simulink, SPICE modeling and solving differential equations. The simulation time for the delta transform model was about ten times the simulation time for modeling the CT ∑ΔM by using the CT/DT transformation. The result is because the delta transform calculates 10 times more loop filter signal values at times other than the ∑ΔMs' sampling times. Although not the fastest method, the delta transform is a very simple method that yields accurate results very close to that of SPICE simulation. Thus, the authors concluded that at reasonable speeds and without much difficulty, the delta transform can be used to get accurate results.

In [5], J. A. Cherry and W. M. Snelgrove use the impulse-invariant transformation to map DT domain transfer functions to CT domain transfer functions. The authors use a design procedure that starts by determining an H(z) and then transform it into equivalent H(s) by using the impulse-invariant transformation. In [5], two modulators are considered equivalent if for same input waveform the quantizer input voltages are same at sampling instants; i.e, H(s) and H(z) are equivalent if

$$q(n) \; = \; q_c(t)|_{t=nT} \text{ for all n} \qquad (3.2)$$

where $q(n)$ and $q_c(t)$ are the quantizer inputs of the DT ∑ΔM and CT ∑ΔM, respectively. The authors also argue that if the modulators satisfy (3.2), then the output and the noise-performance of the equivalent modulators will also be identical.

To illustrate this method, consider the equivalent DT ∑ΔM and CT ∑ΔM shown in Fig. 3.1.

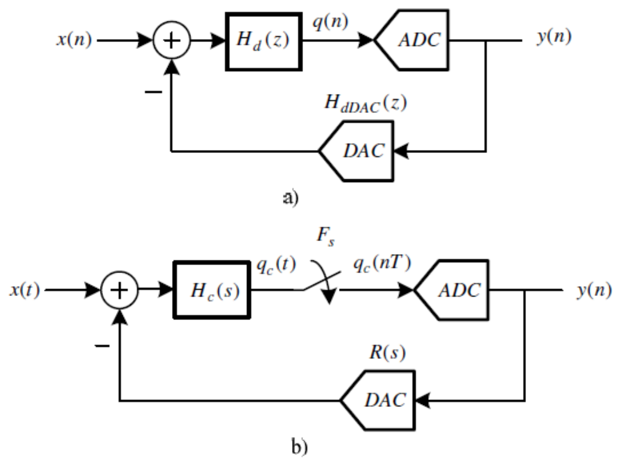


Figure 3.1: The block diagrams of a) DT ∑ΔM and b) CT ∑ΔM

If both the CT ∑ΔM's and DT ∑ΔM 's open-loop filter's impulse responses are identical at the sampling instants then

$$h(n) = h(t)|_{t=nT} \tag{3.3}$$

which implies that

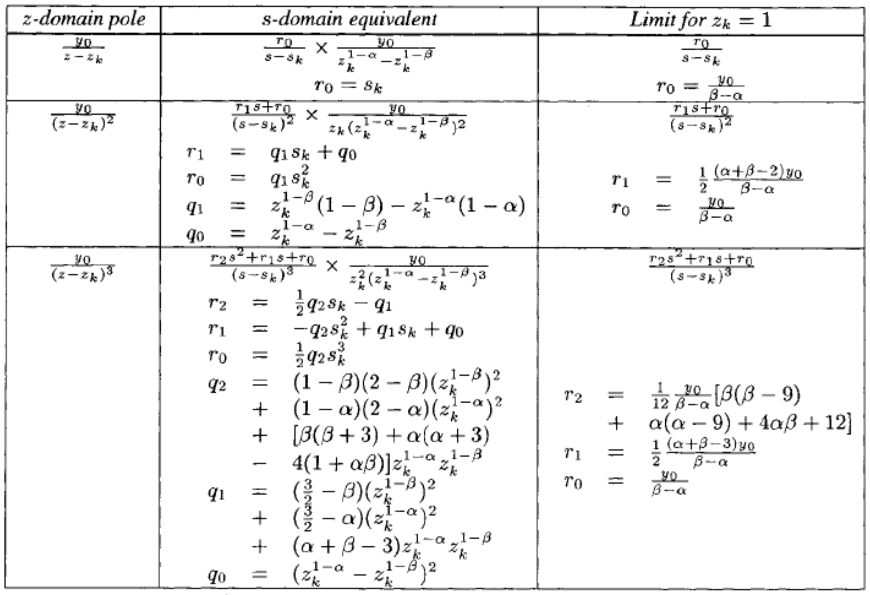
$$\mathcal{Z}^{-1}\{H_{dDAC}(z)H_d(z)\} = \mathcal{L}^{-1}\{R(s)H_c(s)\}|_{t=nT} \tag{3.4}$$

where $\mathcal{Z}^{-1}$, $\mathcal{L}^{-1}$, $R(s)$, $H_c(s)$ and $H_d(z)$ represent the inverse *z*-transform, the inverse Laplace transform, the CT DAC transfer function, the continuous-time loop filter and the discrete-time loop filter, respectively. Because $H_{dDAC}(z) = 1$, (3.4) can be simplified to

$$\mathcal{Z}^{-1}\{H_d(z)\} = \mathcal{L}^{-1}\{R(s)H_c(s)\}|_{t=nT}. \tag{3.5}$$

The transformation in (3.4) is the impulse-invariance transformation. Thus, to calculate an *H*(*s*) for a CT ∑ΔM with identical noise shaping behavior as that of *H*(*z*) for a DT ∑ΔM, *H*(*z*) is expanded into partial fractions. The authors then use a table they created to transform the poles of *H*(*z*) to *s*-domain poles using the following formula,

$$s_k = \ln(z_k). \tag{3.6}$$



z-domain pole	*s-domain equivalent*	*Limit for $z_k = 1$*
$\frac{y_0}{z-z_k}$	$\frac{r_0}{s-s_k} \times \frac{y_0}{z_k^{1-\alpha}-z_k^{1-\beta}}$ $\quad r_0 = s_k$	$\frac{r_0}{s-s_k}$ $\quad r_0 = \frac{y_0}{\beta-\alpha}$
$\frac{y_0}{(z-z_k)^2}$	$\frac{r_1 s+r_0}{(s-s_k)^2} \times \frac{y_0}{z_k(z_k^{1-\alpha}-z_k^{1-\beta})^2}$ $\begin{aligned} r_1 &= q_1 s_k + q_0 \\ r_0 &= q_1 s_k^2 \\ q_1 &= z_k^{1-\beta}(1-\beta) - z_k^{1-\alpha}(1-\alpha) \\ q_0 &= z_k^{1-\alpha} - z_k^{1-\beta} \end{aligned}$	$\frac{r_1 s+r_0}{(s-s_k)^2}$ $\begin{aligned} r_1 &= \frac{1}{2}\frac{(\alpha+\beta-2)y_0}{\beta-\alpha} \\ r_0 &= \frac{y_0}{\beta-\alpha} \end{aligned}$
$\frac{y_0}{(z-z_k)^3}$	$\frac{r_2 s^2+r_1 s+r_0}{(s-s_k)^3} \times \frac{y_0}{z_k^2(z_k^{1-\alpha}-z_k^{1-\beta})^3}$ $\begin{aligned} r_2 &= \tfrac{1}{2}q_2 s_k - q_1 \\ r_1 &= -q_2 s_k^2 + q_1 s_k + q_0 \\ r_0 &= \tfrac{1}{2}q_2 s_k^3 \\ q_2 &= (1-\beta)(2-\beta)(z_k^{1-\beta})^2 \\ &+ (1-\alpha)(2-\alpha)(z_k^{1-\alpha})^2 \\ &+ [\beta(\beta+3)+\alpha(\alpha+3) \\ &- 4(1+\alpha\beta)]z_k^{1-\alpha}z_k^{1-\beta} \\ q_1 &= (\tfrac{3}{2}-\beta)(z_k^{1-\beta})^2 \\ &+ (\tfrac{3}{2}-\alpha)(z_k^{1-\alpha})^2 \\ &+ (\alpha+\beta-3)z_k^{1-\alpha}z_k^{1-\beta} \\ q_0 &= (z_k^{1-\alpha}-z_k^{1-\beta})^2 \end{aligned}$	$\frac{r_2 s^2+r_1 s+r_0}{(s-s_k)^3}$ $\begin{aligned} r_2 &= \tfrac{1}{12}\tfrac{y_0}{\beta-\alpha}[\beta(\beta-9) \\ &+ \alpha(\alpha-9)+4\alpha\beta+12] \\ r_1 &= \tfrac{1}{2}\tfrac{(\alpha+\beta-3)y_0}{\beta-\alpha} \\ r_0 &= \tfrac{y_0}{\beta-\alpha} \end{aligned}$

Table 3.1: *s*-domain equivalent of *z*-domain poles

The authors table is shown in Table 3.1. Using Table 3.1, each partial fraction *z*-domain pole of *H*(*z*) is converted into an equivalent *s*-domain pole. After that the *s*-domain poles are combined with a rectangular DAC pulse shape to get *H*(*s*). The pulse shape $r(t)$ has a magnitude of 1 from α to β which implies that

$$r(\alpha,\beta)(t) \;=\begin{cases} 1, & \alpha \le t < \beta, 0 \le \alpha < \beta \le 1 \\ 0, & otherwise \end{cases} \tag{3.7}$$

which implies that

$$R(s) = \frac{e^{-\alpha T s} - e^{-\beta T s}}{s} \tag{3.8}$$

and

$$H_c(s) = \frac{H(s)}{R(s)}. \tag{3.9}$$

where $H(s)$ is the transfer function of the equivalent CT ∑ΔM, $R(s)$ is the transfer function of the pulse shape $r(t)$ and $H_c(s)$ is the transfer function of the required continuous-time loop filter. Thus, to determine a CT loop filter, a DT loop filter that meets the required performance specifications is designed and then the equivalent CT ∑ΔM DAC pulse shape is obtained by using the impulse invariance transform.

In their book “Continuous Time Sigma Delta Analog to Digital Conversion” [4], Ortmanns and F. Gerfers use the impulse-invariant transformation and modified z-transformation for DT to CT transformation. The authors recommend designing a CT ∑ΔM by designing a DT loop filter $H(z)$, simulating the ideal DT ∑ΔM to speed up the design procedure and then proceed with a DT domain to CT domain conversion to obtain the equivalent CT ∑ΔM. The authors use an impulse invariance transformation technique similar to the one used by J. A. Cherry and W. M. Snelgrove in [5]. The authors also use Table 3.1 to transform z-domain poles to their equivalent s-domain pole and discuss on the possibility of transforming every DT loop filter into an equivalent CT loop filter.

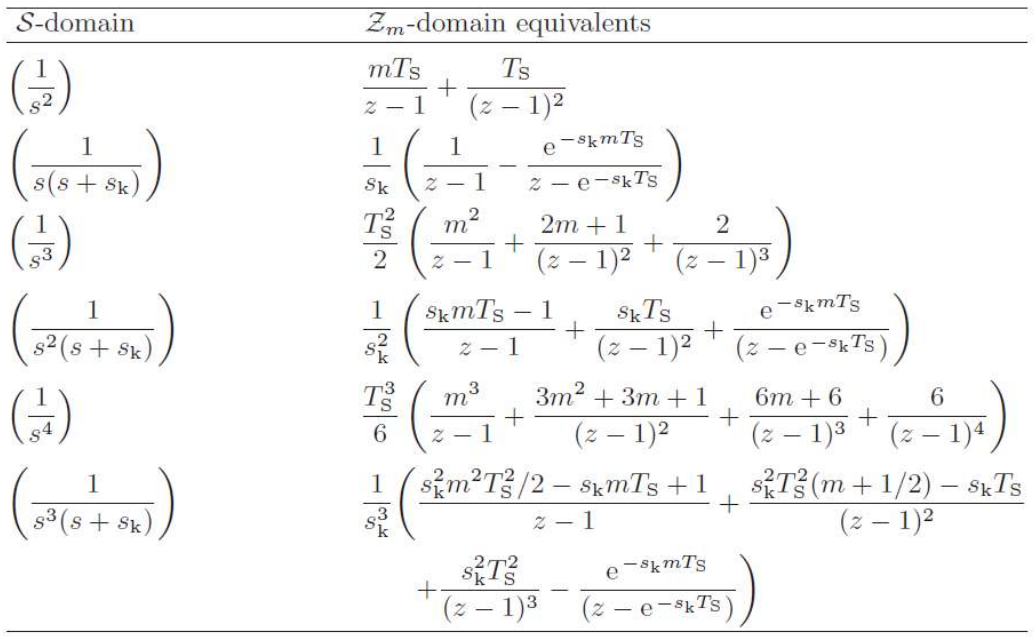
The authors use a modified z-transform which is an extension of the general z-transform because it calculates discrete system behavior at all instants of time and this property is very useful for mixed signal and multirate sampled systems. As in the impulse invariance transformation, the discrete-time loop transfer function is calculated first and compared with the original discrete-time loop transfer function. This transform can be written as

$$H(z) = \sum_i Z_m\{H(s)R_{DAC}(s)\} \tag{3.10}$$

where $H(s)$ is multiplied with the desired system function $R_{DAC}(s)$ of the DAC and Z_m is the modified z-transform. In (3.10), m is the delay factor and is a very important parameter when using the modified z-transform. The value of m is normalized and bounded in the range $0 < m < 1$, where 0 means the previous sample instant and 1 means the next sample instant. An additional delay parameter is introduced for every time instance when there is change in the CT loop filter’s behavior. For an ideal NRZ DAC pulse:

- The rising edge of the DAC pulse at $t = 0$ is the first instant, which results in $m_1 = 1 - \frac{0}{T_s} = 1$.
- The falling edge at $t = T_s$ is the second instant. Therefore, m_2 yields $m_2 = 1 - \frac{T_s}{T_s} = 0$.

Next, each of the loop filter's term is mapped with respect to all the time instances according to Table 3.2.



$\mathcal{S}$-domain	$\mathcal{Z}_m$-domain equivalents
$\left(\frac{1}{s^2}\right)$	$\frac{mT_\mathrm{S}}{z-1} + \frac{T_\mathrm{S}}{(z-1)^2}$
$\left(\frac{1}{s(s+s_\mathrm{k})}\right)$	$\frac{1}{s_\mathrm{k}}\left(\frac{1}{z-1} - \frac{\mathrm{e}^{-s_\mathrm{k}mT_\mathrm{S}}}{z-\mathrm{e}^{-s_\mathrm{k}T_\mathrm{S}}}\right)$
$\left(\frac{1}{s^3}\right)$	$\frac{T_\mathrm{S}^2}{2}\left(\frac{m^2}{z-1} + \frac{2m+1}{(z-1)^2} + \frac{2}{(z-1)^3}\right)$
$\left(\frac{1}{s^2(s+s_\mathrm{k})}\right)$	$\frac{1}{s_\mathrm{k}^2}\left(\frac{s_\mathrm{k}mT_\mathrm{S}-1}{z-1} + \frac{s_\mathrm{k}T_\mathrm{S}}{(z-1)^2} + \frac{\mathrm{e}^{-s_\mathrm{k}mT_\mathrm{S}}}{(z-\mathrm{e}^{-s_\mathrm{k}T_\mathrm{S}})}\right)$
$\left(\frac{1}{s^4}\right)$	$\frac{T_\mathrm{S}^3}{6}\left(\frac{m^3}{z-1} + \frac{3m^2+3m+1}{(z-1)^2} + \frac{6m+6}{(z-1)^3} + \frac{6}{(z-1)^4}\right)$
$\left(\frac{1}{s^3(s+s_\mathrm{k})}\right)$	$\frac{1}{s_\mathrm{k}^3}\left(\frac{s_\mathrm{k}^2m^2T_\mathrm{S}^2/2 - s_\mathrm{k}mT_\mathrm{S}+1}{z-1} + \frac{s_\mathrm{k}^2T_\mathrm{S}^2(m+1/2) - s_\mathrm{k}T_\mathrm{S}}{(z-1)^2} + \frac{s_\mathrm{k}^2T_\mathrm{S}^2}{(z-1)^3} - \frac{\mathrm{e}^{-s_\mathrm{k}mT_\mathrm{S}}}{(z-\mathrm{e}^{-s_\mathrm{k}T_\mathrm{S}})}\right)$

Table 3.2: Modified *z*-transform for corresponding loop filter order

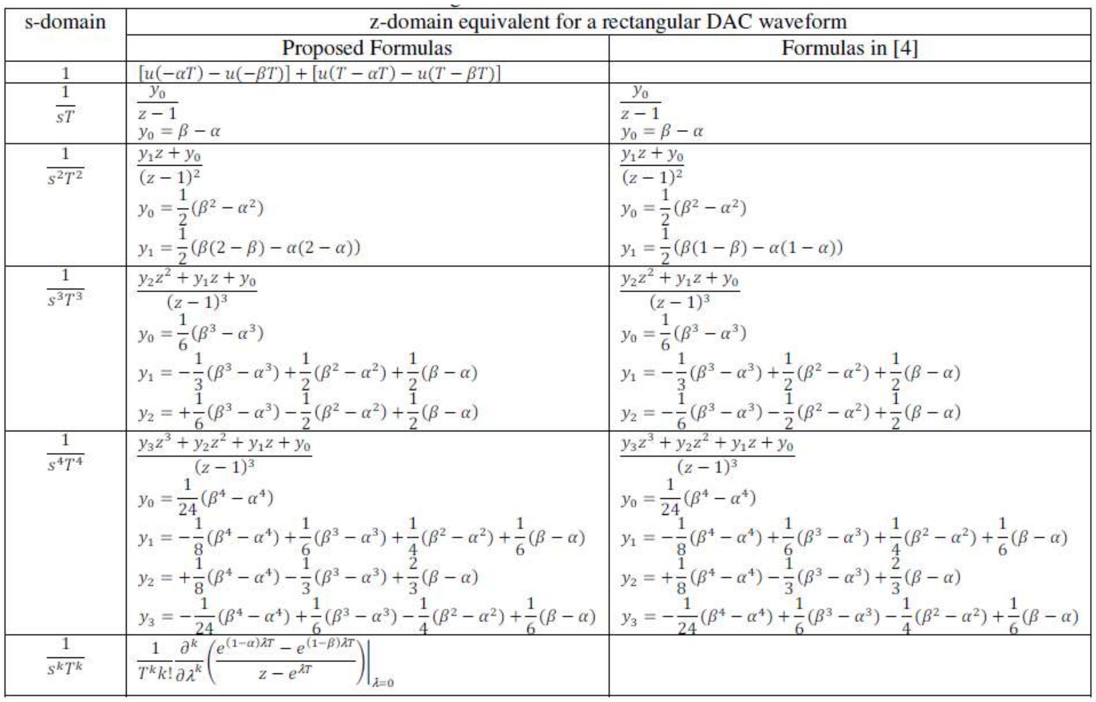
Finally, the coefficients of the CT ∑ΔM's loop transfer function are obtained by comparing coefficients with the original DT ∑ΔM's loop filter function.

In [21], J. Talebzadeh and I. Kale present a general formula that uses the impulse invariant transformation to convert nth-order DT ∑ΔMs to equivalent nth-order CT ∑ΔMs. The authors use a method of determining an equivalent CT ∑ΔM from a DT ∑ΔM that is similar to the method used by J. A. Cherry and W. M. Snelgrove in [5] and M. Ortmanns and F. Gerfers in [4]. This method uses the impulse invariance transformation equation in (3.4). The authors also consider using DAC waveforms that are similar to the DAC waveforms in (3.5) and (3.6). They also derive an equivalent z-domain transfer function of a CT ∑ΔM. For first order s-domain equations, the z-domain equivalent formula they derive is

$$H_{1d}(z) = T\frac{\beta - \alpha}{z-1} \tag{3.11}$$

which implies that

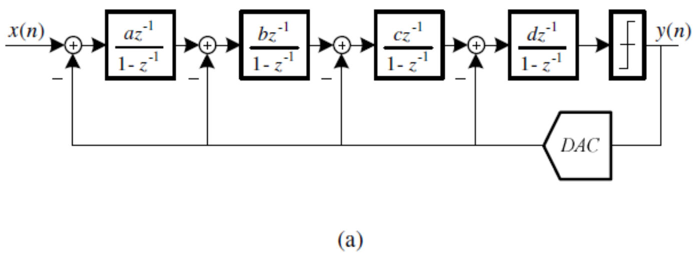
$$\frac{1}{s} \rightarrow T\frac{(\beta - \alpha)z^{-1}}{1 - z^{-1}} \tag{3.12}$$



s-domain	z-domain equivalent for a rectangular DAC waveform	
	Proposed Formulas	Formulas in [4]
1	$[u(-\alpha T) - u(-\beta T)] + [u(T - \alpha T) - u(T - \beta T)]$	
$\frac{1}{sT}$	$\frac{y_0}{z-1}$ $y_0 = \beta - \alpha$	$\frac{y_0}{z-1}$ $y_0 = \beta - \alpha$
$\frac{1}{s^2T^2}$	$\frac{y_1 z + y_0}{(z-1)^2}$ $y_0 = \frac{1}{2}(\beta^2 - \alpha^2)$ $y_1 = \frac{1}{2}(\beta(2-\beta) - \alpha(2-\alpha))$	$\frac{y_1 z + y_0}{(z-1)^2}$ $y_0 = \frac{1}{2}(\beta^2 - \alpha^2)$ $y_1 = \frac{1}{2}(\beta(1-\beta) - \alpha(1-\alpha))$
$\frac{1}{s^3T^3}$	$\frac{y_2 z^2 + y_1 z + y_0}{(z-1)^3}$ $y_0 = \frac{1}{6}(\beta^3 - \alpha^3)$ $y_1 = -\frac{1}{3}(\beta^3 - \alpha^3) + \frac{1}{2}(\beta^2 - \alpha^2) + \frac{1}{2}(\beta - \alpha)$ $y_2 = +\frac{1}{6}(\beta^3 - \alpha^3) - \frac{1}{2}(\beta^2 - \alpha^2) + \frac{1}{2}(\beta - \alpha)$	$\frac{y_2 z^2 + y_1 z + y_0}{(z-1)^3}$ $y_0 = \frac{1}{6}(\beta^3 - \alpha^3)$ $y_1 = -\frac{1}{3}(\beta^3 - \alpha^3) + \frac{1}{2}(\beta^2 - \alpha^2) + \frac{1}{2}(\beta - \alpha)$ $y_2 = -\frac{1}{6}(\beta^3 - \alpha^3) - \frac{1}{2}(\beta^2 - \alpha^2) + \frac{1}{2}(\beta - \alpha)$
$\frac{1}{s^4T^4}$	$\frac{y_3 z^3 + y_2 z^2 + y_1 z + y_0}{(z-1)^3}$ $y_0 = \frac{1}{24}(\beta^4 - \alpha^4)$ $y_1 = -\frac{1}{8}(\beta^4 - \alpha^4) + \frac{1}{6}(\beta^3 - \alpha^3) + \frac{1}{4}(\beta^2 - \alpha^2) + \frac{1}{6}(\beta - \alpha)$ $y_2 = +\frac{1}{8}(\beta^4 - \alpha^4) - \frac{1}{3}(\beta^3 - \alpha^3) + \frac{2}{3}(\beta - \alpha)$ $y_3 = -\frac{1}{24}(\beta^4 - \alpha^4) + \frac{1}{6}(\beta^3 - \alpha^3) - \frac{1}{4}(\beta^2 - \alpha^2) + \frac{1}{6}(\beta - \alpha)$	$\frac{y_3 z^3 + y_2 z^2 + y_1 z + y_0}{(z-1)^3}$ $y_0 = \frac{1}{24}(\beta^4 - \alpha^4)$ $y_1 = -\frac{1}{8}(\beta^4 - \alpha^4) + \frac{1}{6}(\beta^3 - \alpha^3) + \frac{1}{4}(\beta^2 - \alpha^2) + \frac{1}{6}(\beta - \alpha)$ $y_2 = +\frac{1}{8}(\beta^4 - \alpha^4) - \frac{1}{3}(\beta^3 - \alpha^3) + \frac{2}{3}(\beta - \alpha)$ $y_3 = -\frac{1}{24}(\beta^4 - \alpha^4) + \frac{1}{6}(\beta^3 - \alpha^3) - \frac{1}{4}(\beta^2 - \alpha^2) + \frac{1}{6}(\beta - \alpha)$
$\frac{1}{s^kT^k}$	$\frac{1}{T^k k!}\frac{\partial^k}{\partial \lambda^k}\left(\frac{e^{(1-\alpha)\lambda T} - e^{(1-\beta)\lambda T}}{z - e^{\lambda T}}\right)\Big\rvert_{\lambda=0}$	

Table 3.3: CT-to-DT transformation for rectangular DAC waveforms

The authors provide a conversion formula table which is shown in Table (3.3) for the impulse invariance transformation and compare the table with conversion table given in [4] and [5]. A comparison of the tables shows that y_1 in second order term and y_2 in third order term are different. To validate their formula, the authors use the formula to convert a single loop fourth order DT ∑ΔM into a single loop fourth order CT ∑ΔM. They implemented it in a three-bit fourth-order DT ∑ΔM with an OSR of 64 and converted it into a three-bit fourth-order CT ∑ΔM using their formula. They used the `Schreier` toolbox for the conversion and used NonReturn-to-Zero (NRZ) DAC waveforms. An extra feedback was given to compensate for excess loop delay. The values of α and β were chosen as 0.2 and 1.2 respectively. A sinusoidal input signal of 0.7 V amplitude and a frequency 61.34 KHz was applied to both modulators. The SNR obtained from the CT ∑ΔM was around 130.21 dB and that from DT ∑ΔM was 130.37 dB with a bandwidth of 625 MHz and clock frequency of 80 MHz Similar output spectra and in-band noise has been observed for both CT ∑ΔM and DT ∑ΔM. Their block diagrams of equivalent fourth-order CT ∑ΔM and DT ∑ΔM are shown in Fig 3.2.



x(n)
$\frac{az^{-1}}{1-z^{-1}}$
$\frac{bz^{-1}}{1-z^{-1}}$
$\frac{cz^{-1}}{1-z^{-1}}$
$\frac{dz^{-1}}{1-z^{-1}}$
y(n)
DAC
(a)

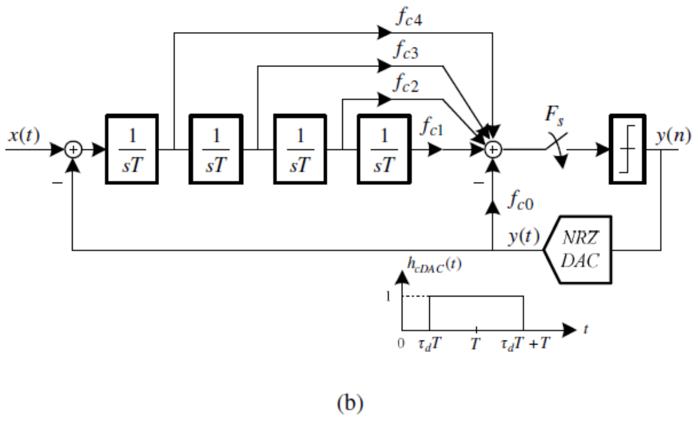


Figure 3.2: The block diagram of fourth-order: a) DT ∑ΔM and b) CT ∑ΔM

Similarly, the obtained combined output spectra of DT ∑ΔM and CT ∑ΔM is given in Fig 3.3.

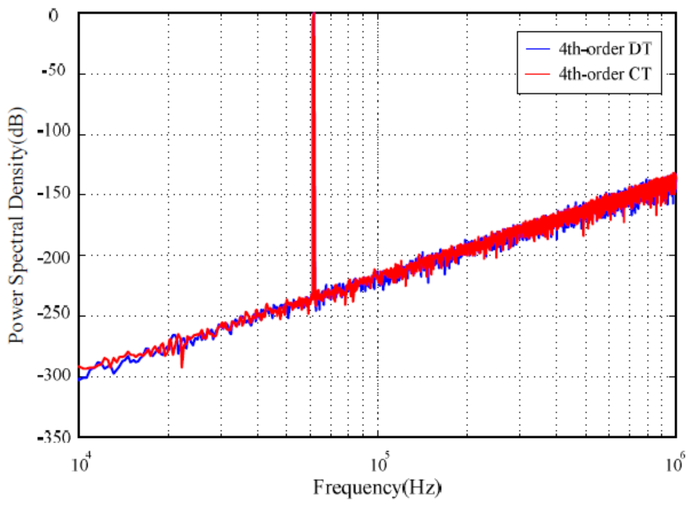


Figure 3.3: The combined output spectra of fourth-order DT ∑ΔM and CT ∑ΔM

In Fig. 3.3, it can be observed that the spectra and in-band noise of both DT ∑ΔM and CT ∑ΔM are similar. Thus, the authors concluded that the resulting CT ∑ΔM modulator performed like the initial discrete-time ∑ΔM without any degradation in performance. Therefore, similar results of both DT ∑ΔM and CT ∑ΔM supports the validity of the formula described in their paper.

3.2 Simulation Methods used in CT ∑ΔM:

CT ∑ΔMs are comparatively more difficult to design and simulate than DT ∑ΔMs because of the mixed signal nature of CT ∑ΔMs which use both analog and digital circuits in their loops [3]. Several approaches for simulating CT ∑ΔMs have been developed and implemented such as using SystemC-AMS, difference equations, Simulink, Verilog-AMS, VHDL-AMS, Cadence, SPICE modeling and solving and implementing differential equations analytically and numerically**.** Each simulation method has a tradeoff between various metrics such as speed, accuracy, and simplicity. In this section, we will discuss some relevant papers on simulation methods used for CT ∑ΔMs.

In [22], G. Zheng, S. P. Mohanty and E. Kougianos compare MATLAB/Simulink and Verilog analog and mixed signal (AMS) simulation models of a single-bit CT ∑ΔM. Digital languages such as VHDL, Verilog, SystemVerilog and SystemC are used to simulate discrete-time systems;

and languages such as VHDL-AMS, Verilog-AMS, and SystemC-AMS are used to simulate analog and mixed signal systems. In [22], a CT ∑ΔM is designed for a biomedical application that require a signal bandwidth of 10 KHz and at least 10-bits of resolution. Fig. 3.4 shows the design flow used to design CT ∑ΔM in [22].

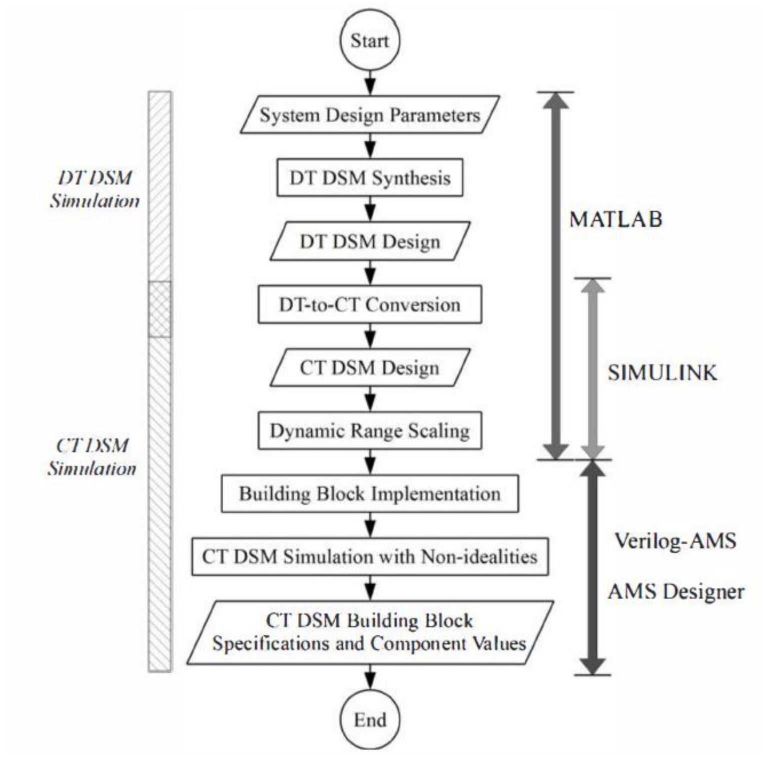


Figure 3.4: Proposed system level design flow of Continuous Time (CT) ∑Δ

Because the design methods for DT ∑ΔMs are more mature than for CT ∑ΔMs, the authors first designed a DT ∑ΔM to meet the required specifications and performance parameters using MATLAB delta-sigma toolbox. After the system-level synthesis and design of the DT ∑ΔM, the DT ∑ΔM design was mapped to a CT ∑ΔM topology. To synthesize the DT ∑ΔM, the NTF was designed first using the MATLAB synthesizeNTF function provided in MATLAB's delta sigma design toolbox. The synthesized NTF function uses design specifications such as the order of the ∑ΔM, oversampling ratio (OSR), quantization levels and out-of-band gain (OBG). OBG determines the gain for signal at the sampling frequency and offers lower in-band noise but at the cost of increasing instability and higher jitter noise. The NTF was then evaluated in the frequency domain to verify that the ∑ΔM meets the performance requirements and is also stable. To ensure that the outputs of all the stages of the modulator to be bounded, the authors performed required dynamic range scaling. The scaling is done by bounding the integrators outputs to the allowable range that is determined by the range of the power supply. For DT-CT conversion, the authors selected a cascade of integrators with feedforward (CIFF) loop filter architecture; and computed the coefficients for the loop filter of the CT ∑ΔM architecture by using simulations to match the impulse responses of the integrators of the CT ∑ΔM and DT ∑ΔM. The authors wrote a script in MATLAB to control the simulation flow, and to numerically determine the CT ∑ΔM's loop filter coefficients. After determining the CT ∑ΔM coefficients, the CT ∑ΔM was modeled in Verilog-AMS and Simulink. Simulink contains built-in libraries of quantizers, integrators, summers, etc and a behavioral model of the CT ∑ΔM can be easily built using Simulink. For Verilog-AMS, the behavioral model can be built by creating symbols and writing Verilog code to describe the symbols. After the dynamic range scaling, the proper dynamically scaled loop filters coefficients were determined. Thus, an equivalent CT ∑ΔM of the DT ∑ΔM was obtained.

The authors also modeled two critical non-idealities of the CT ∑ΔM, finite gain bandwidth product of the integrator and clock jitter of the quantizer. The authors used Simulink with ideal building blocks such as a sampling clock that has no jitter and integrators with infinite gain bandwidth. As a result, clock jitter and finite gain bandwidth were not modeled using Simulink. However, the authors model these non-idealities using Verilog-AMS. To model the finite gain bandwidth product, the authors created a Verilog-AMS algorithm that model the integrator with the proper component values of the resistors, capacitors and operational amplifiers. To model clock jitter, the authors used the Verilog-AMS function $rdist_normal.

The authors compared the Simulink and Verilog-AMS CT ∑ΔM models using simulation speed, simplicity, performance and accuracy. Simulation performance and accuracy was determined by comparing the CT ∑ΔM's power spectral density with the power spectral density of DT ∑ΔM. Using Simulink's Ode23s type solver, the authors observed that while maintaining comparable accuracy of both simulation methods, the Simulink simulation required almost double simulation time the Verilog-AMS simulation. The authors state this result may be due to setting the relative tolerance of the Simulink simulation to be half of the Verilog-AMS relative tolerance. The authors conclude that Simulink model was simpler to set up because Simulink has the required blocks in its library and it was even simpler to modify the designs. However, modeling non-idealities such as clock-jitter using Simulink was difficult because the blocks in Simulink do not model clock jitter. Verilog-AMS can easily model non-idealities with a few lines of code, that can be easily integrated with the actual circuit parameters. The authors concluded that Simulink is very suitable for system modeling in high level and Verilog-AMS tool is suitable for lower level system modeling that includes non-idealities.

In [3] and [18], K. Kang and P. Stubberud compare simulation methods such as SPICE modeling, MATLAB/Simulink, delta transform, CT/DT transformation, and solving differential equations for simulating second, third, fourth and fifth single-bit CT ∑ΔMs. The comparison is with respect to speed which is based on total elapsed time taken for the simulation, accuracy which is based on the value of SQNR and also simplicity of the simulation method. SPICE simulations begin with macro level simulations by using ideal components such as voltage controlled current sources or voltage controlled voltage sources and ideal quantizers. After meeting the performance specifications using the macro level ideal components, a little higher level or transistor level systems such as transconductance amplifiers, operational amplifiers, and digital to analog converters (DACs) can be replaced for macrolevel components to observe the non-ideal effects such as finite bandwidths, finite amplifier gains, parasitic capacitances and quantizer metastability. Since the non-ideal effects such as clock jitter and finite gain bandwidth product can be easily reflected in the circuit, SPICE simulation is expected to give more accurate results. MATLAB/Simulink simulations are relatively fast and simple to implement. In Simulink, modeling is done by selecting the required functional blocks for the continuous-time integrators, summers, gains, quantizers, input signals, clocks, DACs and so on. However, the non-idealities cannot be modeled in Simulink because Simulink's blocks are ideal blocks. In [3] and [18], the delta transform was used to model CT ∑ΔMs by converting the differential equations to difference equations using (2.72). The resulting difference equations were implemented using MATLAB code. In [3] and [18], the authors also used differential equations to simulate CT ∑ΔM. When using differential equations, the non-ideal effects such as finite bandwidths and finite amplifier gains can be modeled using the equations. The modulator's performance was determined by solving differential equations numerically or analytically using the modulator's input signal and the output signal form the quantizer's feedback to determine the input signal at the quantizer's next sample. The authors observed that this method is faster than SPICE but comparatively slower and not as simple as the other methods. The simulation results obtained by solving the differential equations were closest to the simulations results obtained by SPICE simulation. SQNRs obtained by delta transform method were observed to be very close to those obtained using SPICE simulations which the authors assume is the most accurate method of simulation. Similarly, CT/DT transform simulation results in [3] and [8] are also similar to that of SPICE. SQNR results obtained from simulation methods such as Simulink are noticeably different to those obtained by SPICE. The authors report that SPICE modeling is the slowest method whereas CT/DT transformation method which takes only few seconds to complete is the fastest method. Simulink is the second fastest method but also the simplest method of simulating CT ∑ΔMs.

In [23], P. Benabes and C. Tugui use Simulink and VHDL-AMS to model CT sigma delta modulators. Because transistor level simulators such as Cadence and PSpice require large computation times, the authors suggest using effective high-level system modeling using software such as Simulink to reduce the computation time. The authors present a design methodology that uses software tools to translate analog circuits in Cadence schematics into macro-models for VHDL-AMS and MATLAB/Simulink. This process is shown in the Fig. 3.5.

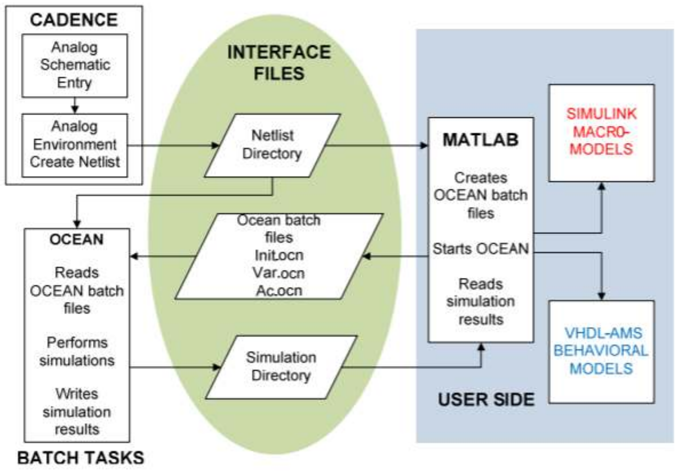


Figure 3.5: Macro-model extraction framework

The authors used CADENCE’s Open Command Environment for Analysis (OCEAN) software to control and interface various software tools such as MATLAB and Cadence SPECTRE. OCEAN started all the simulations. The results of the simulation were read by MATLAB by using Cadence functions via Cadence’s Virtuoso Multi-Mode Simulation (MMSIM) Spectre/RF toolbox. MATLAB then automated the analog simulation process, extracted the *s*-domain models of the transfer functions, combined the gains, offsets and nonlinearities and, synthesized the required macro-model. After this, the macro-model was used by MATLAB/Simulink and VHDL-AMS.

The transfer function was modeled using four types of macro-models, Current Controlled Current Source (CCCS), Voltage Controlled Current Source (VCCS), Voltage Controlled Voltage Source (VCVS), and, Current Controlled Voltage Source (CCVS). These macro models can be implemented in Simulink blocks and VHDL-AMS modules and can be used for implementation in system level. For example, Fig. 3.6 shows a Simulink macro-model of VCVS type.

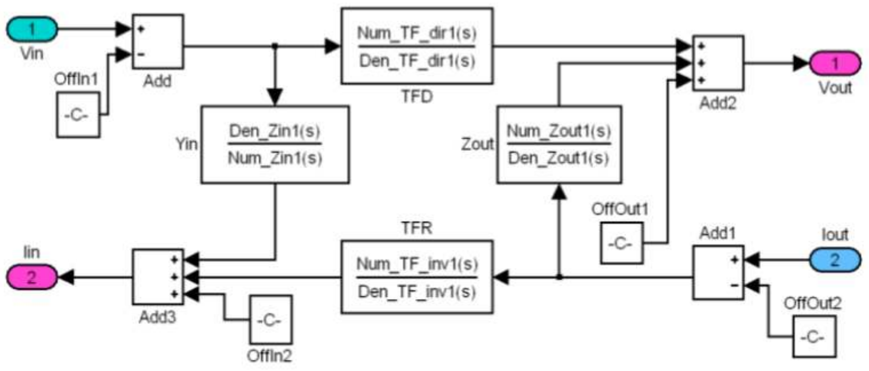


Figure 3.6: VCVS Simulink macro-model

The authors claim that all unipolar/differential circuits and all multiple-input multiple-output designs can be extracted with these Simulink models. The model extraction technique was applied to sixth-order CT ∑ΔM and for this, MATLAB CADENCE interface started common mode and did differential AC analyses on the circuit inputs and separate AC analyses was done for the outputs. The extraction algorithm depended on the complexity on the transistor-level function in terms of the zeros and poles versus the maximum order of the selected *s*-model. To provide offsets extraction, DC analyses was done on the input and output. A transient analysis was performed to determine the system’s impulse response and verify the stability of the design. In this way, the entire CT ∑ΔM was simulated using CADENCE at the transistor-level and Simulink and VHDL-AMS at the macro model level. The authors performed simulation of sixth-order CT ∑ΔM and this resulted in a considerable speed improvement over SPICE and resulted in consistent results. Ode15s type solver was used in Simulink. For 1000 output samples, SPICE simulation required 3h37m1s whereas Simulink simulation required 7m8s and VHDL-AMS simulation required 6m41s. VHDL-AMS was observed to be the fastest method of simulation. Therefore, in this paper, the authors developed a Simulink – CADENCE – VHDL-AMS framework for the model extraction and this system level implementation was applied for designing sixth-order CT ∑ΔM. The authors concluded that this method resulted in a consideration amount of speed improvement and consistent results.

In [24], M. Webb and H. Tang present a system-level simulation of CT ∑ΔM in MATLAB/Simulink. In this paper, the authors described methods on how CT ∑ΔM non-idealities such as clock jitter, operational amplifier noise, integrator non-idealities, finite DC gain, slew rate, finite bandwidth, amplifier saturation and transconductor nonlinearity can be implemented in Simulink. After modeling the nonidealities associated with the ideal functional blocks of a CT ∑ΔM, the authors derived a complete fourth-order CT ∑ΔM block diagram that modeled all the non-idealities. The authors claim that the derived block diagram's specifications can further be applied as inputs in circuit-level designs.

The authors use Simulink's sign block to model a single-bit quantizer. To model the quantizer's clock jitter which is variation in the quantizer's clock period, a normally distributed random number with zero mean is added to the sample time in the Simulink's sign block. This can be done as

$$ComparatorTime = T_s + randn * stddev \tag{3.13}$$

where MATLAB's function $randn$ generates a normally distributed random number with zero mean. The desired standard deviation is achieved by multiplying $randn$ with a scaling factor

$stddev$. This results in non-uniform sampling and whitening of the quantization noise which degrades the SNR. The authors found experimentally that $stddev$ must be less than $5.6e-4T_s$ to prevent SNR degradation of more than 10dB.

The authors use Simulink's gain block to model an amplifier. To model amplifier noise, the authors add a normally distributed random number with zero mean to the amplifier's output. Because of the NTF, the noise at the first integrator adds the most noise power to the CT ∑ΔM's output so the authors introduce noise only at the first integrator.

Since most of the CT ∑ΔM nonidealities are in located in the integrator, the authors designed a non-ideal integrator model that models the integrator nonidealities such as finite bandwidth, slew rate, finite gain, and saturation time. Fig.3.7 shows this non-ideal integrator model.

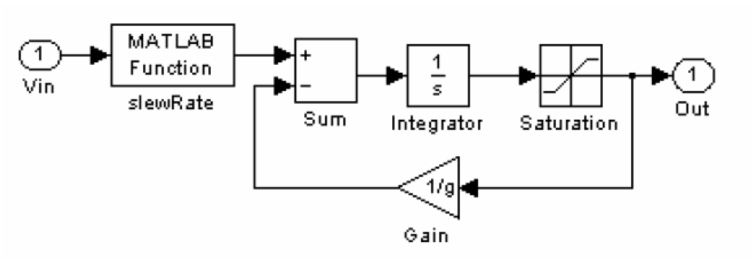


Figure 3.7: Model of non-ideal integrator

For example, in Fig. 3.7, the finite DC gain was obtained by subtracting a fraction of output from the input of the integrator which is contained in the gain block in the feedback loop.

The slew rate and the finite bandwidth of the amplifier are modeled in the user-defined function before the summer which is shown in Fig. 3.7. This function implements the following condition:

$$|\varepsilon| = \begin{cases} \begin{cases} |Vin| - SR * Ts & if\ tsl \geq Ts \\ (|Vin| - SR * tsl) * e^{\frac{-t\exp}{\tau}} & if\ tsl < Ts \end{cases} & if \left|\frac{dVin}{dt}\right| t = 0 > SR \\ |Vin| * e^{\frac{-t\exp}{\tau}} & if \left|\frac{dVin}{dt}\right| t = 0 \leq SR \end{cases} \tag{3.14}$$

where $\mathcal{E}$ is defines the condition, Vin is the input voltage, *SR* is the slew rate, T_s is the sampling rate and tsl is non-linear settling time. When the authors examined each stage of the modulator, they discovered that the signal change rate increased with each successive integrator stage. The authors conclude that the ∑ΔM's signal that has the maximum rate of change is the signal at the quantizer's input or the last integrator's output. But the *SR* of first integrator is also very important to allow the signal to be as analogous as possible to the original signal and thus is as important as the last integrator to have the best overall *SR*. Therefore, for slew rate and finite bandwidth modeling the function must be included at each stage.

To model the saturation of the amplifiers, the Saturation block from Simulink is included after the integrator as shown in Fig. 3.7.

To model the integrator non-linearity, a user-defined block was used to implement the function

$$m = v + n * v^3 \tag{3.15}$$

where m is the output integrator non-linearity, v is the input value and n is the non-linearity coefficient. The user-defined block is shows in Fig. 3.8 before the first integrator. It was seen that the non-linearity at the first integrator has mostly effected the SNR but the effect is slowly getting negligible at the consequent stages. The non-ideality coefficient of value 0.01 was applied to the first integrator only. A finite gain of 5000, the saturation levels of $\pm$ 1.25, slew rate of 100 V/μsec, p-p jitter of 7.25 sec, RMS noise of 10 μV and finite bandwidth of 300 MHz were used. Fig. 3.8

shows the overall functional block implementation of a fourth-order CT ∑ΔM including all the non-idealities that was used in Simulink.

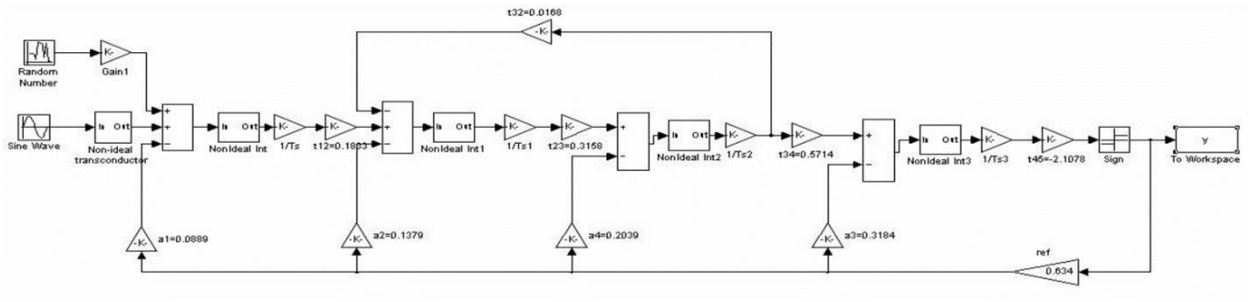
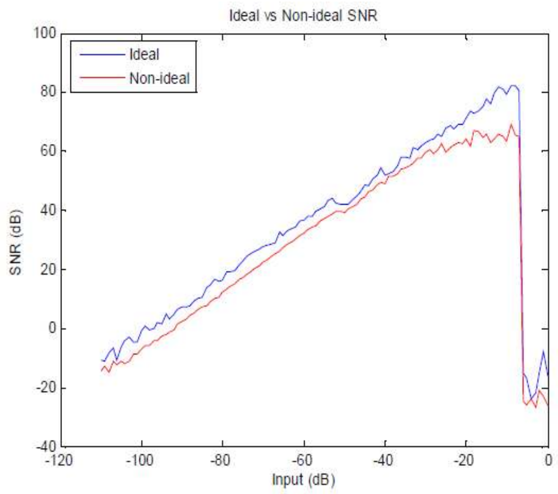
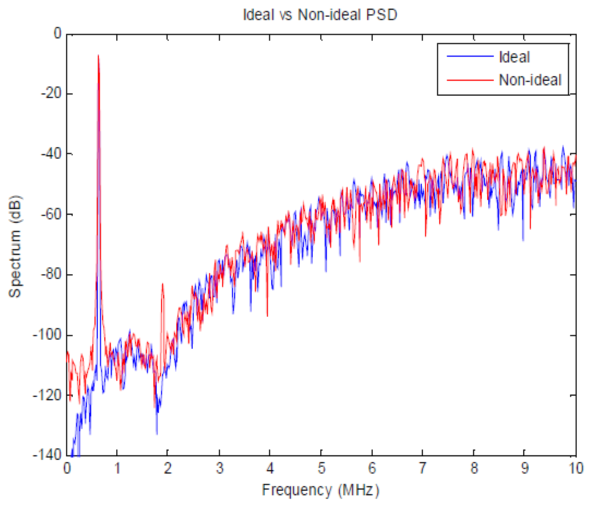


Figure 3.8: CT ∑ΔM model including all main non-idealities

The authors designed a single-bit fourth order CT ∑ΔM for Wide Band Code-Division-Multiple-Access (WCDMA) communications system that needed a SNR of at least 70 dB at an input signal bandwidth of 3.84 MHz. An oversampling ratio of 40 was selected and thus the sampling frequency was 152.6 MHz. The maximum input amplitude of 0.631 was chosen. The authors then compared the SNRs and PSDs of Simulink simulations using both ideal integrators and non-ideal integrators. Fig. 3.9 shows the plots comparing ideal and non-ideal integrators based on SNR and PSD of the CT ∑ΔM.



(a)



(b)

Figure 3.9: Comparison of ideal and non-ideal CT ∑ΔM based on a) SNR b) PSD

The authors noticed that the non-ideal block implementation influenced the SNR and PSD of the system since it included all the non-idealities. Thus, the authors conclude that the derived non-ideal CT ∑ΔM block specifications in Simulink can be used as inputs in the circuit level design. The paper also suggests an efficient way of viewing a circuit before fabricating and testing an actual CT ∑ΔM circuit.

Chapter 4

IMPLEMENTATION

CT ∑ΔMs are more difficult to design and simulate than DT ∑ΔMs because of the mixed-signal nature of the feedback loop which use both analog and digital circuits. A CT ∑ΔM's NTF and STF can be determined using various methods. In this thesis, the STFs and NTFs are designed using Chebyshev Type 2 filters. After determining a CT ∑ΔM's STF and NTF, the STF and NTF are implemented in a hardware architecture. In this chapter, the CT ∑ΔM architectures are represented by block diagrams. The subsequent block diagrams are then converted into difference equations, and using these different equations, the architecture's NTFs and STFs are determined. These STFs and NTFs are compared with the NTFs and STFs obtained from Chebyshev Type 2 filters and the coefficients for the CT ∑ΔM's architectures can be calculated.

Common hardware architectures for CT ∑ΔMs include cascade of resonators feedforward (CRFF), cascade of resonators feedback (CRFB), cascade of integrators feedforward (CIFF)and cascade of integrators feedback (CIFB) implementations. The feedback architectures feedback the modulator output to each integrator while the feedforward architectures feed a signal which is the sum of the input signal and all integrator outputs at the input of the quantizer. Single loop feedback architectures have more signal distortion than single loop feedforward architectures because the amplifier nonlinearities in single loop feedback architectures generate harmonic distortion which depends on the amplifier's input signal. However, feedforward architectures require more circuitry and thus require more power than feedback architectures.

In most ∑ΔM architectures, integrators are implemented using RC integrators or G_mC integrators. RC integrators have better linearity for larger output signal swings than G_mC integrators having similar specifications. The linearity of the G_mC integrators can be improved by adding additional linearization circuitry but the added circuitry adds phase in the feedback loop that can negatively affect the modulator’s stability. In this thesis, cascade of integrators feedback (CIFB) architecture is used for implementing CT ∑ΔMs and the CIFB architecture is implemented using *RC* integrators.

In this thesis, the CT ∑ΔM NTFs are designed as highpass Chebyshev Type 2 filters. After determining a NTF, a STF is designed as a low pass filter that uses the numerator of a lowpass Chebyshev Type 2 filter and denominator of the NTF. For this thesis, all CT ∑ΔMs have a sampling rate of 1GHz and a signal bandwidth of 20 MHz. All first order, second order, third order, fourth order and fifth order CT ∑ΔMs with their block diagrams are described as below.

4.1 First-Order lowpass CT ∑ΔM

Fig. 4.1 shows the block diagram of a first-order lowpass CT ∑ΔM implemented using CIFB architecture. As shown in Fig. 4.1, the first order lowpass CT ∑ΔM consists of a single integrator.

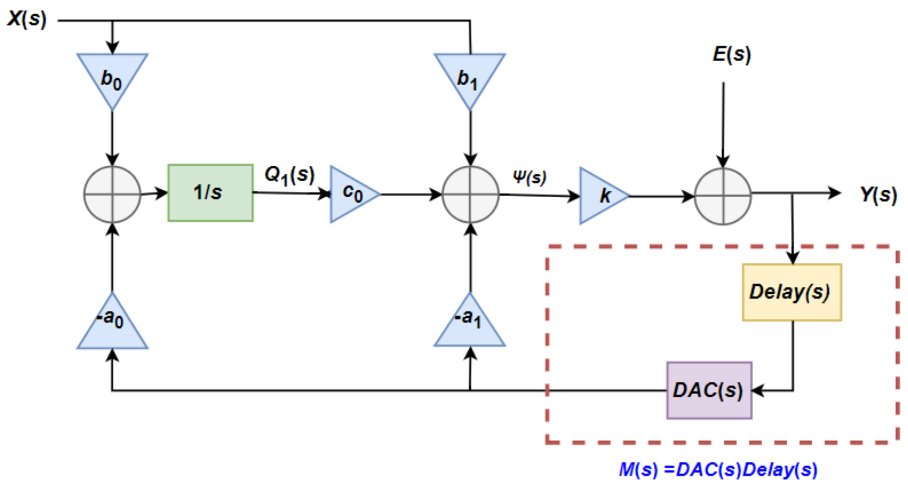


Figure 4.1: First-order lowpass CT ∑ΔM block diagram

By inspecting Fig. 4.1,

$$Y(s) = k\Psi(s) + E(s) \tag{4.1}$$

$$E(s) = Q[X(s)] - K\Psi(s) \tag{4.2}$$

and

$$M(s) = DAC(s)Delay(s) \tag{4.3}$$

where $X(s)$ is the modulator's input, $Y(s)$ is the modulator's output, $\Psi(s)$ is the input to the quantizer and $E(s)$ is the additive random error signal that represents the quantizer noise. Also from Fig. 4.1, the integrator's output, $Q_1(s)$, the quantizer's input, $\Psi(s)$, and the CT ∑ΔM's output, $Y(s)$, can be determined as

$$Q_1(s) = \{b_0X(s) - a_0M(s)Y(s)\}\frac{1}{s} \tag{4.4}$$

and

$$\Psi(s) = b_1X(s) - a_1M(s)Y(s) + c_0Q_1(s) \tag{4.5}$$

Substituting (4.4) into (4.5) and the resulting equation into (4.1), the STF and NTF of the first order lowpass CT ∑ΔM shown in Fig. 4.1 can be written as

$$NTF(s) = \frac{Y(s)}{E(s)} = \frac{k}{1 + a_1kM(s) + a_0c_0kM(s)\frac{1}{s}} = \frac{s}{(\frac{1}{k}+a_1M(s))s + a_0c_0M(s)} \tag{4.6}$$

and

$$STF(s) = \frac{Y(s)}{X(s)} = \frac{b_1k + b_0c_0k\frac{1}{s}}{1 + a_1kM(s) + a_0c_0kM(s)\frac{1}{s}} = \frac{b_1s + b_0c_0}{M(s)[(\frac{1}{k}+a_1)s + a_0c_0]} \tag{4.7}$$

To determine a desired NTF, a highpass Chebyshev Type 2 filter with a cut-off frequency near the CT ∑ΔM's 20 MHz bandwidth is designed. After determining the NTF, the STF is then designed as a lowpass filter using the numerator of a lowpass Chebyshev Type 2 filter and the NTF's denominator. For this thesis, a sampling rate of 1GHz and a signal bandwidth of 20MHz is used. The following MATLAB code shows an example of the design of such first-order lowpass CT ∑ΔM STF and NTF.

```
[NTFnum, NTFden] = cheby2(1, 16, 2*pi*22e6, 'high', 's'); % NTF
NTF = tf(NTFnum, NTFden);
[STFnum, STFden] = cheby2(1, 20, 2*pi*850e6, 's'); % STF
STFnum = STFnum/STFnum(end) * NTFden(end);
STF = tf(STFnum, NTFden);
```

The above code produces

$$NTF(s) = \frac{s}{s + 8.66e08} \tag{4.8}$$

and

$$STF(s) = \frac{8.66e08}{s + 8.66e08} \tag{4.9}$$

Equations (4.6) and (4.7) are compared with (4.8) and (4.9), respectively, to calculate the coefficients a_0, a_1, b_0, b_1, and, c_0 in Fig 4.1. In this thesis, k is set to unity. Fig 4.2 shows the magnitude response of the NTF in (4.8) and the STF in (4.9).

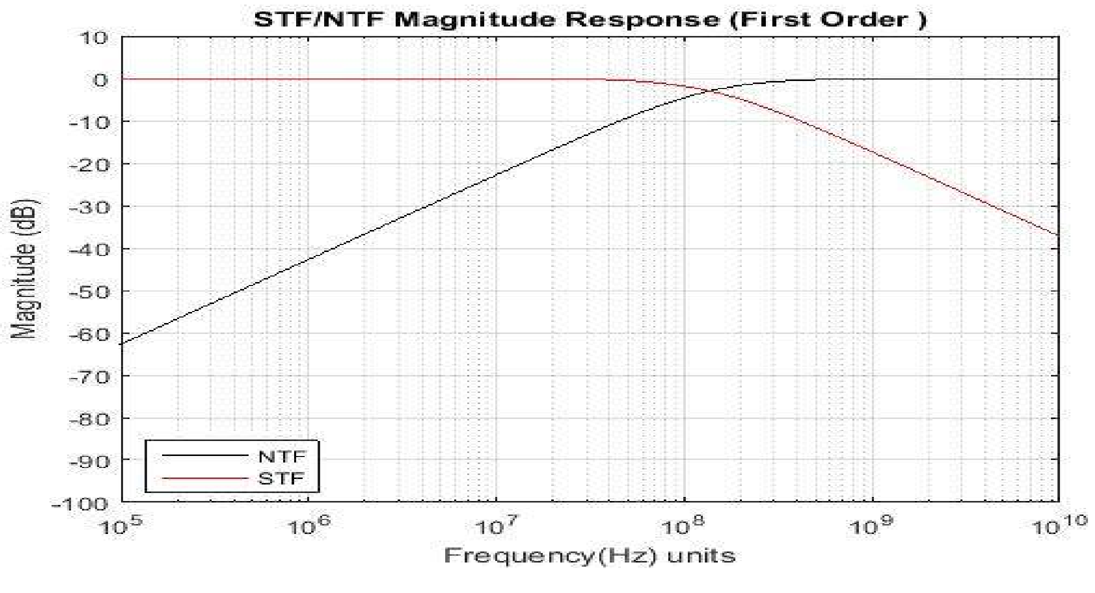


Figure 4.2: STF/NTF magnitude response of first-order lowpass CT ∑ΔM

Fig 4.2 shows that for frequencies below 20MHz, the magnitude response of the NTF is almost much less than one and hence the quantization noise will be attenuated for frequencies below 20 MHz. Fig 4.2also shows that the magnitude response of the STF is approximately one for frequencies below 20 MHz which implies that signal frequencies are passed within the signal's bandwidth without significant attenuation.

4.2 Second-Order lowpass CT ∑ΔM

Fig. 4.3 shows the block diagram of a CIFB implementation of a second-order lowpass CT ∑ΔM. As shown in Fig. 4.3, a second-order lowpass CT ∑ΔM has two integrators.

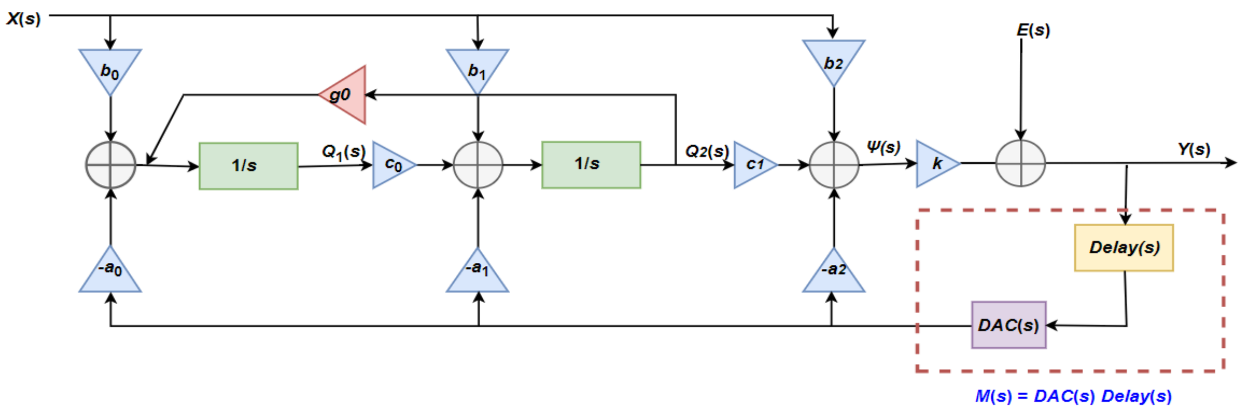


Figure 4.3: Second-order lowpass CT ∑ΔM block diagram

In Fig. 4.3, $X(s)$ is the modulator's input, $Y(s)$ is the modulator's output, $Q_1(s)$ is the first integrator's output, $Q_2(s)$ is the second integrator's output, $\Psi(s)$ is the input to the quantizer and $E(s)$ is the additive random error signal that represents the quantizer noise. From Fig. 4.3, $Q_1(s)$, $Q_2(s)$, $\Psi(s)$ and, $Y(s)$ can be determined as

$$Y(s) = k\Psi(s) + E(s) \tag{4.10}$$

$$Q_1(s) = \{b_0X(s) - a_0M(s)Y(s) + g_0Q_2(s)\}\frac{1}{s} \tag{4.11}$$

$$Q_2(s) = \{b_1X(s) - a_1M(s)Y(s) + c_0Q_1(s)\}\frac{1}{s} \tag{4.12}$$

$$\Psi(s) = b_2X(s) - a_2M(s)Y(s) + c_1Q_2(s) \tag{4.13}$$

Substituting (4.11) into (4.12), solving for $Q_2(s)$, substituting this expression into (4.13) and the resulting expression into (4.10), the STF and NTF of the second order lowpass CT ∑ΔM in Fig. 4.3 can be written as

$$NTF(s) = \frac{Y(s)}{E(s)} = \frac{s^2 - g_0c_0}{\left(1 + a_2kM(s)\right)s^2 + a_1c_1kM(s)s + (a_0c_0c_1 - g_0a_2c_0)kM(s) - g_0c_0} \tag{4.14}$$

and

$$STF(s) = \frac{Y(s)}{X(s)} = \frac{k(\, b_2s^2 + b_1c_1s + b_0c_0c_1 - g_0b_2c_0)}{\left(1 + a_2kM(s)\right)s^2 + a_1c_1kM(s)s + (a_0c_0c_1 - g_0a_2c_0)kM(s) - g_0c_0} \tag{4.15}$$

To determine a desired NTF, a highpass Chebyshev Type 2 filter with a cut-off frequency near the CT ∑ΔM's 20 MHz bandwidth is designed. After determining the NTF, the STF is designed as a lowpass filter using the numerator of a lowpass Chebyshev Type 2 filter and the poles of the NTF as the denominator. The following MATLAB code shows an example of the design of such a second-order lowpass CT ∑ΔM.

```
[NTFnum, NTFden] = cheby2(2, 37, 2*pi*22e6, 'high', 's'); % NTF
NTF = tf(NTFnum, NTFden);
[STFnum, STFden] = cheby2(2, 40, 2*pi*750e6, 's'); % STF
STFnum = STFnum/STFnum(end) * NTFden(end);
STF = tf(STFnum, NTFden);
```

The above code produces

$$NTF(s) = \frac{\mathrm{s}^2 - 2.384\mathrm{e}{-07}\ \mathrm{s} + 9.554\mathrm{e}15}{\mathrm{s}^2 + 1.155\mathrm{e}09\ \mathrm{s} + 6.764\mathrm{e}17} \tag{4.16}$$

and

$$STF(s) = \frac{0.01523\mathrm{s}^2 + 6.764\mathrm{e}17}{\mathrm{s}^2 + 1.155\mathrm{e}09\ \mathrm{s} + 6.764\mathrm{e}17} \tag{4.17}$$

Equations (4.14) and (4.15) are compared with (4.16) and (4.17), respectively, to calculate the CT ∑ΔM's coefficients $a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1$, and, g_0 in Fig. 4.3. Fig 4.4 shows the magnitude response of the NTF in (4.16) and the STF in (4.17).

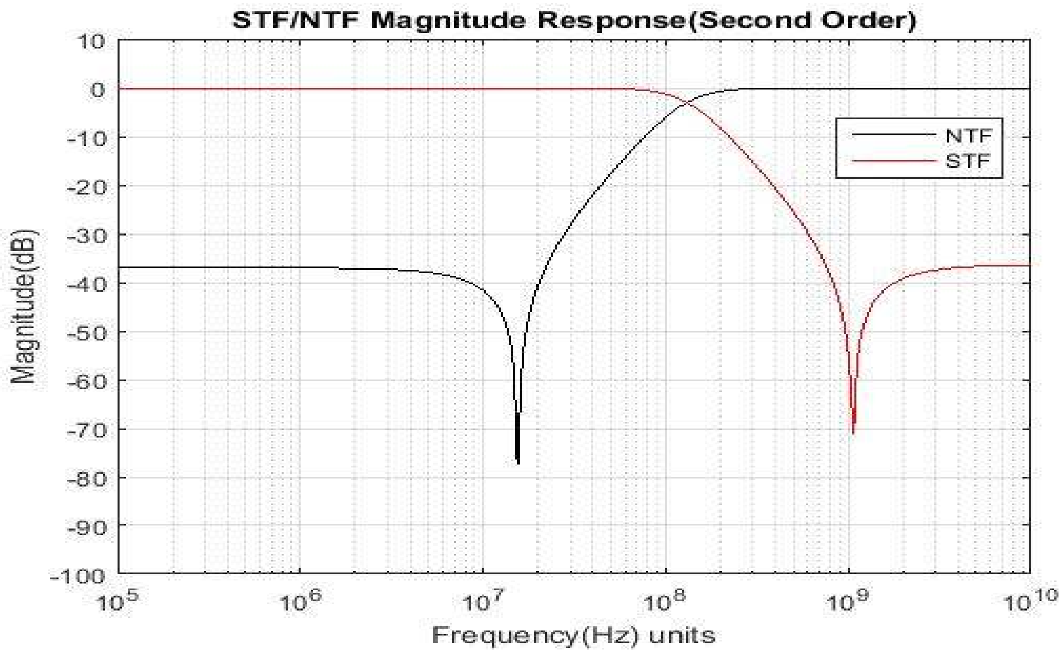


Figure 4.4: STF/NTF magnitude response of second-order lowpass CT ∑ΔM

Fig. 4.4 shows that for frequencies below 20 MHz, the magnitude response of the second-order NTF is much less than one and hence the quantization noise is attenuated at frequencies below 20 MHz. Fig. 4.4 also shows that the magnitude response of the STF is approximately one for frequencies below 20 MHz which implies that the signal frequencies are passed within the signal's bandwidth without significant attenuation. The dips in the magnitsude responses are due to the presence of a zero in the STF and NTF transfer functions.

4.3 Third-Order lowpass CT ∑ΔM

Fig. 4.5 shows the block diagram of a CIFB implementation of a third-order lowpass CT ∑ΔM. As shown in Fig. 4.5, a third-order lowpass CT ∑ΔM has three integrators.

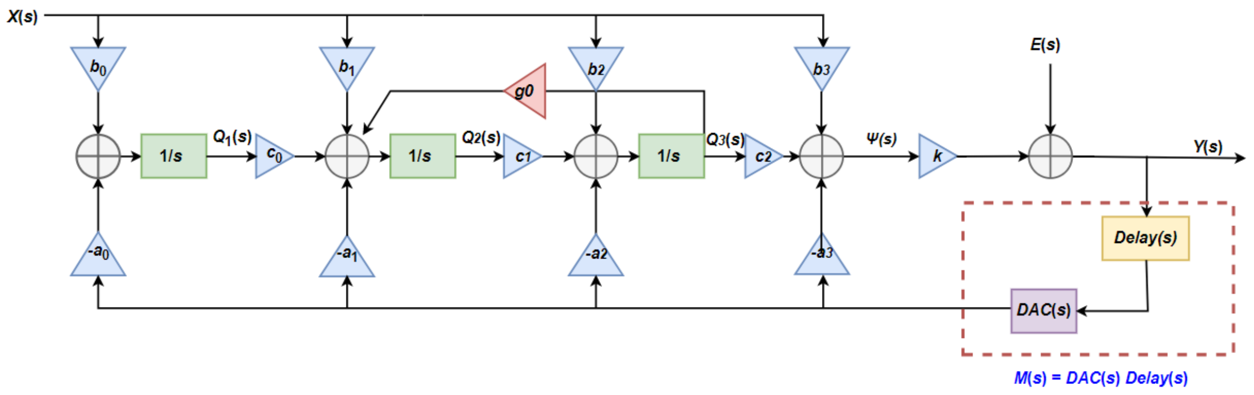


Figure 4.5: Third-order lowpass CT ∑ΔM block diagram

In Fig. 4.5, $X(s)$ is the modulator's input, $Y(s)$ is the modulator's output, $Q_1(s)$ is the output of the first integrator, $Q_2(s)$ is the output of the second integrator, $Q_3(s)$ is the output of the third integrator, $\Psi(s)$ is the input to the quantizer and $E(s)$ is the additive random error signal that represents the quantization noise. From Fig. 4.5, $Q_1(s)$, $Q_2(s)$, $Q_3(s)$, $\Psi(s)$, and, $Y(s)$ can be determined as

$$Y(s) = k\Psi(s) + E(s) \tag{4.18}$$

$$Q_1(s) = \{b_0X(s) - a_0M(s)Y(s)\}\frac{1}{s} \tag{4.19}$$

$$Q_2(s) = \{b_1X(s) - a_1M(s)Y(s) + c_0Q_1(s) + g_0Q_3(s)\}\frac{1}{s} \tag{4.20}$$

$$Q_3(s) = \{b_2X(s) - a_2M(s)Y(s) + c_1Q_2(s)\}\frac{1}{s} \tag{4.21}$$

$$\Psi(s) = b_3X(s) - a_3M(s)Y(s) + c_2Q_3(s) \tag{4.22}$$

Substituting (4.19) into (4.20), (4.20) into (4.21) and solving for $Q_3(s)$, substituting this expression into (4.22) and the resulting expression into (4.18), the STF and NTF of the third-order lowpass CT ∑ΔM in Fig. 4.5 can be written as

$$NTF(s) = \frac{s(s^2 - g_0c_1)}{(1 + a_3kM(s))s^3 + a_2c_2kM(s)s^2 + ((a_1c_1c_2 - g_0a_3c_1)kM(s) - g_0c_1)s - a_0c_0c_1c_2KM(s)} \tag{4.23}$$

and

$$STF(s) = \frac{k(b_3 s^3 + b_2 c_2 s^2 + (b_1 c_1 c_2 - g_0 b_3 c_1)s + b_0 c_0 c_1 c_2)}{(1 + a_3 kM(s))s^3 + a_2 c_2 kM(s)s^2 + ((a_1 c_1 c_2 - g_0 a_3 c_1)kM(s) - g_0 c_1)s - a_0 c_0 c_1 c_2 KM(s)} \tag{4.24}$$

The NTF and STF of such a third-order CT ∑ΔM can be designed using a design method similar to the ones used for the first and second order CT ∑ΔMs. The following MATLAB code shows an example of the design of such a third-order lowpass CT ∑ΔM that has a bandwidth of 20 MHz and a sampling rate of 1 GHz.

```
[NTFnum, NTFden] = cheby2(3, 60, 2*pi*22e6, 'high', 's'); % NTF
 NTF = tf(NTFnum, NTFden;
[STFnum, STFden] = cheby2(3, 62, 2*pi*850e6, 's'); % STF
STFnum = STFnum/STFnum(end) * NTFden(end);
STF =tf(STFnum, NTFden);
```

The above code produces

$$NTF(s) = \frac{s^3 + 2.126e{-}07\, s^2 + 1.433e16\, s + 1.147e11}{s^3 + 1.731e09\, s^2 + 1.512e18\, s + 6.603e26} \tag{4.25}$$

and

$$STF(s) = \frac{1.736e07\, s^2 + 6.603e26}{s^3 + 1.731e09\, s^2 + 1.512e18\, s + 6.603e26} \tag{4.26}$$

Equations (4.23) and (4.24) are compared with (4.25) and (4.26), respectively, to calculate the coefficients $a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3, c_0, c_1, c_2, g_0,$ and, g_1 in Fig. 4.5. Fig 4.6 shows the magnitude response of the STF and NTF of third-order lowpass CT ∑ΔM given in (4.25) and (4.26) respectively.

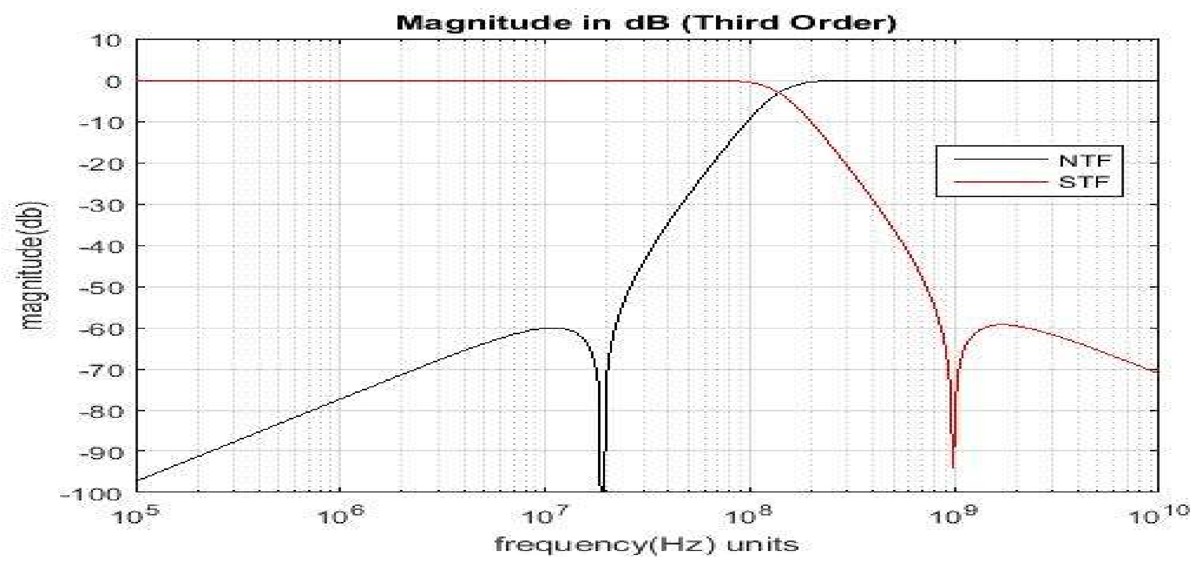


Figure 4.6: STF/NTF magnitude response of third-order lowpass CT ∑ΔM

4.4 Fourth-Order lowpass CT ∑ΔM

Fig. 4.7 shows the block diagram of a CIFB implementation of a fourth-order lowpass CT ∑ΔM. As shown in Fig. 4.7, a fourth-order lowpass CT ∑ΔM has four integrators.

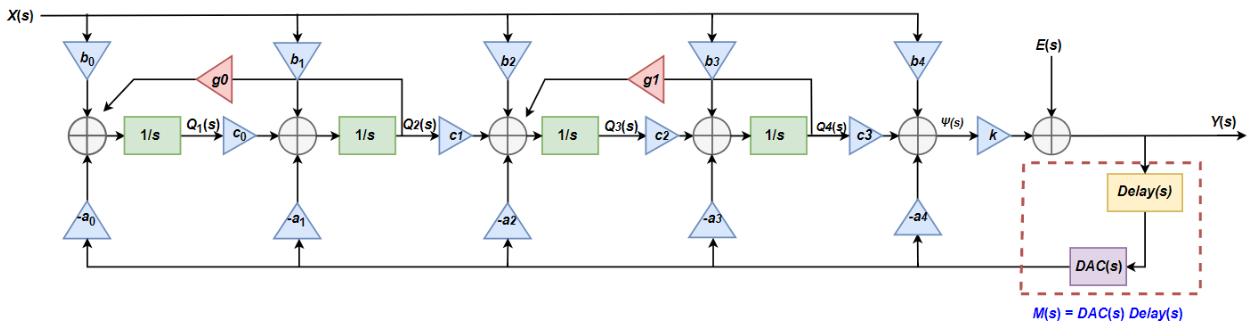


Figure 4.7: Fourth-order lowpass CT ∑ΔM block diagram

In Fig. 4.7, $X(s)$ is the modulator's input, $Y(s)$ is modulator's output, $Q_1(s)$ is the output of the first integrator, $Q_2(s)$ is the output of the second integrator, $Q_3(s)$ is the output of the third integrator, $Q_4(s)$ is the output of the fourth integrator, $\Psi(s)$ is the input to the quantizer and $E(s)$ is the additive random error signal that represents the quantization noise. From Fig. 4.7, $Q_1(s)$, $Q_2(s), Q_3(s)$, $Q_4(s)$, $\Psi(s)$, and, $Y(s)$ can be determined as

$$Y(s) = k\Psi(s) + E(s) \tag{4.27}$$

$$Q_1(s) = \{b_0X(s) - a_0M(s)Y(s) + g_0Q_2(s)\}\frac{1}{s} \tag{4.28}$$

$$Q_2(s) = \{b_1X(s) - a_1M(s)Y(s) + c_0Q_1(s)\}\frac{1}{s} \tag{4.29}$$

$$Q_3(s) = \{b_2X(s) - a_2M(s)Y(s) + c_1Q_2(s) + g_1Q_4(s)\}\frac{1}{s} \tag{4.30}$$

$$Q_4(s) = \{b_3X(s) - a_3M(s)Y(s) + c_2Q_3(s)\}\frac{1}{s} \tag{4.31}$$

$$\Psi(s) = b_4X(s) - a_4M(s)Y(s) + c_3Q_4(s) \tag{4.32}$$

Substituting (4.28) into (4.29), (4.29) into (4.30), (4.30) into (4.31) and solving for $Q_4(s)$, substituting this expression into (4.32) and the resulting expression into (4.27), the STF and NTF of the fourth order lowpass CT ∑ΔM in Fig. 4.7 can be written as

$$NTF(s) = \frac{(s^2 - g_0c_0)(s^2 - g_1c_2)}{\begin{array}{c}(1 + a_4kM(s))s^4 + a_3c_3kM(s)s^3 + \{(a_2c_2c_3 - g_0a_4c_0 - g_1a_4c_2)kM(s) - (g_0c_0 + g_1c_2)\}s^2 + kM(s)(a_1c_1c_2c_3 - g_0a_3c_0c_3)s + \\ (a_0c_0c_1c_2c_3 + g_0a_2c_0c_2c_3 + g_0g_1b_4c_0c_2)kM(s) + g_0g_1c_0c_1c_2c_3\end{array}} \tag{4.33}$$

and

$$STF(s) = \frac{k\{(b_4s^4 + b_3c_3s^3 + (b_2c_2c_3 - g_0b_4c_0 - g_1b_4c_2)s^2 + (b_1c_1c_2c_3 - g_0b_3c_0c_3)s + (b_0c_0c_1c_2c_3 - g_0b_2c_0c_2c_3 + g_0g_1b_4c_0c_1c_2c_3))\}}{\begin{array}{c}(1 + a_4kM(s))s^4 + a_3c_3kM(s)s^3 + \{(a_2c_2c_3 - g_0a_4c_0 - g_1a_4c_2)kM(s) - (g_0c_0 + g_1c_2)\}s^2 + kM(s)(a_1c_1c_2c_3 - g_0a_3c_0c_3)s + \\ (a_0c_0c_1c_2c_3 + g_0a_2c_0c_2c_3 + g_0g_1b_4c_0c_2)kM(s) + g_0g_1c_0c_1c_2c_3\end{array}} \tag{4.34}$$

The NTF and STF of such a fourth-order CT ∑ΔM can be designed using a design method similar to the ones used for the first, second and third-order CT ∑ΔMs. The following MATLAB code shows an example of the design of such a fourth-order lowpass CT ∑ΔM that has a bandwidth of 20 MHz and a sampling rate of 1 GHz.

```
[NTFnum, NTFden] = cheby2(4, 85, 2*pi*22e6, 'high', 's');   % NTF

NTF = tf(NTFnum, NTFden);

[STFnum, STFden] = cheby2(4, 75, 2*pi*950e6, 's');    % STF

STFnum = STFnum/STFnum(end) * NTFden(end);

STF =tf(STFnum, NTFden);
```

The above code produces

$$NTF(s) = \frac{\mathrm{s}^4 - 4.091\mathrm{e}{-07}\,\mathrm{s}^3 + 1.911\mathrm{e}16\,\mathrm{s}^2 - 4.727\mathrm{e}11\,\mathrm{s} + 4.564\mathrm{e}31}{\mathrm{s}^4 + 2.467\mathrm{e}09\,\mathrm{s}^3 + 3.062\mathrm{e}18\,\mathrm{s}^2 + 2.229\mathrm{e}27\,\mathrm{s} + 8.116\mathrm{e}35} \tag{4.35}$$

and

$$STF(s) = \frac{7.991\mathrm{e}{-05}\,\mathrm{s}^4 - 5.035\mathrm{e}{-11}\,\mathrm{s}^3 + 2.278\mathrm{e}16\,\mathrm{s}^2 - 9.782\mathrm{e}09\,\mathrm{s} + 8.116\mathrm{e}35}{\mathrm{s}^4 + 2.467\mathrm{e}09\,\mathrm{s}^3 + 3.062\mathrm{e}18\,\mathrm{s}^2 + 2.229\mathrm{e}27\,\mathrm{s} + 8.116\mathrm{e}35} \tag{4.36}$$

Equations (4.33) and (4.34) are compared with (4.35) and (4.36), respectively, to calculate the coefficients $a_0, a_1, a_2, a_3, a_4, b_0, b_1, b_2, b_3, b_4, c_0, c_1, c_3, g_0$, and g_1 given in Fig. 4.7. Fig 4.8 shows the magnitude response of the STF and NTF of the fourth-order lowpass CT ∑ΔM given in (4.35) and (4.36) respectively.

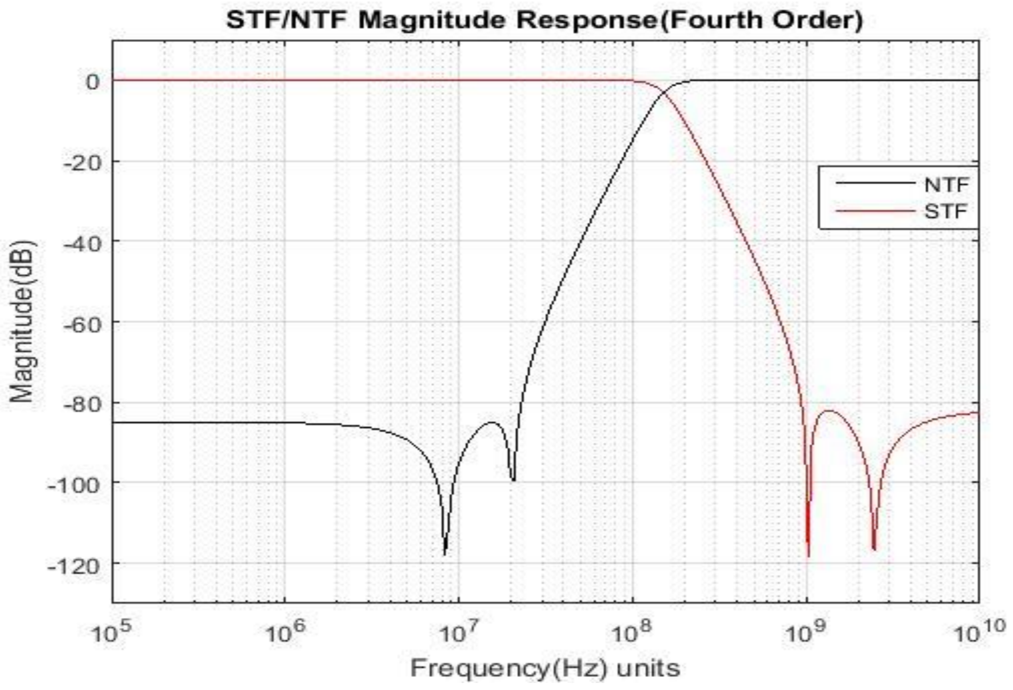


Figure 4.8: STF/NTF magnitude response of fourth-order lowpass CT ∑ΔM

The two dips in each STF and NTF's magnitude response are due to the presence of two zeros in both STF's and NTF's transfer function.

4.5 Fifth-Order lowpass CT ∑ΔM

Fig 4.9 shows the block diagram of a general fifth-order lowpass CT ∑ΔM. The fifth-order lowpass CT ∑ΔM consists of five integrators.

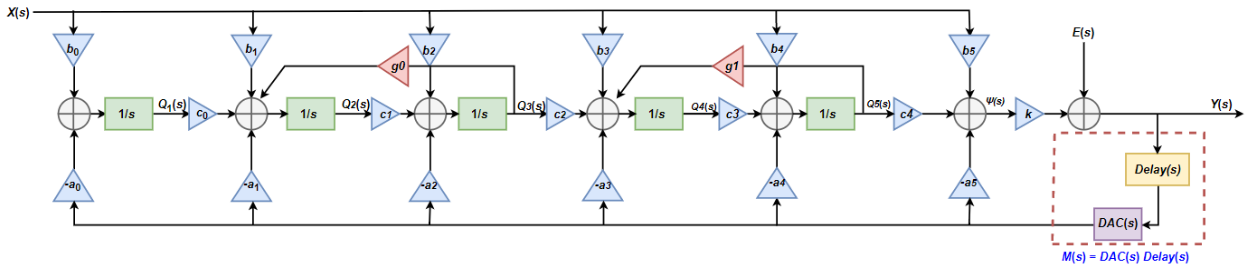


Figure 4.9: Fifth-order lowpass CT ∑ΔM block diagram

In Fig. 4.9, $X(s)$ is the modulator's input, $Y(s)$ is the modulator's output, $Q_1(s)$ is first integrator's output, $Q_2(s)$ is the second integrator's output, $Q_3(s)$ is the third integrator's output, $Q_4(s)$ is the fourth integrator's output, $Q_5(s)$ is the fifth integrator's output, $\Psi(s)$ is the input to the quantizer and $E(s)$ is the additive random error signal that represents the quantization noise. From Fig. 4.9, $Q_1(s)$, $Q_2(s)$, $Q_3(s)$, $Q_4(s)$, $Q_5(s)$, $\Psi(s)$, and, $Y(s)$ can be determined as

$$Y(s) = k\Psi(s) + E(s) \tag{4.37}$$

$$Q_1(s) = \{b_0X(s) - a_0M(s)Y(s)\}\ \frac{1}{s} \tag{4.38}$$

$$Q_2(s) = \{b_1X(s) - a_1M(s)Y(s) + c_0Q_1(s) + g_0Q_3(s)\}\frac{1}{s} \tag{4.39}$$

$$Q_3(s) = \{b_2X(s) - a_2M(s)Y(s) + c_1Q_2(s)\}\ \frac{1}{s} \tag{4.40}$$

$$Q_4(s) = \ \{b_3X(s) - a_3M(s)Y(s) + c_2Q_3(s) + g_1Q_5(s)\}\frac{1}{s} \tag{4.41}$$

$$Q_5(s) = \{b_4X(s) - a_4M(s)Y(s) + c_3Q_4(s)\}\frac{1}{s} \tag{4.42}$$

$$\Psi(s) = b_5X(s) - a_5M(s)Y(s) + c_4Q_5(s) \tag{4.43}$$

Substituting (4.38) into (4.39), (4.39) into (4.40), (4.40) into (4.41), (4.41) into (4.42) and solving for $Q_5(s)$, substituting this expression into (4.43) and the resulting expression into (4.37), the STF and NTF of the fifth-order lowpass CT ∑ΔM in Fig. 4.9 can be written as

$NTF(s) =$

$$\frac{s(s^2 - g_0c_1)(s^2 - g_1c_2)}{\big(1 + a_5kM(s)\big)s^5 + a_4c_4kM(s)s^4 + \{(a_3c_3c_4 - g_0a_5c_1 - g_1a_5c_5)kM(s) - (g_0c_1 + g_1c_3)\}s^3 + kM(s)(a_2c_2c_3c_4 - g_0a_4c_1c_4)s^2 + \{kM(s)(a_1c_1c_2c_3c_4 - g_0a_3c_1c_3c_4 + g_0g_1a_5c_1c_3c_4) + g_0g_1a_5c_1c_2c_3c_4\}s + kM(s)a_0c_0c_1c_2c_3c_4} \tag{4.44}$$

and

$STF(s) =$

$$\frac{k\{\big(b_5s^5 + b_4c_4s^4 + (b_3c_3c_4 - g_0b_5c_1 - g_1b_5c_5\big)s^3 + (b_2c_2c_3c_4 - g_0b_4c_1c_4)s^2 + (b_1c_1c_2c_3c_4 - g_0b_3c_1c_3c_4 + g_0g_1b_5c_1c_3)s + b_0c_0c_1c_2c_3c_4\}}{\big(1 + a_5kM(s)\big)s^5 + a_4c_4kM(s)s^4 + \{(a_3c_3c_4 - g_0a_5c_1 - g_1a_5c_5)kM(s) - (g_0c_1 + g_1c_3)\}s^3 + kM(s)(a_2c_2c_3c_4 - g_0a_4c_1c_4)s^2 + \{kM(s)(a_1c_1c_2c_3c_4 - g_0a_3c_1c_3c_4 + g_0g_1a_5c_1c_3c_4) + g_0g_1a_5c_1c_2c_3c_4\}s + kM(s)a_0c_0c_1c_2c_3c_4} \tag{4.45}$$

Similarly, the following MATLAB code shows the example of such a fifth-order lowpass CT ∑ΔM that has a bandwidth of 20 MHz and a sampling rate of 1 GHz.

```
[NTFnum, NTFden] = cheby2(5, 107, 2*pi*22e6,'high', 's');   % NTF
NTF = tf(NTFnum, NTFden);
[STFnum, STFden] = cheby2(5, 115, 2*pi*950e6, 's');   % STF
STFnum = STFnum/STFnum(end) * NTFden(end);
STF =tf(STFnum, NTFden);
```

The above MATLAB code produces

$$NTF(s) = \frac{s^5 + 7.332e{-}07\ s^4 + 2.388e16\ s^3 + 1.444e12\ s^2 + 1.141e32\ s + 2.088e29}{s^5 + 3.002e09\ s^4 + 4.53e18\ s^3 + 4.232e27\ s^2 + 2.445e36\ s + 7.061e44} \tag{4.46}$$

and

$$STF(s) = \frac{1.738e05\ s^4 + 2.477e25\ s^2 + 7.061e44}{s^5 + 3.002e09\ s^4 + 4.53e18\ s^3 + 4.232e27\ s^2 + 2.445e36\ s + 7.061e44} \qquad (4.47)$$

Equations (4.44) and (4.45) are compared with (4.46) and (4.47), respectively, to calculate the coefficients $a_0, a_1, a_2, a_3, a_4, a_5, b_0, b_1, b_2, b_3, b_4, b_5, c_0, c_1, c_3, c_4, g_0$, and g_1 given in Fig. 4.9. Fig 4.10 shows the magnitude response of the STF and NTF of fifth-order lowpass CT ∑ΔM given in (4.46) and (4.47) respectively.

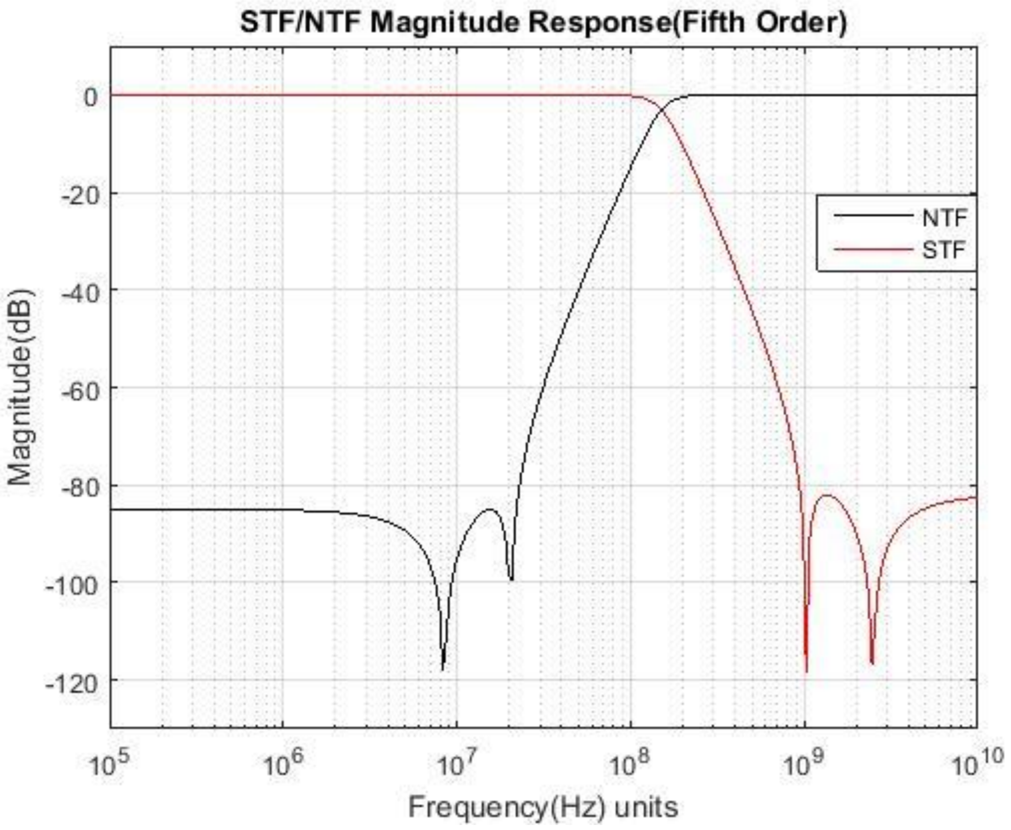


Figure 4.10: STF/NTF magnitude response of fifth-order lowpass CT ∑ΔM

4.6 SUMMARY

Table 4.1 summarizes the STFs and NTFs of CIFB implementations of first, second, third, fourth and fifth order CT ∑ΔMs. Table 4.2 summarizes the desired STF and NTF transfer functions of first, second, third, fourth and fifth order CT ∑ΔMs used in this thesis.

Order		Implementation
1st	STF	$\frac{b_1 s + b_0 c_0}{M(s)[(\frac{1}{k} + a_1)s + a_0 c_0]}$
	NTF	$\frac{s}{(\frac{1}{k} + a_1 M(s))s + a_0 c_0 M(s)}$
2nd	STF	$\frac{k(b_2 s^2 + b_1 c_1 s + b_0 c_0 c_1 - g_0 b_2 c_0)}{\left(1 + a_2 kM(s)\right)s^2 + a_1 c_1 kM(s)s + (a_0 c_0 c_1 - g_0 a_2 c_0)kM(s) - g_0 c_0}$
	NTF	$\frac{s^2 - g_0 c_0}{\left(1 + a_2 kM(s)\right)s^2 + a_1 c_1 kM(s)s + (a_0 c_0 c_1 - g_0 a_2 c_0)kM(s) - g_0 c_0}$
3rd	STF	$\frac{k(b_3 s^3 + b_2 c_2 s^2 + (b_1 c_1 c_2 - g_0 b_3 c_1)s + b_0 c_0 c_1 c_2)}{\left(1 + a_3 kM(s)\right)s^3 + a_2 c_2 kM(s)s^2 + ((a_1 c_1 c_2 - g_0 a_3 c_1)kM(s) - g_0 c_1)s - a_0 c_0 c_1 c_2 KM(s)}$
	NTF	$\frac{s(s^2 - g_0 c_1)}{\left(1 + a_3 kM(s)\right)s^3 + a_2 c_2 kM(s)s^2 + ((a_1 c_1 c_2 - g_0 a_3 c_1)kM(s) - g_0 c_1)s - a_0 c_0 c_1 c_2 KM(s)}$
4th	STF	$\frac{k\{(b_4 s^4 + b_3 c_3 s^3 + (b_2 c_2 c_3 - g_0 b_4 c_0 - g_1 b_4 c_2)s^2 + (b_1 c_1 c_2 c_3 - g_0 b_3 c_0 c_3)s + (b_0 c_0 c_1 c_2 c_3 - g_0 b_2 c_0 c_2 c_3 + g_0 g_1 b_4 c_0 c_1 c_2 c_3)\}}{\left(1 + a_4 kM(s)\right)s^4 + a_3 c_3 kM(s)s^3 + \{(a_2 c_2 c_3 - g_0 a_4 c_0 - g_1 a_4 c_2)kM(s) - (g_0 c_0 + g_1 c_2)\}s^2 + kM(s)(a_1 c_1 c_2 c_3 - g_0 a_3 c_0 c_3)s + (a_0 c_0 c_1 c_2 c_3 + g_0 a_2 c_0 c_2 c_3 + g_0 g_1 b_4 c_0 c_2)kM(s) + g_0 g_1 c_0 c_1 c_2 c_3}$
	NTF	$\frac{(s^2 - g_0 c_0)(s^2 - g_1 c_2)}{\left(1 + a_4 kM(s)\right)s^4 + a_3 c_3 kM(s)s^3 + \{(a_2 c_2 c_3 - g_0 a_4 c_0 - g_1 a_4 c_2)kM(s) - (g_0 c_0 + g_1 c_2)\}s^2 + kM(s)(a_1 c_1 c_2 c_3 - g_0 a_3 c_0 c_3)s + (a_0 c_0 c_1 c_2 c_3 + g_0 a_2 c_0 c_2 c_3 + g_0 g_1 b_4 c_0 c_2)kM(s) + g_0 g_1 c_0 c_1 c_2 c_3}$
5th	STF	$\frac{k\{(b_5 s^5 + b_4 c_4 s^4 + (b_3 c_3 c_4 - g_0 b_5 c_1 - g_1 b_5 c_5)s^3 + (b_2 c_2 c_3 c_4 - g_0 b_4 c_1 c_4)s^2 + (b_1 c_1 c_2 c_3 c_4 - g_0 b_3 c_1 c_3 c_4 + g_0 g_1 b_5 c_1 c_3)s + b_0 c_0 c_1 c_2 c_3 c_4\}}{\left(1 + a_5 kM(s)\right)s^5 + a_4 c_4 kM(s)s^4 + \{(a_3 c_3 c_4 - g_0 a_5 c_1 - g_1 a_5 c_5)kM(s) - (g_0 c_1 + g_1 c_3)\}s^3 + kM(s)(a_2 c_2 c_3 c_4 - g_0 a_4 c_1 c_4)s^2 + \{ kM(s)(a_1 c_1 c_2 c_3 c_4 - g_0 a_3 c_1 c_3 c_4 + g_0 g_1 a_5 c_1 c_3 c_4) + g_0 g_1 a_5 c_1 c_2 c_3 c_4\}s + kM(s)a_0 c_0 c_1 c_2 c_3 c_4}$
	NTF	$\frac{s(s^2 - g_0 c_1)(s^2 - g_1 c_2)}{\left(1 + a_5 kM(s)\right)s^5 + a_4 c_4 kM(s)s^4 + \{(a_3 c_3 c_4 - g_0 a_5 c_1 - g_1 a_5 c_5)kM(s) - (g_0 c_1 + g_1 c_3)\}s^3 + kM(s)(a_2 c_2 c_3 c_4 - g_0 a_4 c_1 c_4)s^2 + \{ kM(s)(a_1 c_1 c_2 c_3 c_4 - g_0 a_3 c_1 c_3 c_4 + g_0 g_1 a_5 c_1 c_3 c_4) + g_0 g_1 a_5 c_1 c_2 c_3 c_4\}s + kM(s)a_0 c_0 c_1 c_2 c_3 c_4}$

Table 4.1: (First, Second, Third, Fourth and Fifth) Order CT ∑ΔM STF and NTF

Order		Calculated STFs and NTFs
1st	STF	$\frac{8.66e08}{s + 8.66e08}$
	NTF	$\frac{s}{s + 8.66e08}$
2nd	STF	$\frac{0.01523\ s^2 + 6.764e17}{s^2 + 1.155e09\ s + 6.764e17}$
	NTF	$\frac{s^2 - 2.384e-07\ s + 9.554e15}{s^2 + 1.155e09\ s + 6.764e17}$
3rd	STF	$\frac{1.736e07\ s^2 + 6.603e26}{s^3 + 1.731e09\ s^2 + 1.512e18\ s + 6.603e26}$
	NTF	$\frac{s^3 + 2.126e-07\ s^2 + 1.433e16\ s + 1.147e11}{s^3 + 1.731e09\ s^2 + 1.512e18\ s + 6.603e26}$
4th	STF	$\frac{7.991e-05\ s^4 - 5.035e-11\ s^3 + 2.278e16\ s^2 - 9.782e09\ s + 8.116e35}{s^4 + 2.467e09\ s^3 + 3.062e18\ s^2 + 2.229e27\ s + 8.116e35}$
	NTF	$\frac{s^4 - 4.091e-07\ s^3 + 1.911e16\ s^2 - 4.727e11\ s + 4.564e31}{s^4 + 2.467e09\ s^3 + 3.062e18\ s^2 + 2.229e27\ s + 8.116e35}$
5th	STF	$\frac{1.738e05\ s^4 + 2.477e25\ s^2 + 7.061e44}{s^5 + 3.002e09\ s^4 + 4.53e18\ s^3 + 4.232e27\ s^2 + 2.445e36\ s + 7.061e44}$
	NTF	$\frac{s^5 + 7.332e-07\ s^4 + 2.388e16\ s^3 + 1.444e12\ s^2 + 1.141e32\ s + 2.088e29}{s^5 + 3.002e09\ s^4 + 4.53e18\ s^3 + 4.232e27\ s^2 + 2.445e36\ s + 7.061e44}$

Table 4.2: (First, Second, Third, Fourth and Fifth) Order CT ∑ΔM Simulated STFs and NTFs

Chapter 5

COMPARISION OF THE SIMULATION METHODS

DT ∑ΔMs can be accurately modeled using difference equations because DT ∑ΔMs are simply made up of gains and delays; however, CT ∑ΔMs are more difficult to design and simulate than DT ∑ΔMs because of the mixed-signal nature of CT ∑ΔMs which use both analog and digital circuits in their loop filters. CT ∑ΔMs can be simulated using various approaches such as using MATLAB/Simulink, numerical integration methods such as the delta transform, SPICE modeling and solving differential equations**.** Each simulation method has a tradeoff between various metrics such as speed, accuracy, and simplicity.

The various methods of simulations used in this thesis include the bilinear transform or trapezoidal integration, the impulse invariance transform, midpoint integration, Simpson's rule, the delta transform or Euler's forward integration rule and Simulink. These methods are used to simulate single-bit and multi-bit CT ∑ΔMs extending from first order to fifth order. These methods are compared with respect to accuracy which is obtained using signal to noise ratio (SNR) and dynamic range (DR), speed of simulation method or total elapsed time, and simplicity of the simulation method. Also, frequency domain analysis is done for all numerical methods and is compared to the CT ∑ΔM's frequency domain analysis and the correctness of the numerical integration *s*-domain to *z*-domain transformation formulas is shown.

5.1 Bilinear Transform or Trapezoidal Integration

In (2.45), the bilinear transform or trapezoidal integration method relates the transfer function in *s*-domain, $H(s)$, and *z*-domain transfer function $H(z)$ using the relation

$$\frac{1}{s} \to \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}. \tag{5.1}$$

The transformation changes the *s*-domain integrators in a CT block diagram to *z*-domain integrators using bilinear transformation or trapezoidal integration as shown in Fig. 5.1.

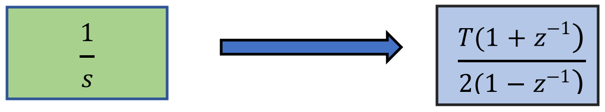


Figure 5.1: Trapezoidal Integrator Block Transformation

For example, if the continuous time integrators in the second order CT ∑ΔM block diagram shown in Fig 4.3 are replaced by their bilinear transform equivalents shown in Fig. 5.1, then the block diagram in Fig. 5.2 shows the DT bilinear transformation model of CT ∑ΔM in Fig.4.3. The bilinear transformation's sampling rate T is chosen to be less than the CT ∑ΔM's sampling rate, T_s.

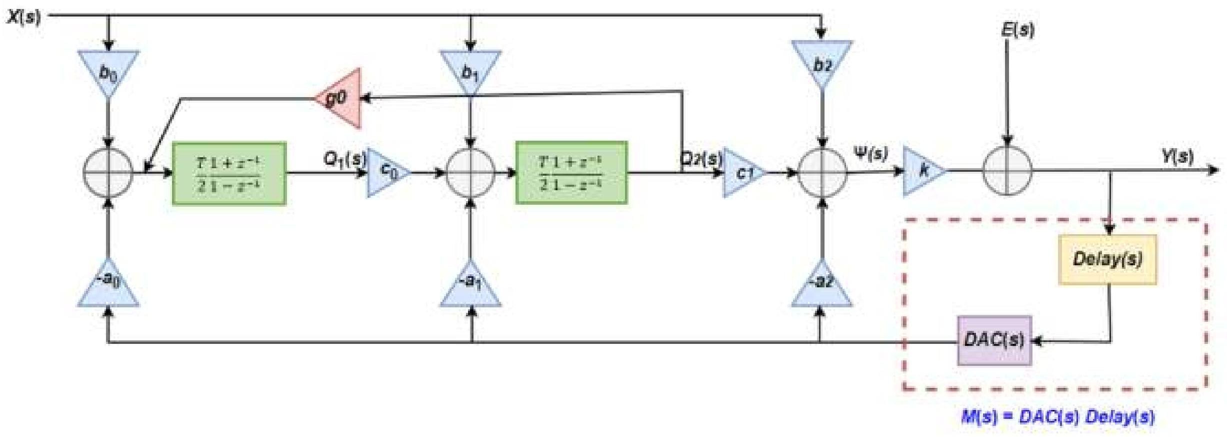


Figure 5.2: Second-order lowpass DT model ∑ΔM block diagram using Trapezoidal Integration

The block diagram in Fig. 5.2 can be used to determine difference equations that can be implemented in MATLAB. The following code implements the block diagram in Fig. 5.2.

```
% Analysis of 2nd Order sigma delta modulator using Trapezoidal Integration
for n = start: finish,
        % First state
        qdot(n,1) = b0 * x(n) - a0 * ydac(n-1) + g0 * q(n-1,2);
        q(n,1) = (T/2) * (qdot(n,1) + qdot(n-1,1)) + q(n-1,1);
        % Second state
        qdot(n,2) = b1 * x(n) - a1 * ydac(n-1) + c0 * q(n,1);
        q(n,2) = (T/2) * (qdot(n,2) + qdot(n-1,2)) + q(n-1,2);
        % Input to quantizer
        Ψ(n) = b2 * x(n) + c1 * q(n,2) - a2 * ydac(n-1);
        % Quantizer
        yq(n) = sign(Ψ(n));
        % DAC
        y(n) = y(n-1);
        if rem(n, TrapOSR) == 0,% Update quantizers every Delta samples
              y(n) = yq(n);
        end
        ydac(n) = y(n-D); % excess loop delay between quantizer and DAC
end
```

5.2 Impulse Invariance Transform

In Table 2.2, the impulse invariance transform relates transfer function in *s*-domain, $H(s)$, to a *z*-domain transfer function $H(z)$ using the relation

$$\frac{1}{s} \rightarrow \frac{T}{1-z^{-1}}. \tag{5.2}$$

The transformation changes the *s*-domain integrators in a CT block diagram to *z*-domain integrators using impulse invariance transformation shown in Fig. 5.3.

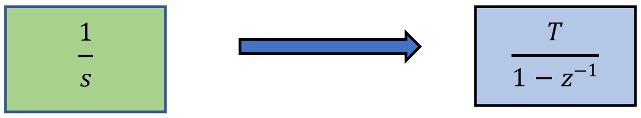


Figure 5.3: Impulse-Invariance Integrator Block Transformation

Fig. 5.4 shows the block diagram of the CT ∑ΔM in Fig. 4.3 where the integrators have been replaced by the impulse invariance equivalents.

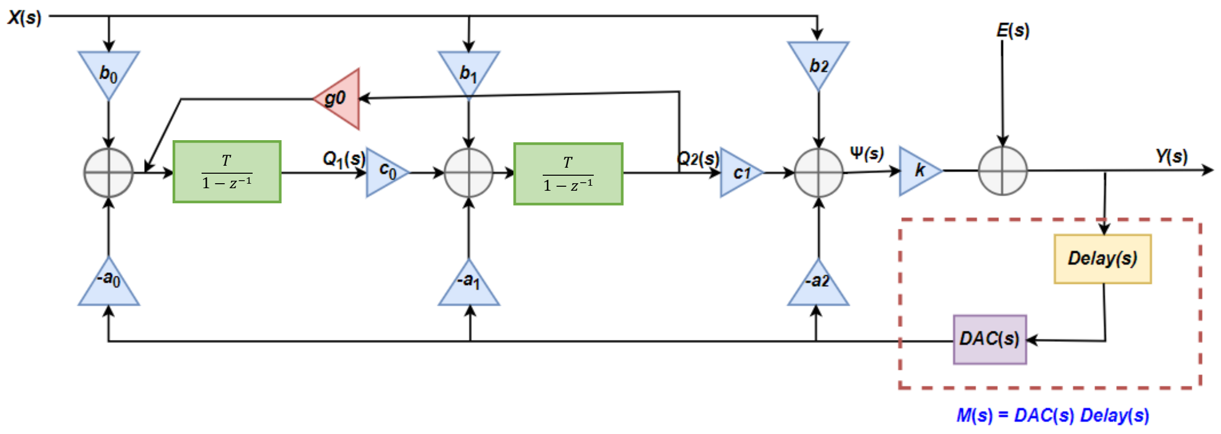


Figure 5.4: Second-order lowpass DT model ∑ΔM block diagram using Impulse Invariance Transformation

The difference equations describing the block diagram in Fig. 5.4 have been implemented in MATLAB using the following code:

```
% Analysis of 2nd Order sigma delta modulator using Impulse Invariance Transformation
for n = start: finish,
       % First state
       qdot(n,1) = b0 * x(n) - a0 * ydac(n-1) + g0 * q(n-1,2);
       q(n,1) = T *(qdot(n,1)) + q(n-1,1);
       % Second state
       qdot(n,2) = b1 * x(n) - a1 * ydac(n-1) + c0 * q(n,1);
       q(n,2) = T * (qdot(n,2)) + q(n-1,2);
       % Input to quantizer
       Ψ(n) = b2 * x(n) + c1 * q(n,2) - a2 * ydac(n-1);
       % Quantizer
       yq(n) = sign(Ψ(n));
       % DAC
       y(n) = y(n-1);
       if rem(n, ImpulseOSR) == 0,% Update quantizers every Delta samples
              y(n) = yq(n);
       end
```

```
        ydac(n) = y(n-D); % excess loop delay between quantizer and DAC
    end
```

5.3 Midpoint Integration

In (2.64), the midpoint integration method relates the transfer function in *s*-domain, $H(s)$, and *z*-domain transfer function $H(z)$ using the relation

$$\frac{1}{s} \longrightarrow \frac{2Tz^{-1}}{1-z^{-2}}. \tag{5.3}$$

The transformation changes the *s*-domain integrators in a CT block diagram to *z*-domain integrators using midpoint integration shown in Fig. 5.5.

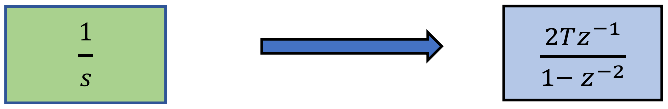


Figure 5.5: Midpoint Integrator Block Transformation

Fig. 5.6 shows the block diagram of the CT ∑ΔM in Fig. 4.3 where the integrators have been replaced by the midpoint integration equivalents.

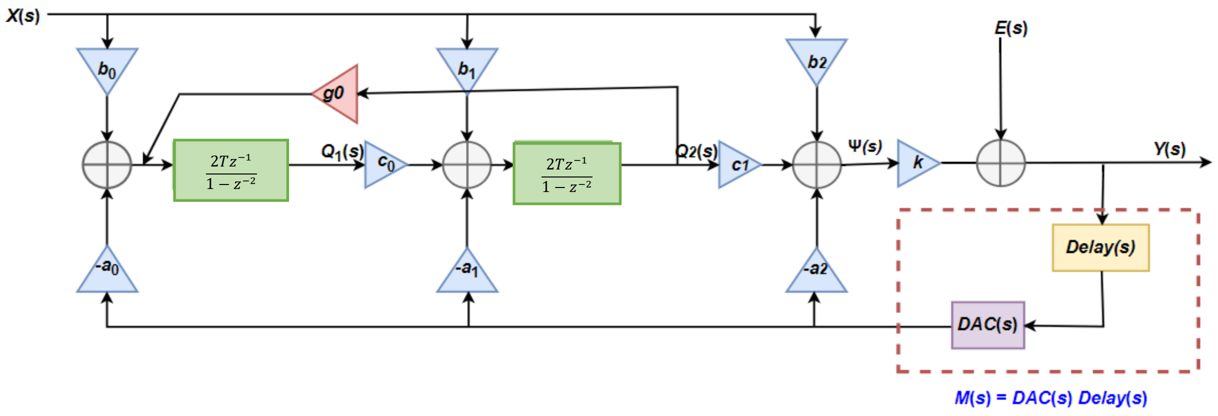


Figure 5.6: Second-order lowpass DT model ∑ΔM block diagram using Midpoint Integration

The difference equations describing the block diagram in Fig. 5.6 have been implemented in MATLAB using the following code:

```
% Analysis of 2nd Order sigma delta modulator using Midpoint Integration
```

```
for n = start: finish,
        % First state
        qdot(n,1) = b0 * x(n) - a0 * ydac(n-1) + g0 * q(n-2,2);
        q(n,1) = 2 * T * (qdot(n-2,1)) + q(n-2, 1);
        % Second state
        qdot(n,2) = b1 * x(n) - a1 * ydac(n-1) + c0 * q(n,1);
        q(n,2) = 2 * T * (qdot(n-2,2)) + q(n-2,2);
        % Input to quantizer
        Ψ(n) = b2 * x(n) + c1 * q(n,2) - a2 * ydac(n-1);
        % Quantizer
        yq(n) = sign(Ψ(n));
        % DAC
        y(n) = y(n-1);
        if rem(n, MidpointOSR) == 0,% Update quantizers every Delta samples
                y(n) = yq(n);
        end
        ydac(n) = y(n-D); % excess loop delay between quantizer and DAC
end
```

5.4 Simpson's Rule

In (2.70), Simpson's rule relates an *s*-domain transfer function $H(s)$ to a *z*-domain transfer function $H(z)$ using the relation

$$\frac{1}{s} \longrightarrow \frac{3}{T}\frac{1-z^{-2}}{1+4z^{-1}+z^{-2}}. \tag{5.4}$$

The transformation changes the *s*-domain integrators in all the block diagram to *z*-domain integrators using Simpson's integration rule shown in Fig. 5.7.

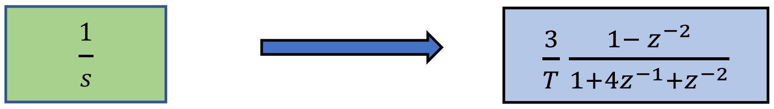


Figure 5.7: Simpsons Rule's Integrator Block Transformation

Fig. 5.8 shows the block diagram of the CT ∑ΔM in Fig. 4.3 where the integrators have been replaced by the Simpson's rule integration equivalents.

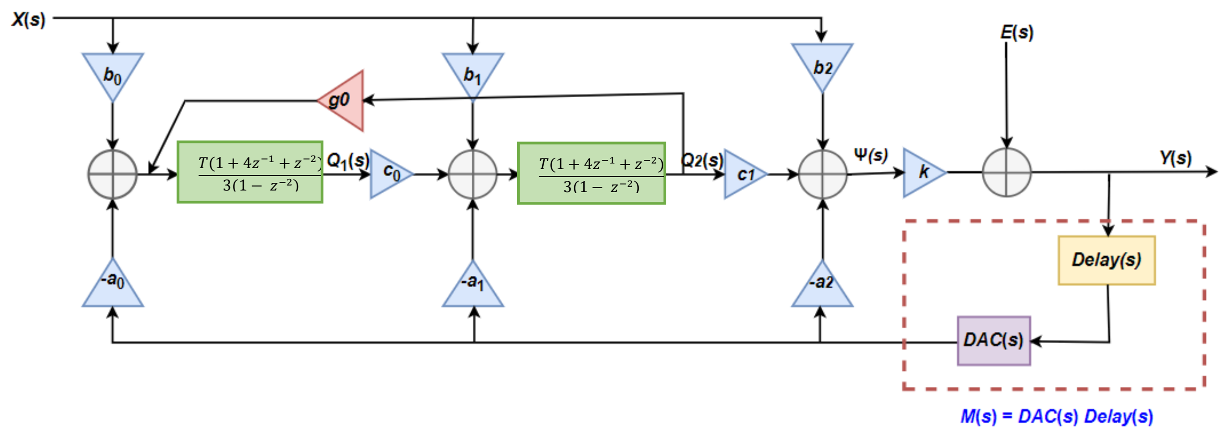


Figure 5.8: Second-order lowpass DT model ∑ΔM block diagram using Midpoint Integration

The difference equations describing the block diagram in Fig. 5.8 have been implemented in MATLAB using the following code:

```
% Analysis of 2nd Order sigma delta modulator using Simpson's Integration Rule
for n = start: finish,
       % First state
       qdot(n,1) = b0 * x(n) - a0 * ydac(n-1) + g0 * q(n-2,2);
       q(n,1) = (T * ((qdot(n,1) + 4 * qdot(n-1,1) + qdot(n-1,1)) + 3 * q(n-2,1))/3;
       % Second state
       qdot(n,2) = b1 * x(n) - a1 * ydac(n-1) + c0 * q(n,1);
       q(n,2) = (T * ((qdot(n,2) + 4 * qdot(n-1,2) + qdot(n-1,2)) + 3 * q(n-2,2))/3;
       % Input to quantizer
       Ψ(n) = b2 * x(n) + c1 * q(n,2) - a2 * ydac(n-1);
       % Quantizer
       yq(n) = sign(Ψ(n));
       % DAC
       y(n) = y(n-1);
       if rem(n, SimpsonsOSR) == 0,% Update quantizers every Delta samples
              y(n) = yq(n);
```

```
        end
        ydac(n) = y(n-D); % excess loop delay between quantizer and DAC
end
```

5.5 Delta Transform or Euler's Forward Integration Rule

In (2.56), the delta transform or Euler's forward integration method relates transfer function in s-domain, $H(s)$, to a z-domain transfer function $H(z)$ using the relation

$$\frac{1}{s} \longrightarrow \frac{Tz^{-1}}{1-z^{-1}}. \qquad (5.5)$$

The transformation changes the s-domain integrators in a CT block diagram to z-domain integrators using delta transformation shown in Fig. 5.9.

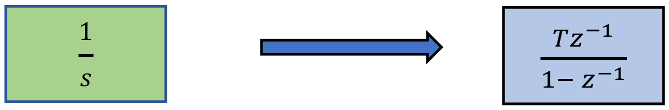


Figure 5.9: Delta Integrator Block Transformation

Fig. 5.10 shows the block diagram of the CT ∑ΔM represented in Fig. 4.3 where the integrators have been replaced by the delta transform equivalents.

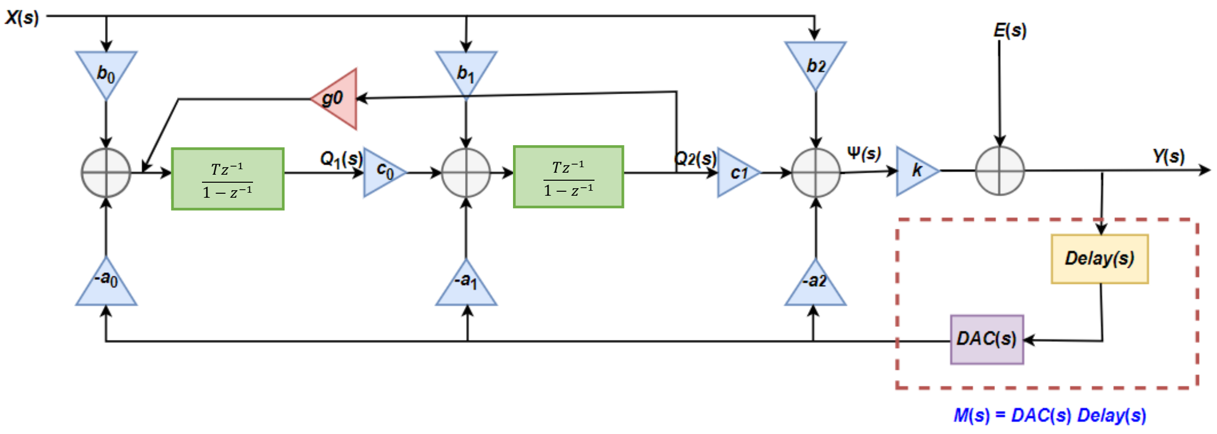


Figure 5.10: Second-order lowpass DT model ∑ΔM block diagram using Midpoint Integration

The difference equations describing the block diagram in Fig. 5.10 have been implemented in MATLAB using the following code:

```
% Analysis of 2nd Order sigma delta modulator using Delta Transform
for n = start: finish,
      % First state
      qdot(n,1) = b0 * x(n) - a0 * ydac(n-1) + g0 * q(n-1,2);
      q(n,1) = T * qdot(n-1,1) + q(n-1,1);
      % Second state
      qdot(n,2) = b1 * x(n) - a1 * ydac(n-1) + c0 * q(n,1);
      q(n,2) = T * qdot(n-1,2) + q(n-1,2);
      % Input to quantizer
      Ψ(n) = b2 * x(n) + c1 * q(n,2) - a2 * ydac(n-1);
      % Quantizer
      yq(n) = sign(Ψ(n));
      % DAC
      y(n) = y(n-1);
      if rem(n, DeltaOSR) == 0,% Update quantizers every Delta samples
            y(n) = yq(n);
      end
      ydac(n) = y(n-D); % excess loop delay between quantizer and DAC
end
```

5.6 MATLAB/Simulink

Modeling schematics in Simulink is often simple because the required blocks can be dragged and connected and simulated easily. Also, simulations with Simulink do not take much simulation time. However, the accuracy of the Simulink depends on the proper selection of the Simulink models [8]. Also, since the blocks used in Simulink are ideal blocks, non-idealities in real circuits are not modeled easily in Simulink simulations.

MATLAB/Simulink provides a set of solvers and each Simulink solver uses a particular method to solve a model. Since, the models are represented by difference equations, the solver applies numerical methods to solve the equations. Simulink solvers are classified as either continuous solvers or discrete solvers in terms of nature of states in the Simulink model. Continuous solvers use numerical integration methods to compute the continuous states of a model and these solvers depend on individual blocks to compute values of discrete states at each step. Discrete solvers are used only for solving purely discrete models. Solvers are classified in terms of step-size as either a fixed-step solver or a variable-step solver. Lower step size usually results in more accurate simulation but at the cost of an increase in simulation time. For the fixed-step solver, the step-size is fixed throughout the simulation either by the user at the start of simulation or is fixed automatically by the solver. For a variable-step solver, the step size varies dynamically during the simulation depending upon the dynamics of the solver. When the states of any model vary rapidly, the step-size is reduced but when the states of a model vary slowly, the step-size is increased. Depending upon number of steps, solvers can be classified as either one-step or multistep continuous solvers, and depending upon the order, solvers are classified as either single-order or variable-order solvers. Also, the solvers are categorized as implicit and explicit solvers. Implicit solvers are used for solving problems that are stiff whereas explicit solvers are used for solving problems that are non-stiff. If the desired solution of any differential equation varies slowly but there are closer solutions of the differential equation which vary rapidly, then this type of problem is classified as a stiff-problem and stiff problems are solved by using implicit solvers. They have to be suitable for both slowly and quickly varying dynamic models. But for non-varying continuous dynamics, explicit solvers are efficient as they can use larger step-sizes and this also reduces the computation time.

The classification of Simulink solvers has been summarized in Table 5.1.

Note: The desired solution to a differential equation varies very slowly in stiff problems. But there are closer solutions that vary rapidly. But for non-stiff problems, the desired solution to a differential equation varies rapidly.

Criteria	Classification
▪ Order of the Solver	▪ Single-Order Solver ▪ Variable-Order Solver
▪ Number of steps	▪ One-step Solver ▪ Multistep Solver
▪ Nature of states	▪ Continuous Solvers ▪ Explicit Solvers
▪ Step-size used in Computation	▪ Fixed-step Solvers ▪ Variable-step Solvers
▪ Nature of the problem	▪ Implicit Solvers ▪ Explicit Solvers

Table 5.1: Classification of Simulink Solvers

For this thesis, the Simulink solvers used for modeling CT ∑ΔMs are variable-step explicit solvers and they are:

- **Ode23:** Solver uses Runge-Kutta, Bogacki and Shampine (2, 3) pair explicit method for numerically integrating differential equations. Three function evaluations per steps are used in Ode23 solver.
- **Ode45:** Solver uses Runge-Kutta, Dormand-Prince (4, 5) pair explicit method for solving ordinary differential equations. It uses six function evaluations to calculate fourth-order and fifth order accurate solutions. It is the default solver of Simulink especially designed for solving models with continuous states.

The Simulink schematics of first, second, third, fourth and fifth-order CT ∑ΔMs using 3-bit quantizers are shown in Fig. 5.12, Fig. 5.13, Fig. 5.14, Fig. 5.15, and Fig. 5.16 respectively.

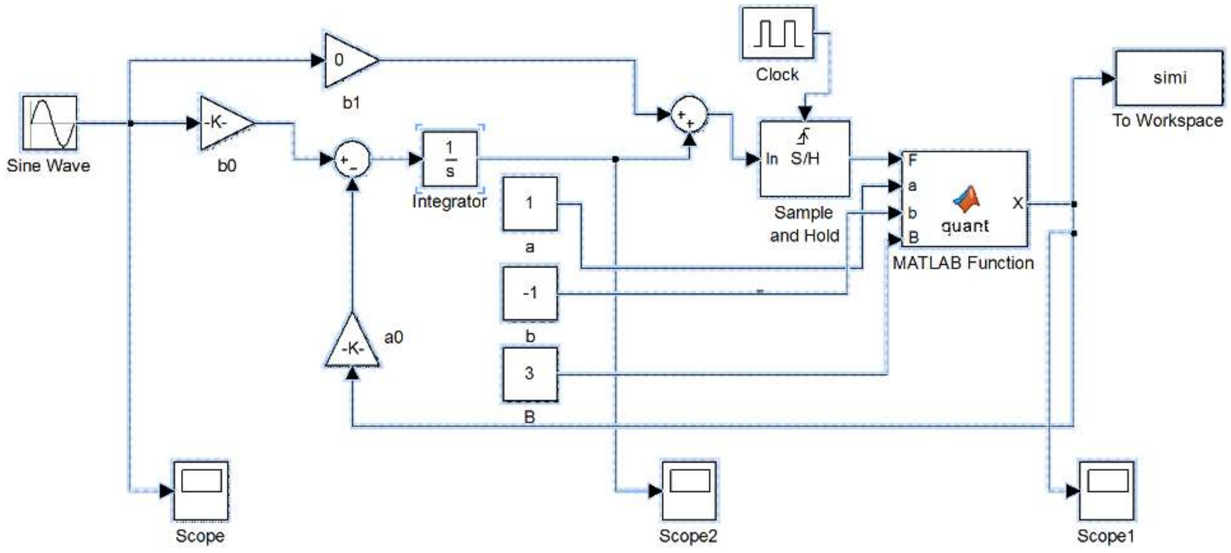


Figure 5.11: Simulink model for a three-bit first-order CT ∑ΔM

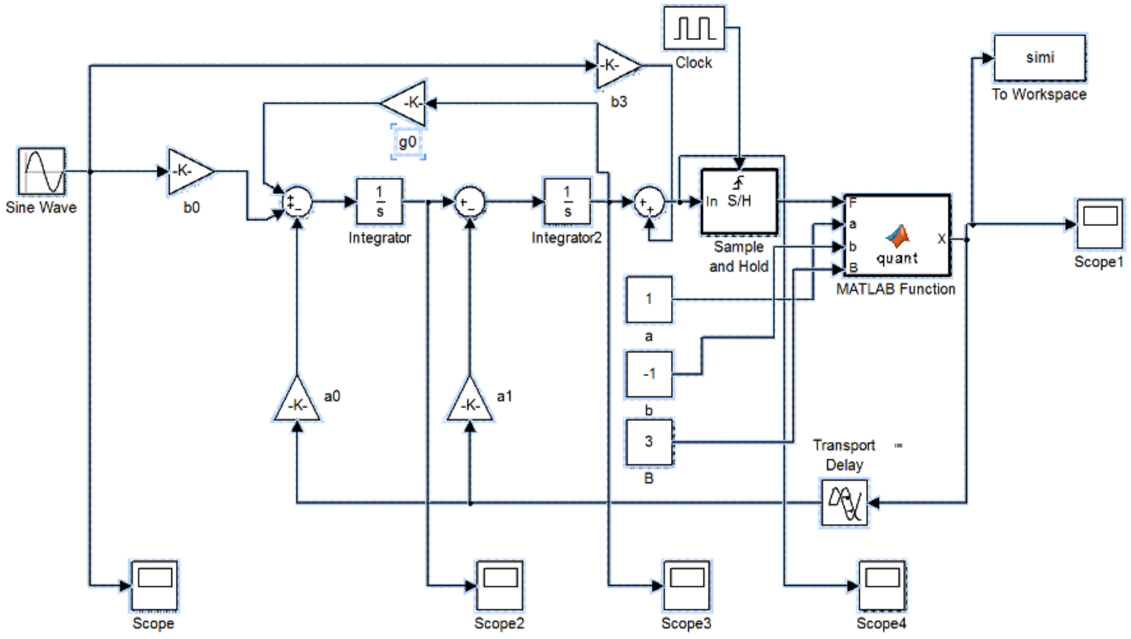


Figure 5.12: Simulink model for a three-bit second-order CT ∑ΔM

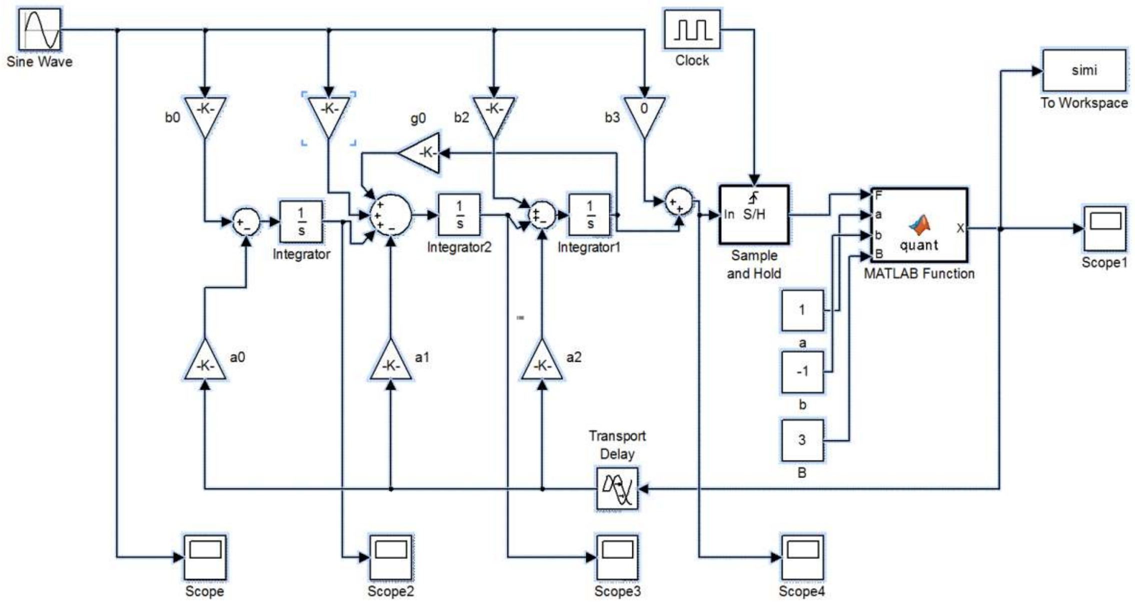


Figure 5.13: Simulink model for a three-bit third-order CT ∑ΔM

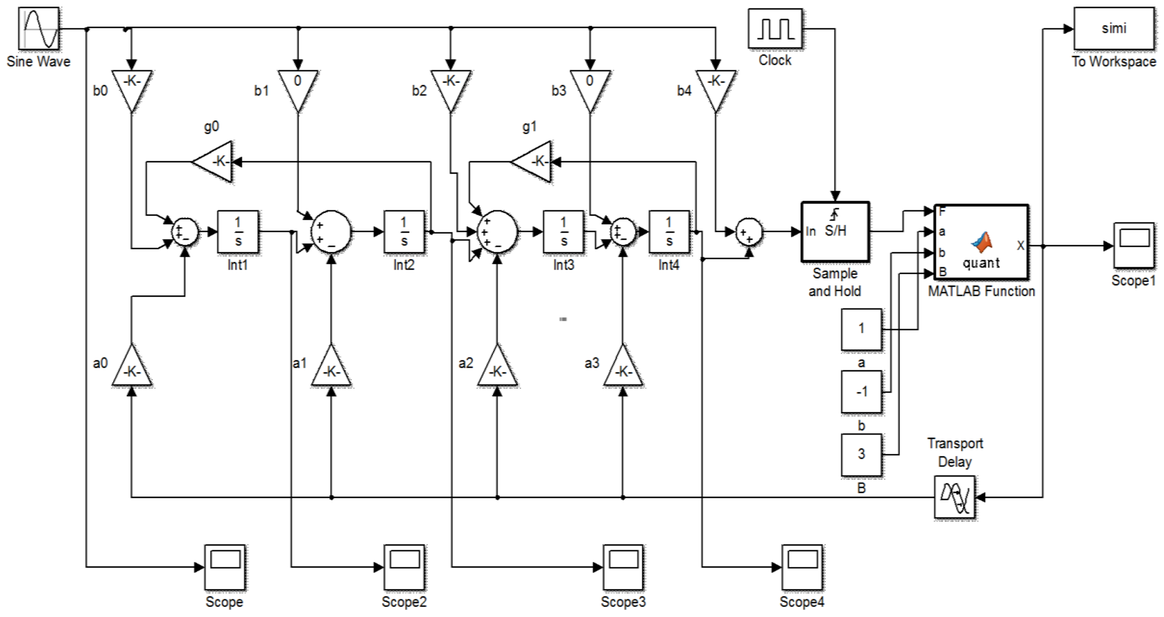


Figure 5.14: Simulink model for a three-bit fourth-order CT ∑ΔM

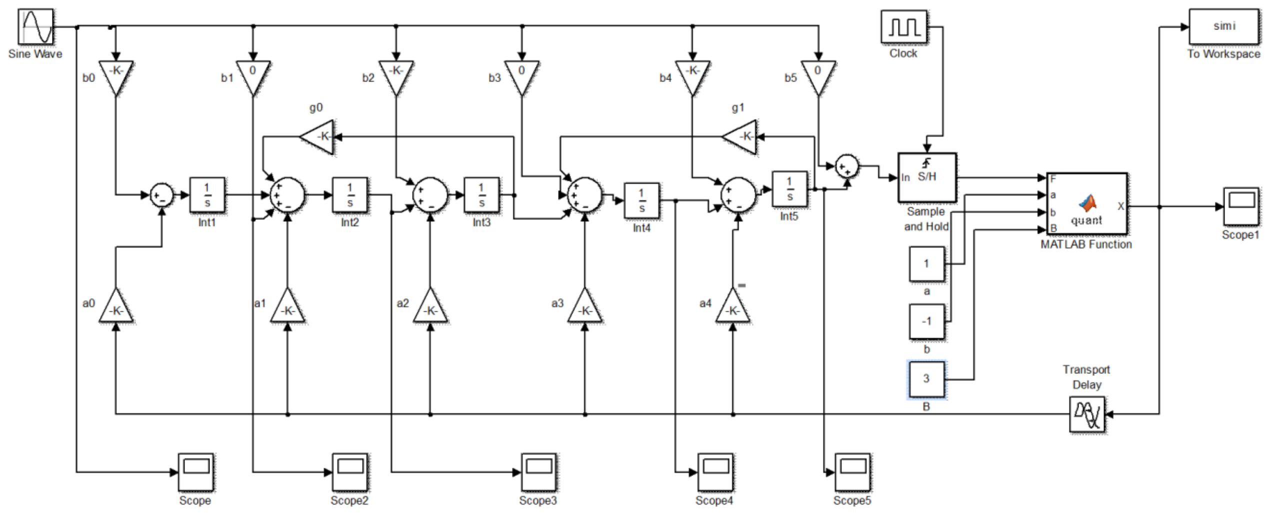


Figure 5.15: Simulink model for a three-bit fifth-order CT ∑ΔM

5.7 Simulation Method Comparison

To compare the five numerical integration methods (delta transform, impulse invariance, midpoint integration, Simpson's rule and trapezoidal integration) and two CT ∑ΔM Simulink solver models (ode23 and ode45), five first, second, third, fourth and fifth order single-bit, two-bit and three-bit CT ∑ΔMs are simulated. The specifications are given in Table 5.1.

Specification	
Number of bits in the quantizer	1, 2, 3
Order of the loop filter	1, 2, 3, 4, 5
∑ΔM Sampling Frequency	1 GHz
∑ΔM Bandwidth	20 MHz
Input Frequency	1 MHz (1st order), 19MHz (2nd order, 4th order, 5th Order), 10MHz (3rd order)
Integration Over Sampling Ratio (OSR)	10

Table 5.2: Simulation Condition

The signal to noise ratios (SNRs), dynamic ranges (DRs) and the total simulation time values of all single-bit, two-bit and three-bit CT ∑ΔMs using all the simulation methods have been tabulated in the sections that follow. Also, the magnitude spectrum of all the simulation methods are compared. For calculating the SNR, the total signal power has been divided by the total noise power. For calculating the DR, the ratio of the signal power has been divided by the maximum noise signal power multiplied by the number of noise signals in the required bandwidth of 20 MHz. Both the SNR and DR values have been converted into decibels (dBs). For calculating the simulation time, MATLAB's builtin function `cputime` has been used during the code simulation. The average SNRs, DRs and the values of total simulation time have been calculated for all simulation methods and all orders of CT ∑ΔMs.

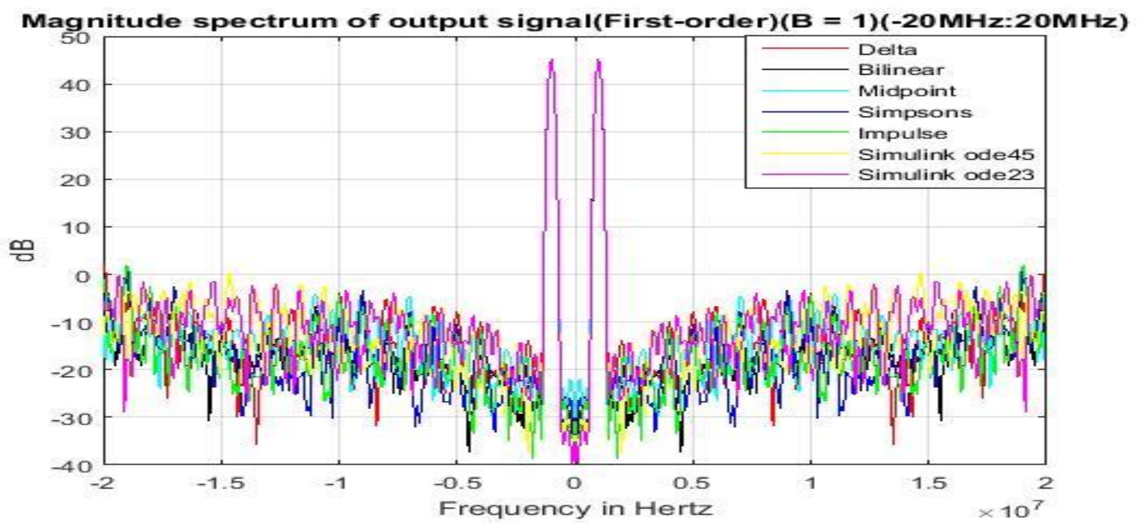
5.7.1 First Order CT ∑ΔM Simulation Results

Table 5.3 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the first-order single-bit CT ∑ΔM simulations.

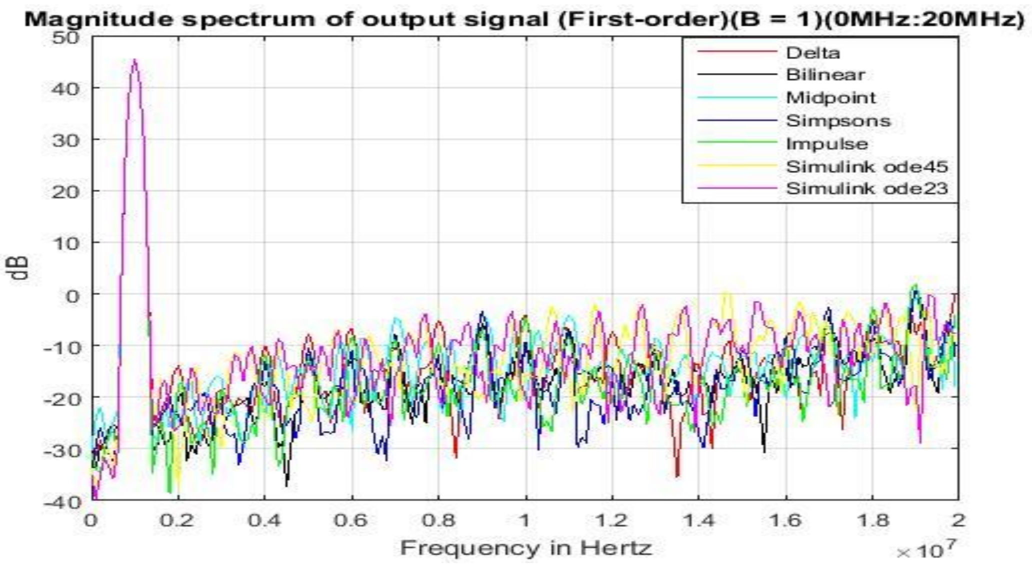
Simulation Methods	**SNR (dB)**	**DR (dB)**	**Time (sec)**
Delta Transform	35.17	15.57	0.36
Impulse Invariance	34.89	15.44	0.36
Midpoint Integration	34.41	15.63	0.43
Simpsons Rule	34.89	15.49	0.43
Trapezoidal Integration	35.00	15.54	0.39
Simulink (Ode45)	32.95	15.55	0.90
Simulink (Ode23)	33.04	15.54	0.78

Table 5.3: SNR, DR and total simulation time for first-order single-bit CT ∑ΔM

Fig. 5.16 shows the magnitude spectrum of all the simulation methods for first-order single-bit CT∑ΔMs.



a)



b)

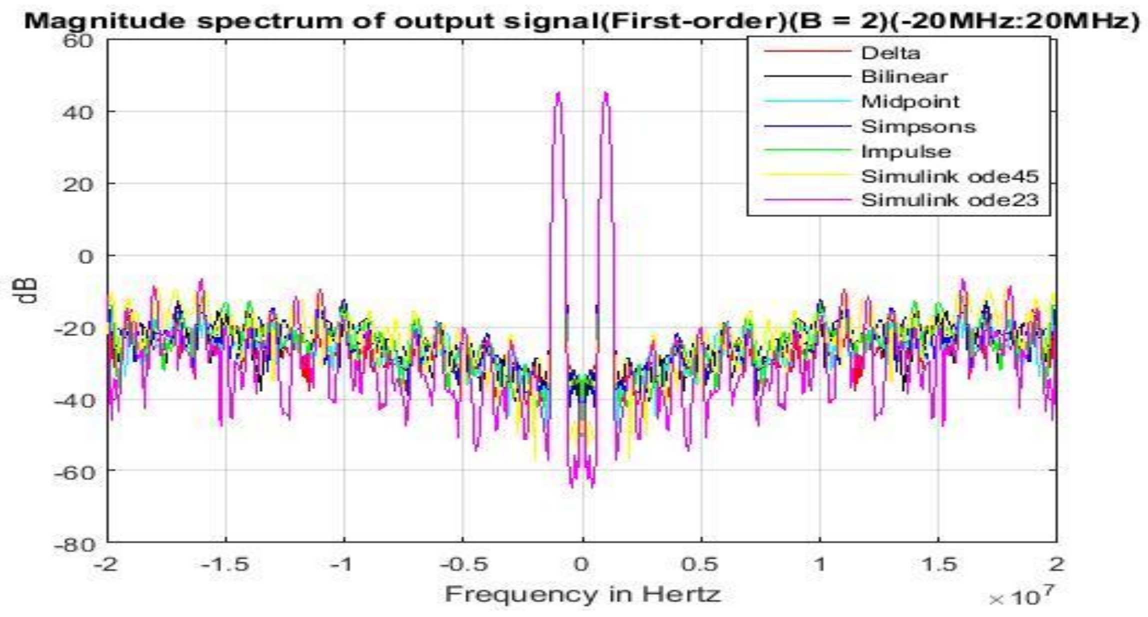
Figure 5.16: The combined output magnitude spectrum of the simulation methods for first-order single-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

Table 5.4 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the first-order two-bit CT ∑ΔM simulations.

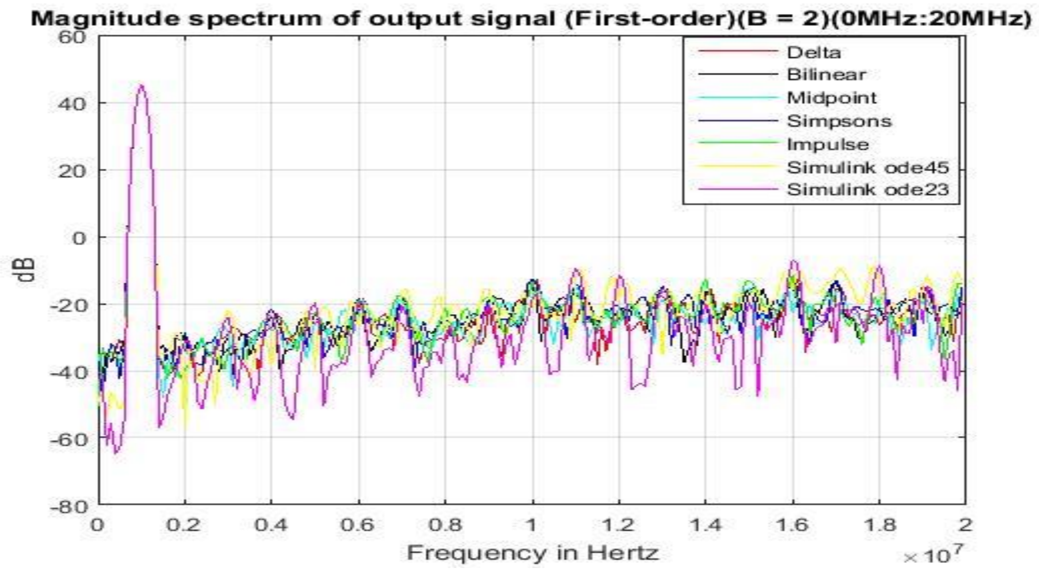
Simulation Methods	SNR (dB)	DR (dB)	Time (sec)
Delta Transform	37.83	15.57	0.37
Impulse Invariance	37.72	15.57	0.42
Midpoint Integration	37.73	15.61	0.38
Simpsons Rule	37.75	15.55	0.43
Trapezoidal Integration	37.89	15.53	0.38
Simulink (Ode45)	37.22	15.53	0.97
Simulink (Ode23)	37.71	15.55	0.78

Table 5.4: SNR, DR and total simulation time for first-order two-bit CT ∑ΔM

Fig. 5.17 shows the magnitude spectrum of all the simulation methods for first-order two-bit CT ∑ΔMs.



a)



b)

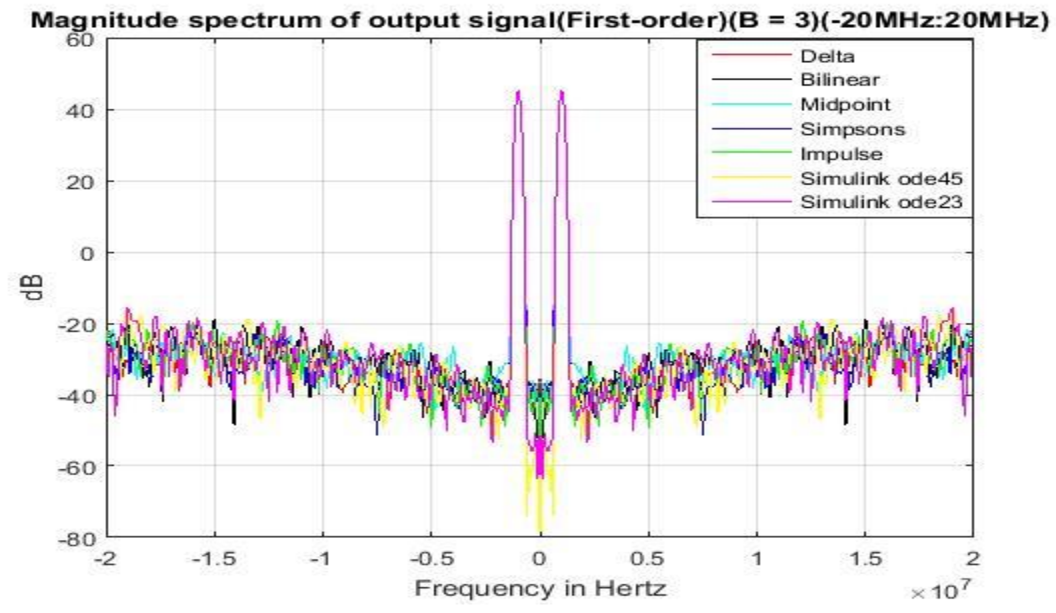
Figure 5.17: The combined output magnitude spectrum of the simulation methods for first-order two-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

Table 5.5 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the first-order three-bit CT ∑ΔM simulations.

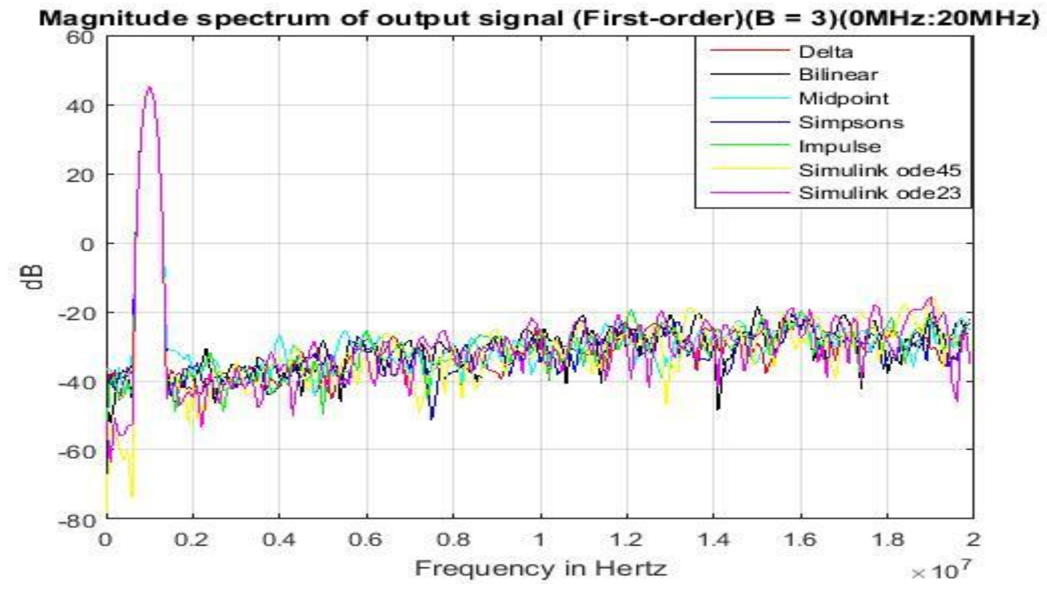
Simulation Methods	**SNR (dB)**	**DR(dB)**	**Time (sec)**
Delta Transform	38.34	15.57	0.37
Impulse Invariance	38.24	15.53	0.43
Midpoint Integration	38.25	15.51	0.39
Simpsons Rule	38.21	15.49	0.43
Trapezoidal Integration	38.30	15.54	0.39
Simulink (Ode45)	38.30	15.56	0.91
Simulink (Ode23)	38.22	15.55	0.77

Table 5.5: SNR, DR and total simulation time for first-order three-bit CT ∑ΔM

Fig. 5.18 shows the magnitude spectrum of all the simulation methods for first-order three-bit CT ∑ΔMs.



a)



b)

Figure 5.18: The combined output magnitude spectrum of the simulation methods for first-order three-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

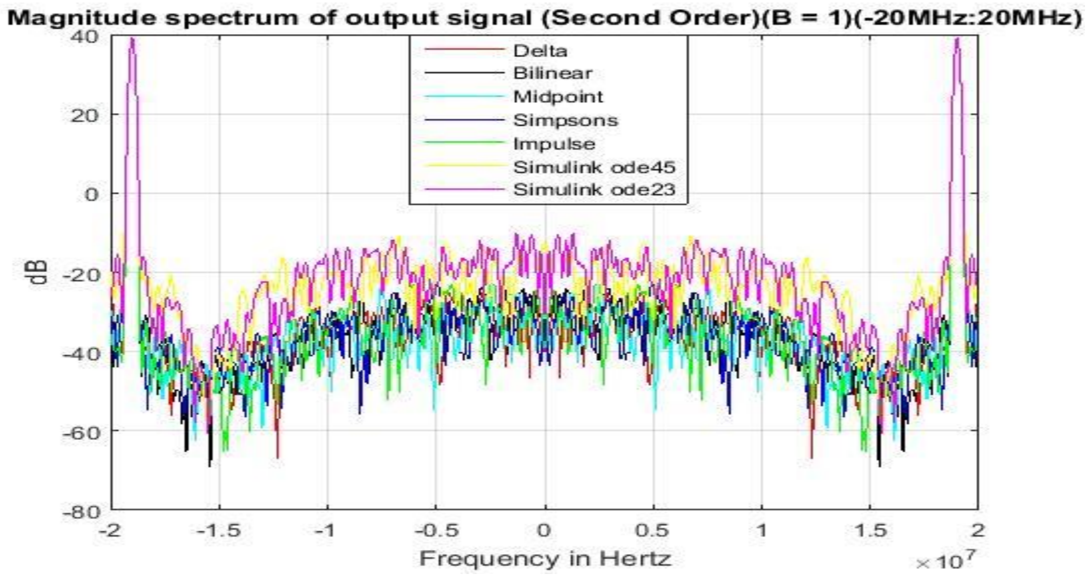
5.7.2 Second Order CT ∑ΔM Simulation Results

Table 5.6 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the second order single-bit CT ∑ΔM simulations.

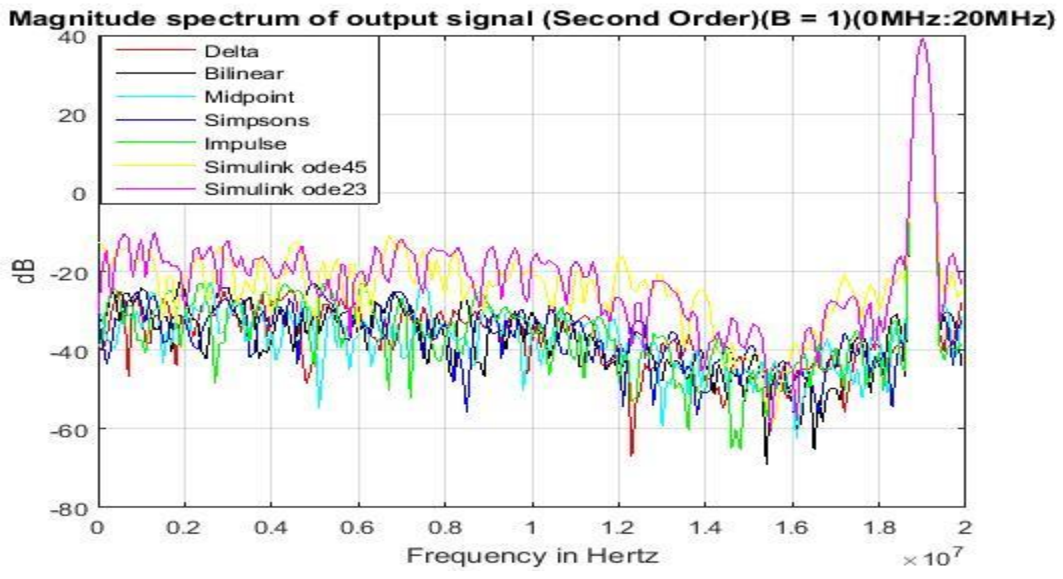
Simulation Methods	SNR (dB)	DR (dB)	Time (sec)
Delta Transform	51.84	41.38	0.48
Impulse Invariance	50.34	38.76	0.50
Midpoint Integration	50.31	38.39	0.46
Simpsons Rule	51.93	39.33	0.65
Trapezoidal Integration	51.41	38.38	0.50
Simulink (Ode45)	35.59	15.64	0.93
Simulink (Ode23)	35.27	14.84	0.92

Table 5.6: SNR, DR and total simulation time for second-order single-bit CT ∑ΔM

Fig.5.19 shows the magnitude spectrum of all the simulation methods for second order single-bit CT ∑ΔMs.



a)



b)

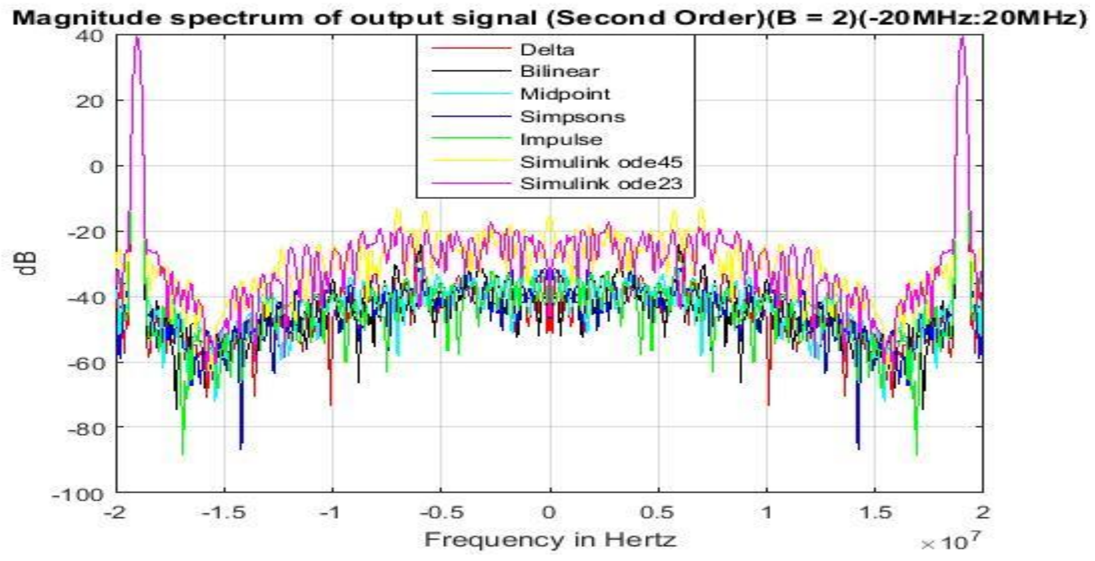
Figure 5.19: The combined output magnitude spectrum of the simulation methods for second-order single-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

Table 5.7 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the second order 2-bit CT ∑ΔM simulations.

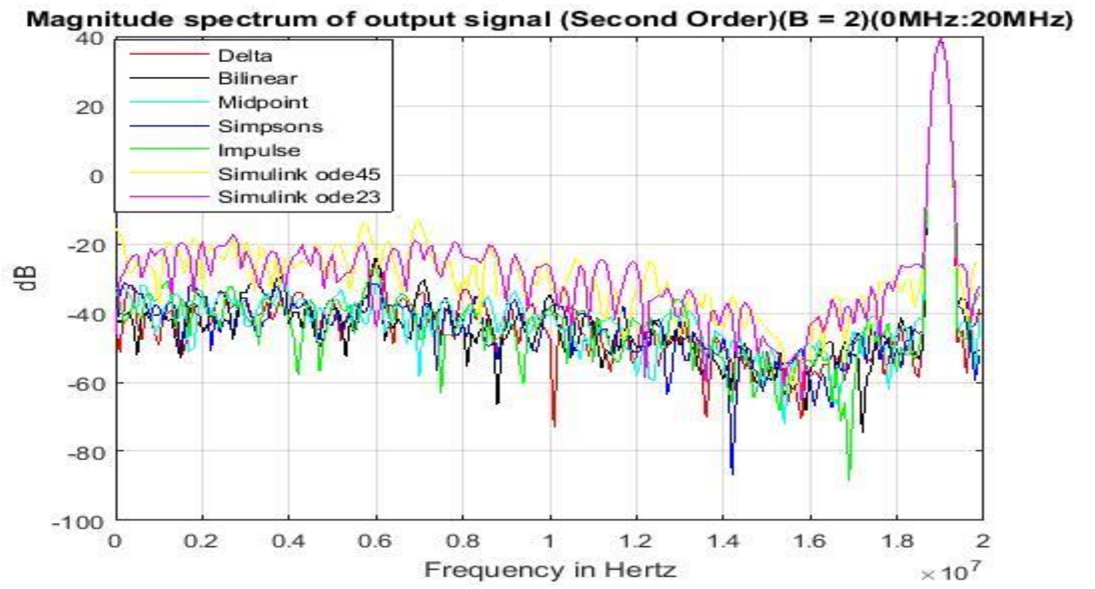
Simulation Methods	**SNR (dB)**	**DR (dB)**	**Time (sec)**
Delta Transform	58.50	47.90	0.53
Impulse Invariance	58.56	47.59	0.50
Midpoint Integration	58.32	46.04	0.53
Simpsons Rule	59.40	44.98	0.65
Trapezoidal Integration	58.52	43.63	0.51
Simulink (Ode45)	37.31	15.18	1.07
Simulink (Ode23)	35.53	15.36	0.99

Table 5.7: SNR, Dynamic Range and total simulation time for Second Order (2-bit) CT ∑ΔM

Figure5.20 shows the magnitude spectrum of all the simulation methods for second-order two-bit CT ∑ΔMs.



a)



b)

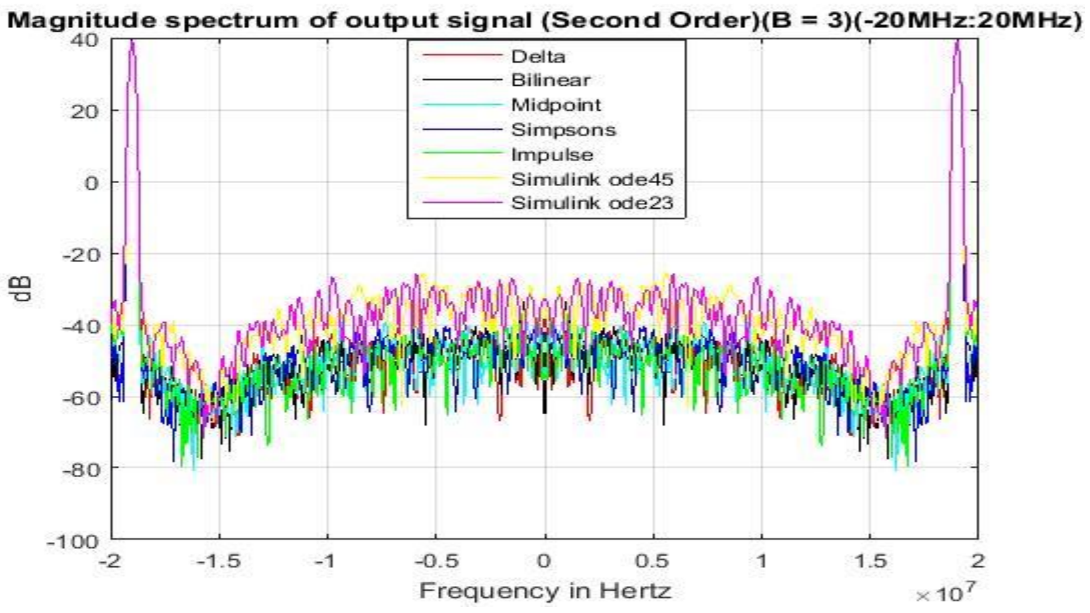
Figure 5.20: The combined output magnitude spectrum of the simulation methods for second-order 2-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

Table 5.8 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the second-order three-bit CT ∑ΔM simulations.

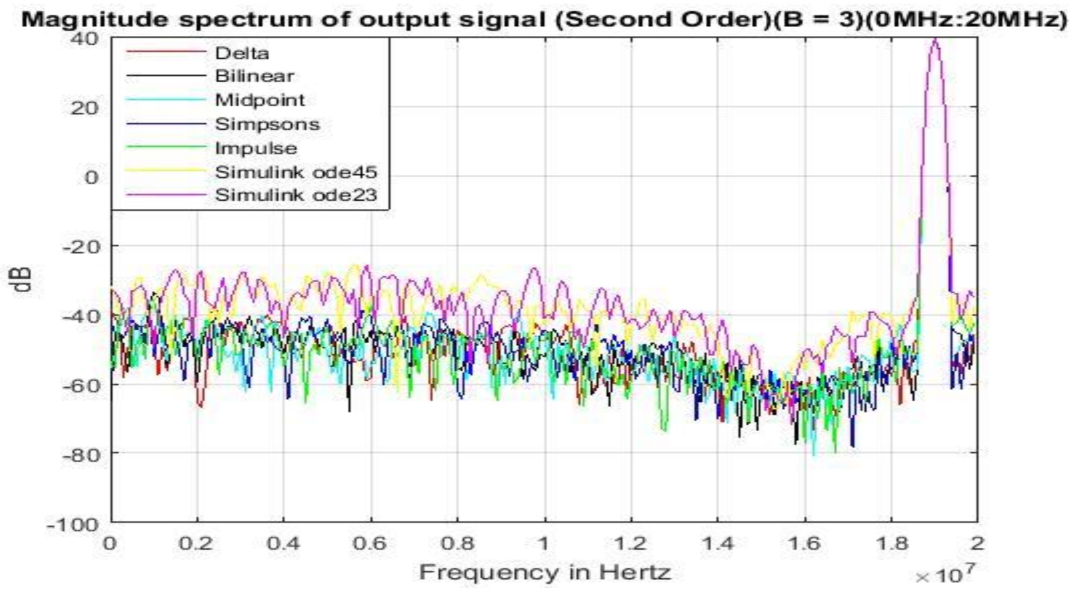
Simulation Methods	SNR (dB)	DR (dB)	Time (sec)
Delta Transform	66.59	54.50	0.53
Impulse Invariance	66.42	53.33	0.50
Midpoint Integration	66.40	52.16	0.53
Simpsons Rule	66.72	54.17	0.67
Trapezoidal Integration	66.37	51.93	0.51
Simulink (Ode45)	38.27	15.33	1.11
Simulink (Ode23)	38.24	15.42	0.95

Table 5.8: SNR, Dynamic Range and total simulation time for second-order three-bit CT ∑ΔM

Fig. 5.21 shows the magnitude spectrum of all the simulation methods for second-order three-bit CT ∑ΔMs.



a)



b)

Figure 5.21: The combined output magnitude spectrum of the simulation methods for second-order three-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

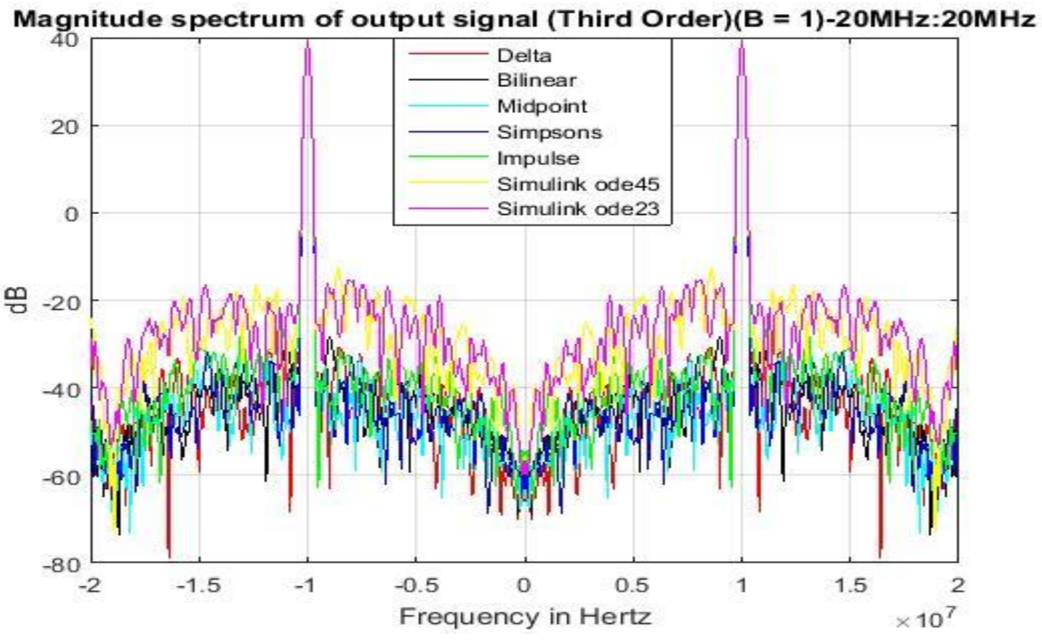
5.7.3 Third Order CT ∑ΔM Simulation Results

Table 5.9 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the third order single-bit CT ∑ΔM simulations.

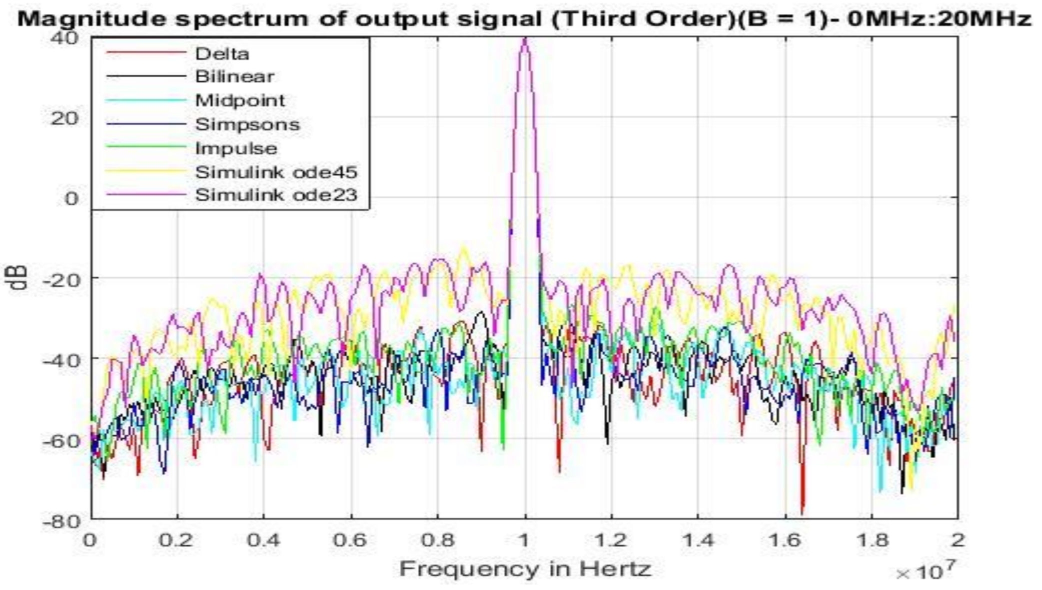
Simulation Methods	SNR (dB)	DR (dB)	Time (sec)
Delta Transform	58.97	47.25	0.57
Impulse Invariance	55.66	44.92	0.54
Midpoint Integration	59.12	46.85	0.54
Simpsons Rule	61.44	47.33	0.83
Trapezoidal Integration	60.29	46.77	0.56
Simulink (Ode45)	37.28	15.09	1.09
Simulink (Ode23)	37.10	15.50	1.04

Table 5.9: SNR, Dynamic Range and total simulation time for third order single-bit CT ∑ΔM

Fig. 5.22 shows the magnitude spectrum of all the simulation methods for third-order single-bit CT ∑ΔMs.



a)



b)

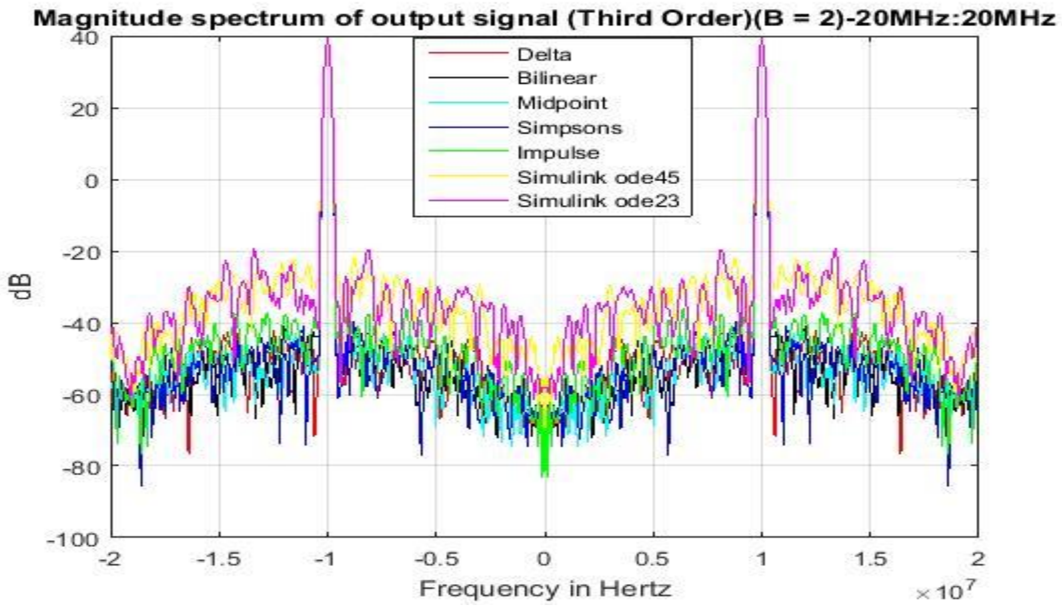
Figure 5.22: The combined output magnitude spectrum of the simulation methods for third-order single-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

Table 5.10 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the third order 2-bit CT ∑ΔM simulations.

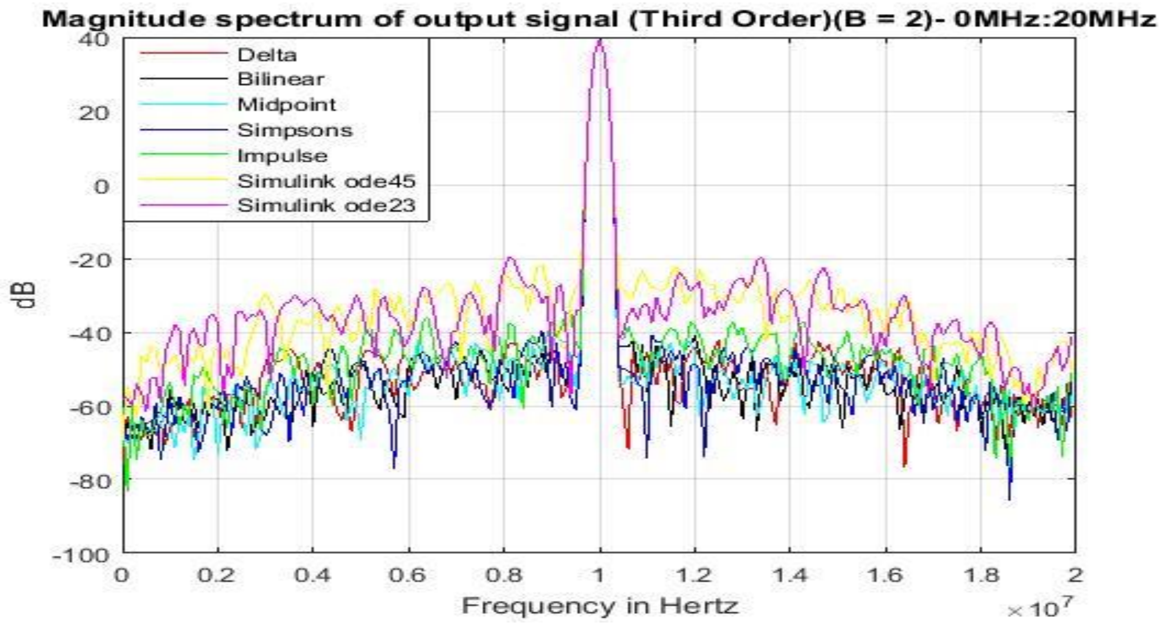
Simulation Methods	**SNR (dB)**	**DR (dB)**	**Time (sec)**
Delta Transform	69.11	58.69	0.56
Impulse Invariance	68.80	55.80	0.54
Midpoint Integration	68.52	57.02	0.55
Simpsons Rule	69.22	56.89	0.83
Trapezoidal Integration	68.78	57.38	0.57
Simulink (Ode45)	37.91	15.14	1.00
Simulink (Ode23)	38.11	15.49	0.99

Table 5.10: SNR, Dynamic Range and total simulation time for third-order two-bit CT ∑ΔM

Fig. 5.22 shows the magnitude spectrum of all the simulation methods for third-order single-bit CT ∑ΔMs.



a)



b)

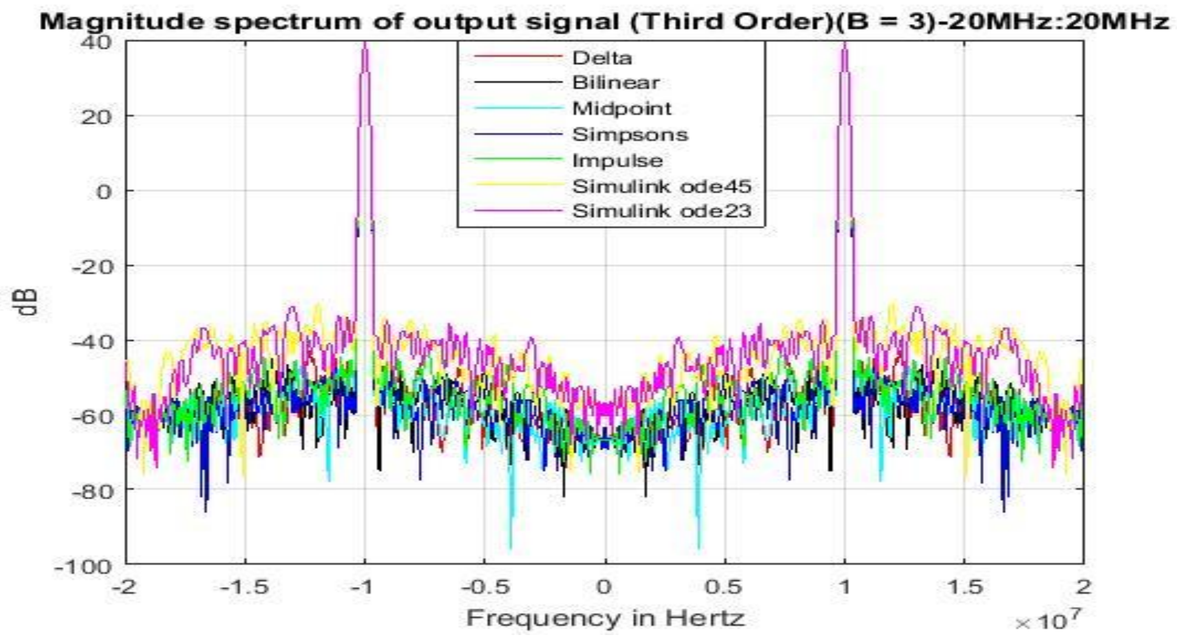
Figure 5.23: The combined output magnitude spectrum of the simulation methods for third-order two-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

Table 5.11 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the third order three-bit CT ∑ΔM simulations.

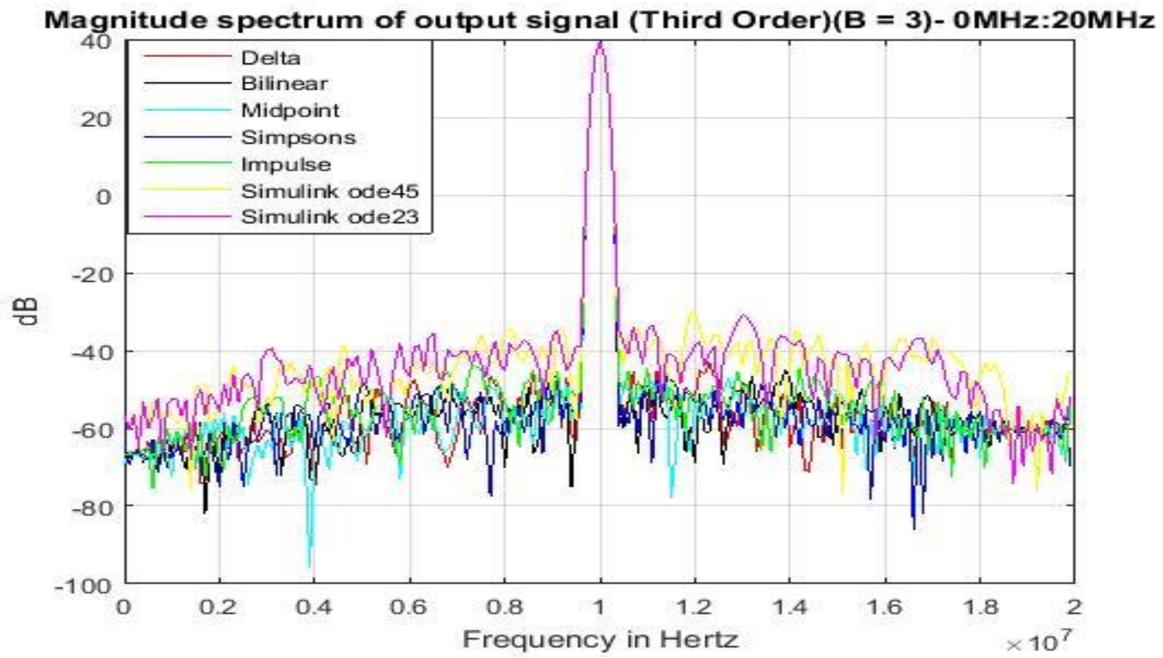
Simulation Methods	**SNR (dB)**	**DR (dB)**	**Time (sec)**
Delta Transform	73.40	59.42	0.54
Impulse Invariance	72.26	60.25	0.56
Midpoint Integration	73.93	61.76	0.54
Simpsons Rule	74.63	62.13	0.83
Trapezoidal Integration	73.97	61.21	0.55
Simulink (Ode45)	38.36	15.37	0.99
Simulink (Ode23)	38.35	15.45	0.89

Table 5.11: SNR, Dynamic Range and total simulation time for Third Order (3-bit) CT ∑ΔM

Fig. 5.16 shows the magnitude spectrum of all the simulation methods for third-order three-bit CT ∑ΔMs.



a)



b)

Figure 5.24: The combined output magnitude spectrum of the simulation methods for third-order three-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

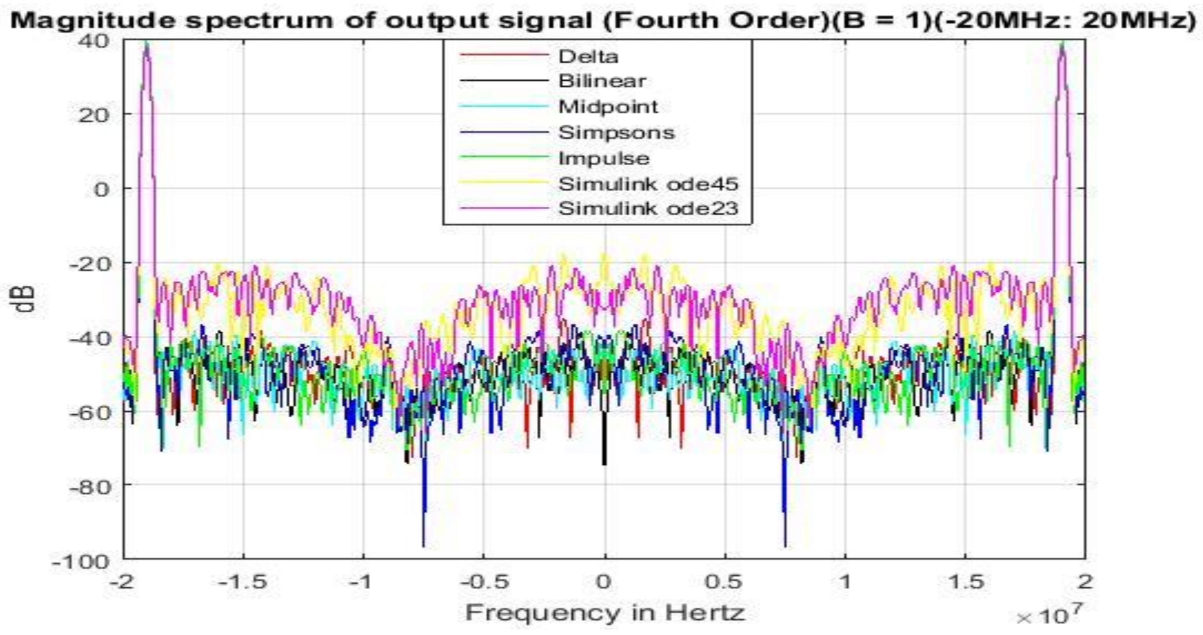
5.7.4 Fourth Order CT ∑ΔM Simulation Results

Table 5.12 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the fourth-order single-bit CT ∑ΔM simulations.

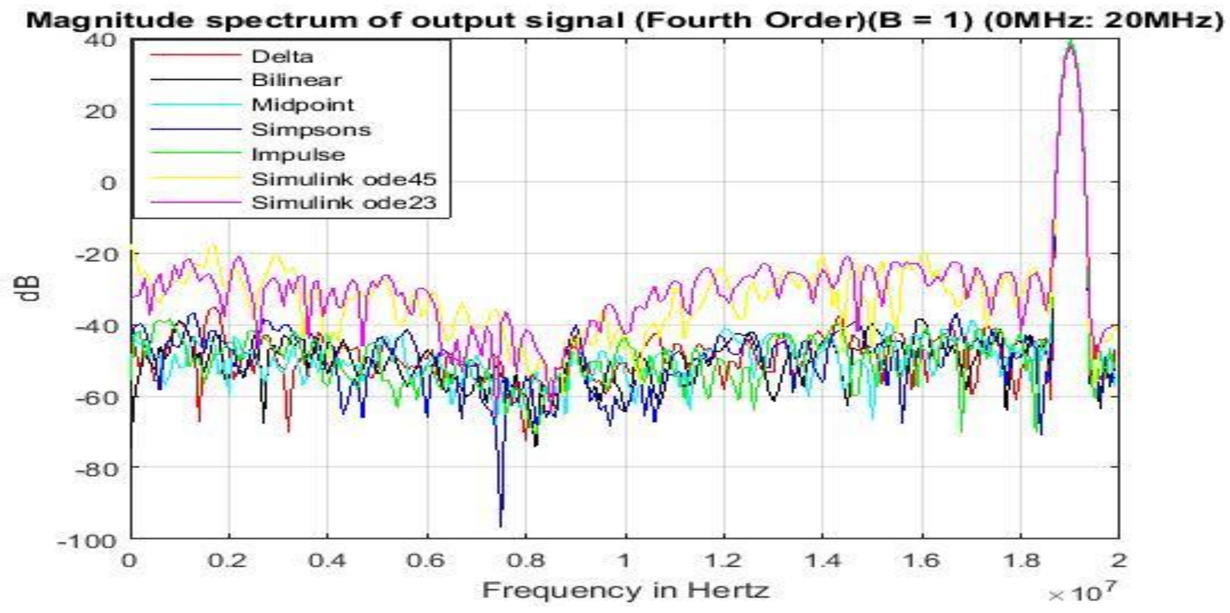
Simulation Methods	SNR (dB)	DR (dB)	Time (sec)
Delta Transform	65.33	51.51	0.59
Impulse Invariance	66.02	54.85	0.57
Midpoint Integration	65.33	55.04	0.57
Simpsons Rule	66.29	53.92	1.00
Trapezoidal Integration	66.13	54.92	0.60
Simulink (Ode45)	37.81	15.92	1.23
Simulink (Ode23)	37.63	15.60	1.15

Table 5.12: SNR, Dynamic Range and total simulation time for fourth-order single-bit CT ∑ΔM

Fig. 5.25 shows the magnitude spectrum of all the simulation methods for fourth-order single-bit CT ∑ΔMs.



a)



b)

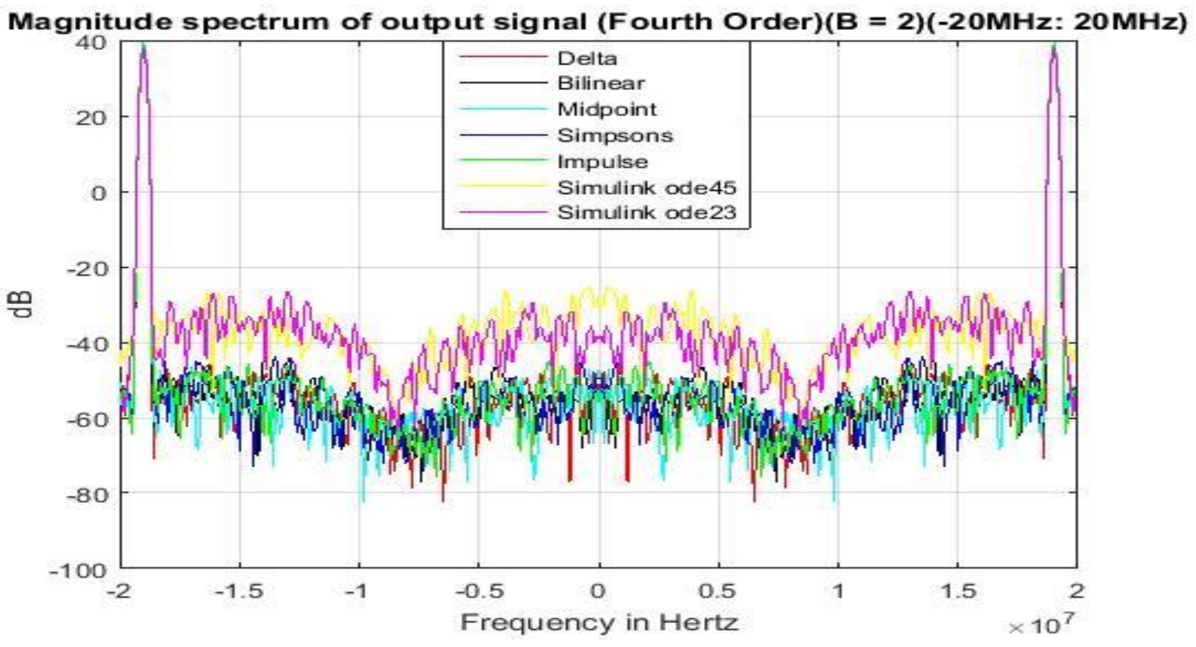
Figure 5.25: The combined output magnitude spectrum of the simulation methods for fourth-order single-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

Table 5.13 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the fourth-order two-bit CT ∑ΔM simulations.

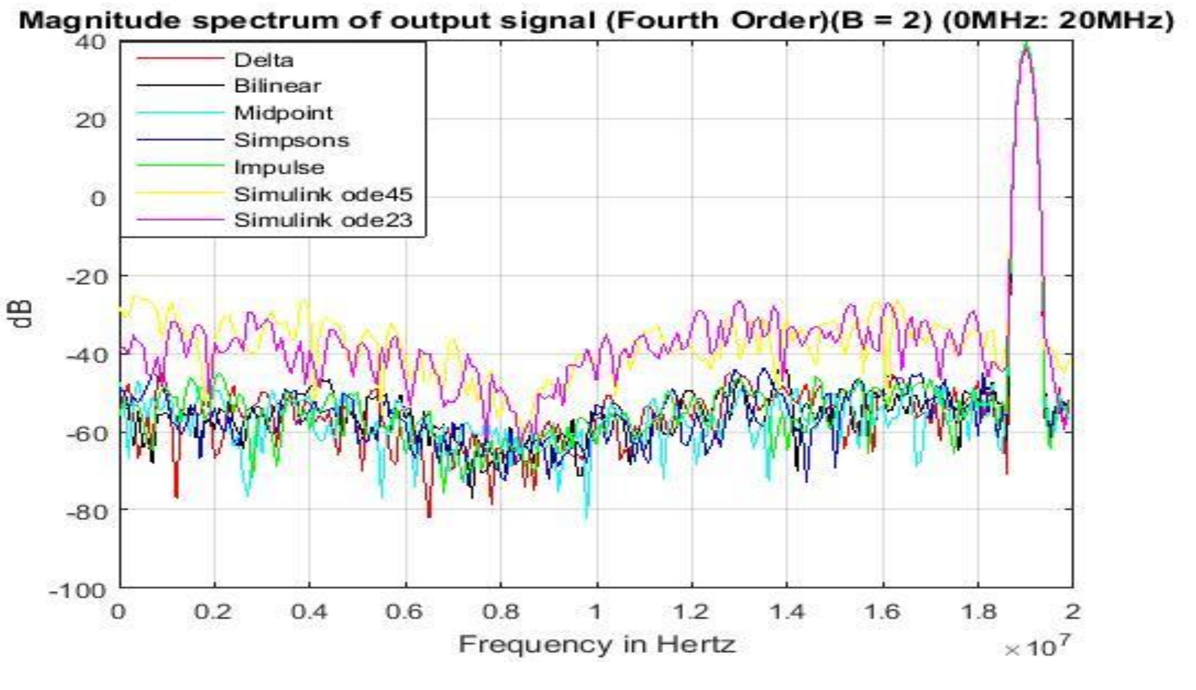
Simulation Methods	**SNR (dB)**	**DR (dB)**	**Time (sec)**
Delta Transform	72.41	61.24	0.59
Impulse Invariance	72.11	61.18	0.57
Midpoint Integration	72.12	63.37	0.57
Simpsons Rule	72.82	60.63	1.00
Trapezoidal Integration	72.38	59.16	0.59
Simulink (Ode45)	38.20	15.49	1.18
Simulink (Ode23)	38.36	14.58	1.17

Table 5.3: SNR, Dynamic Range and total simulation time for fourth-order two-bit CT ∑ΔM

Figure5.16 shows the magnitude spectrum of all the simulation methods for fourth-order two-bit CT ∑ΔMs.



a)



b)

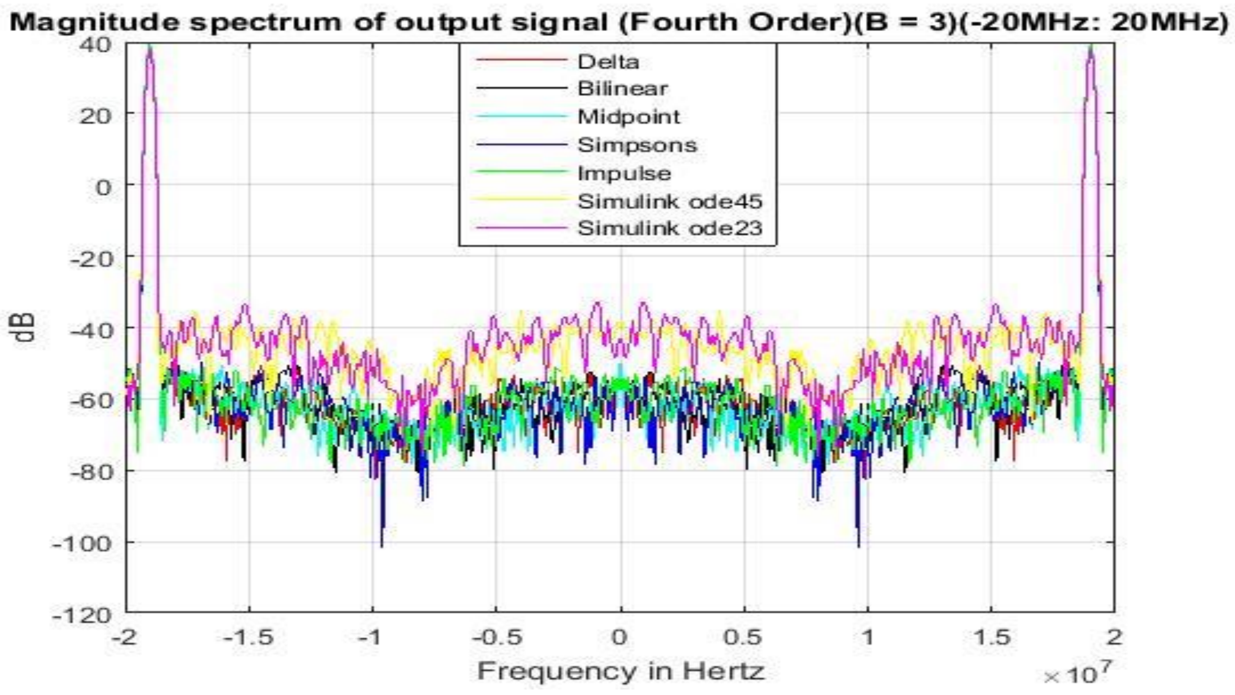
Figure 5.26: The combined output magnitude spectrum of the simulation methods for fourth-order two-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

Table 5.14 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the fourth order three-bit CT ∑ΔM simulations.

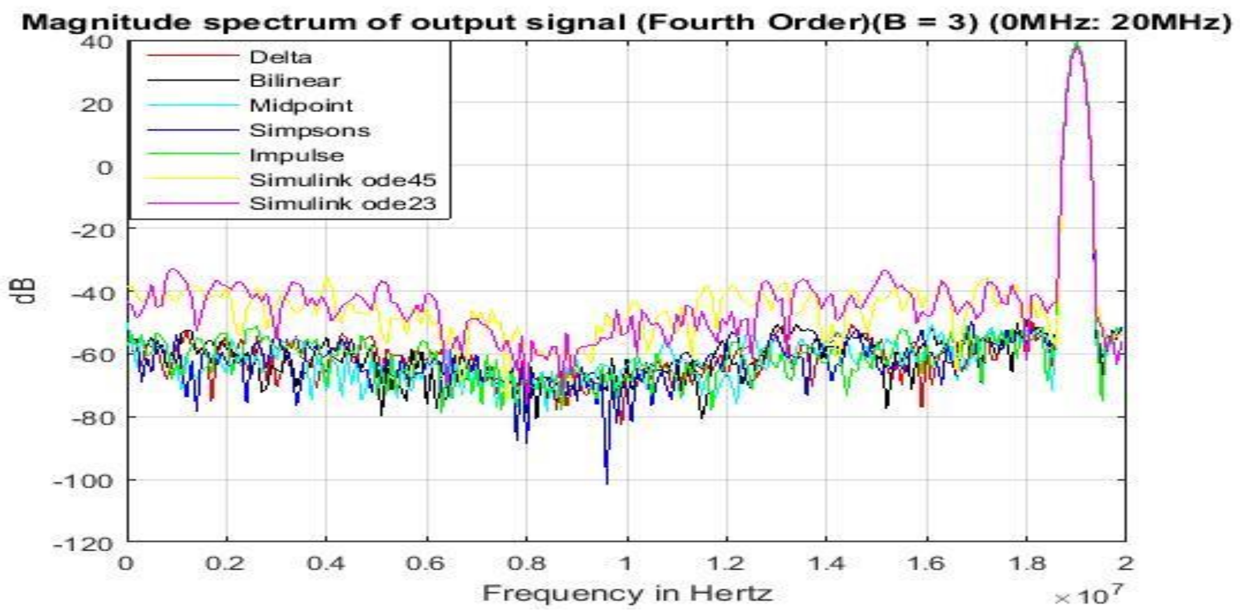
Simulation Methods	SNR (dB)	DR (dB)	Time (sec)
Delta Transform	77.75	65.95	0.60
Impulse Invariance	77.55	67.36	0.57
Midpoint Integration	77.62	66.11	0.57
Simpsons Rule	78.12	68.67	1.04
Trapezoidal Integration	77.75	65.74	0.60
Simulink (Ode45)	38.59	15.55	1.40
Simulink (Ode23)	38.27	15.51	1.21

Table 5.14: SNR, Dynamic Range and total simulation time for fourth-order three-bit CT ∑ΔM

Fig. 5.27 shows the magnitude spectrum of all the simulation methods for fourth-order three-bit CT ∑ΔMs.



a)



b)

Figure 5.27: The combined output magnitude spectrum of the simulation methods for fourth-order three-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

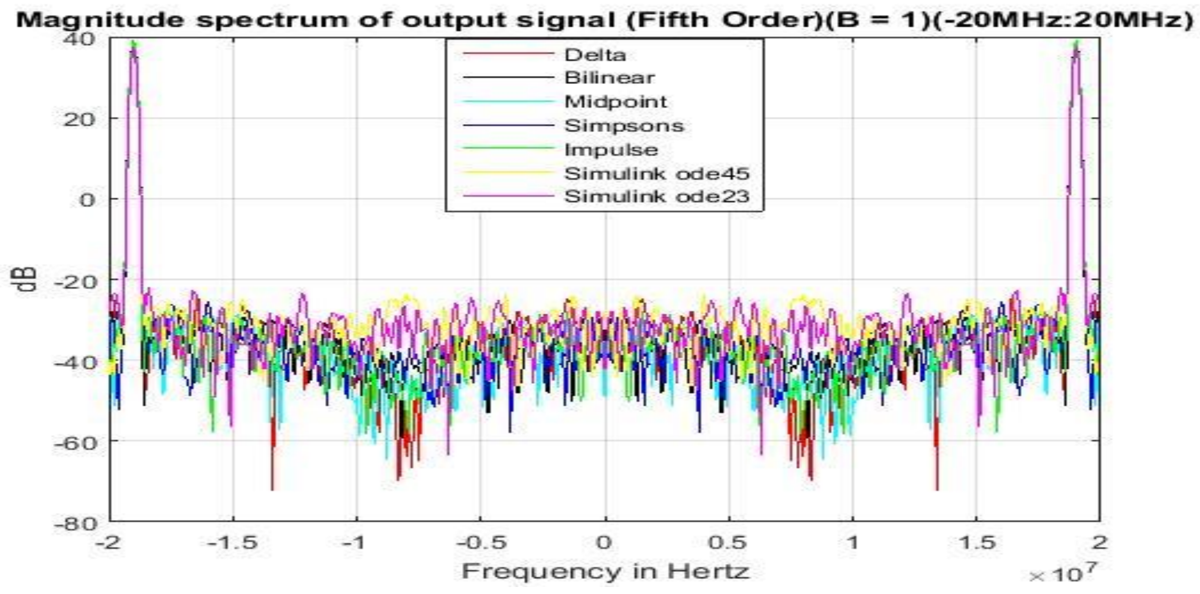
5.7.5 Fifth Order CT ∑ΔM

Table 5.15 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the fifth-order single-bit CT ∑ΔM simulations.

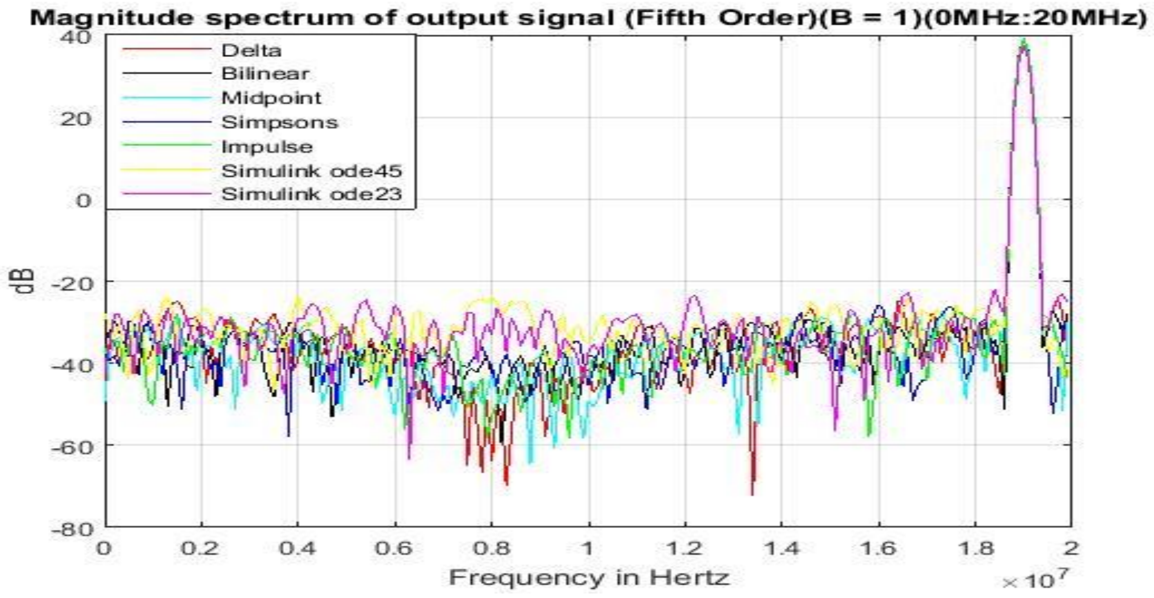
Simulation Methods	SNR (dB)	DR (dB)	Time (sec)
Delta Transform	68.16	49.74	0.67
Impulse Invariance	66.42	48.08	0.64
Midpoint Integration	66.54	47.31	0.64
Simpsons Rule	68.34	50.07	1.20
Trapezoidal Integration	66.75	47.41	0.67
Simulink (Ode45)	38.22	15.53	1.50
Simulink (Ode23)	38.36	15.62	1.29

Table 5.15: SNR, Dynamic Range and total simulation time for fifth-order single-bit CT ∑ΔM

Fig. 5.28 shows the magnitude spectrum of all the simulation methods for first order single-bit CT ∑ΔMs.



a)



b)

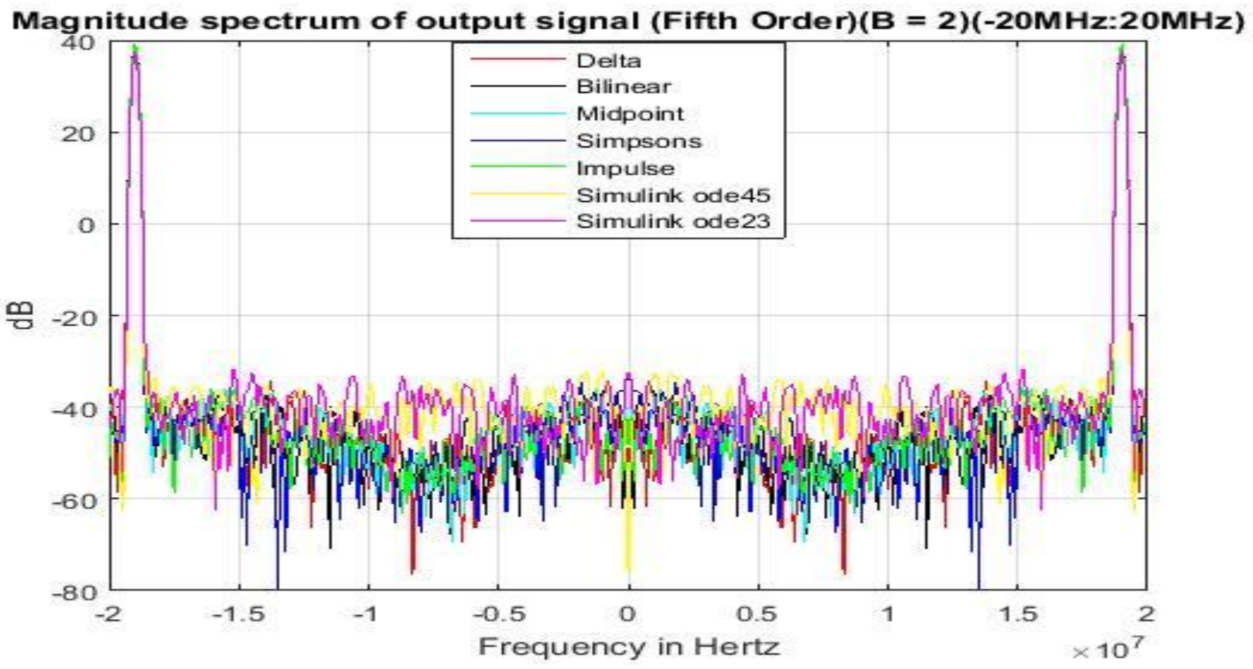
Figure 5.28: The combined output magnitude spectrum of the simulation methods for fifth-order single-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

Table 5.16 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the fifth-order two-bit CT ∑ΔM simulations.

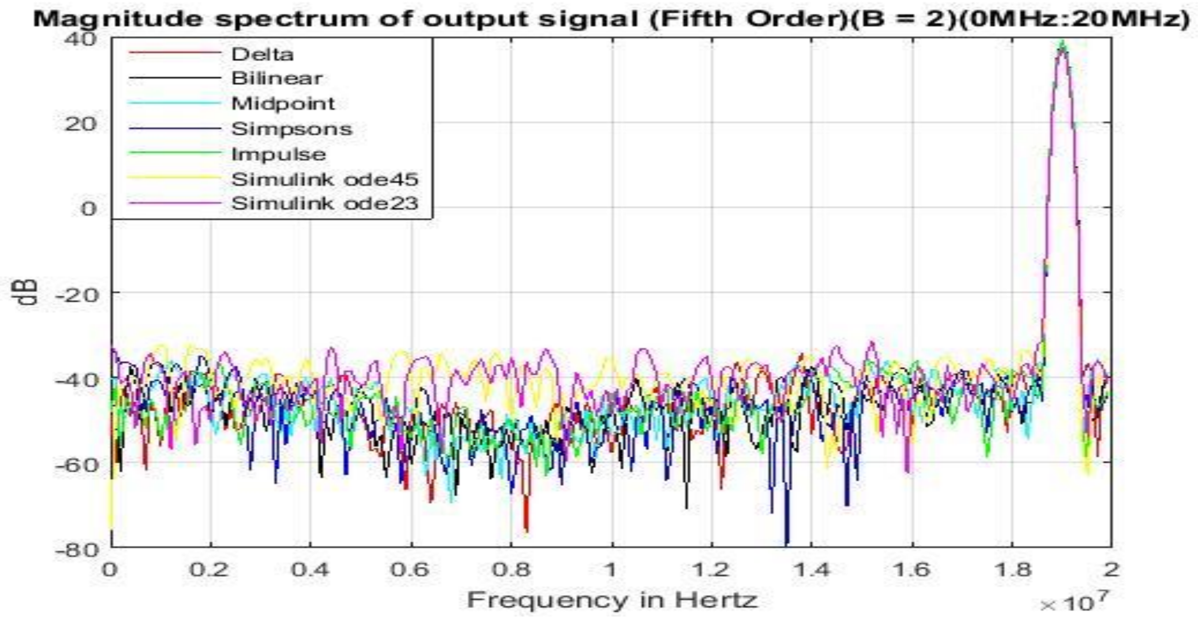
Simulation Methods	**SNR (dB)**	**DR (dB)**	**Time (sec)**
Delta Transform	74.08	58.11	0.67
Impulse Invariance	75.51	59.26	0.66
Midpoint Integration	75.38	57.01	0.64
Simpsons Rule	75.55	54.11	1.20
Trapezoidal Integration	75.47	57.74	0.66
Simulink (Ode45)	38.40	15.51	1.50
Simulink (Ode23)	38.37	15.51	1.36

Table 5.16: SNR, Dynamic Range and total simulation time for fifth-order two-bit CT ∑ΔM

Fig. 29 shows the magnitude spectrum of all the simulation methods for fifth-order two-bit CT ∑ΔMs.



a)



b)

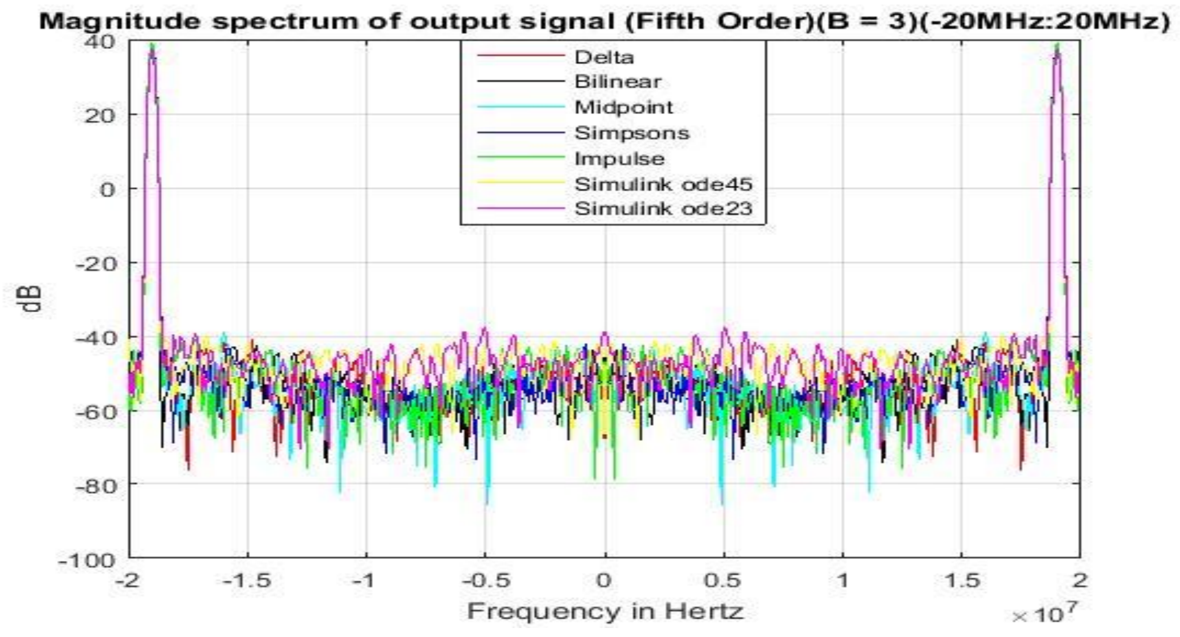
Figure 5.29: The combined output magnitude spectrum of the simulation methods for fifth-order two-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

Table 5.17 shows the signal to noise ratios, dynamic ranges and the total simulation times obtained from the fifth-order three-bit CT ∑ΔM.

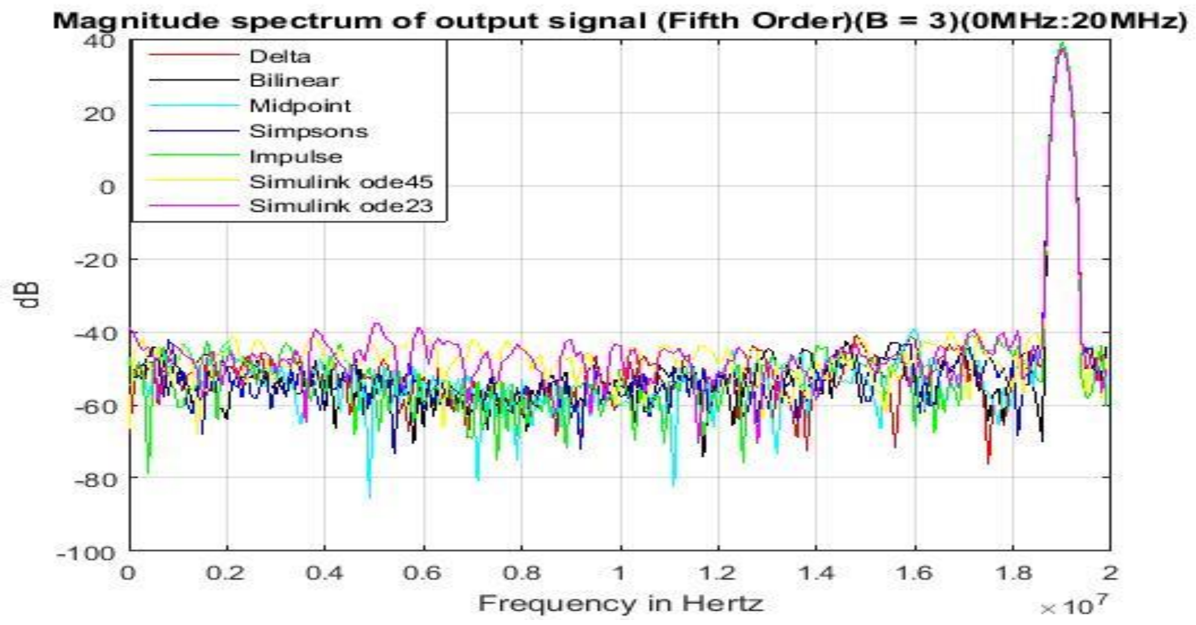
Simulation Methods	**SNR (dB)**	**DR (dB)**	**Time (sec)**
Delta Transform	80.16	61.39	0.67
Impulse Invariance	80.12	64.37	0.67
Midpoint Integration	79.49	62.94	0.66
Simpsons Rule	80.33	61.60	1.18
Trapezoidal Integration	80.05	64.66	0.68
Simulink (Ode45)	38.42	15.57	1.42
Simulink (Ode23)	38.42	15.55	1.23

Table 5.17: SNR, Dynamic Range and total simulation time for fifth-order three-bit CT ∑ΔM

Fig. 30 shows the magnitude spectrum of all the simulation methods for fifth order three-bit CT ∑ΔMs.



a)



b)

Figure 5.30: The combined output magnitude spectrum of the simulation methods for fifth-order three-bit CT ∑ΔM a) full-magnitude spectrum b) half-magnitude spectrum

5.7.6 Data Comparison

The tabulated data are compared on the basis of speed that is given by the total simulation time taken by the simulation methods and obtained signal to noise ratio in the following sections.

5.7.6.1 Based on SNR

In this section, the average calculated SNR values of five single-bit, two-bit and three-bit (first, second, third, fourth and fifth order) CT ∑ΔMs as a function of all the simulation methods has been shown in charts and the values have been compared.

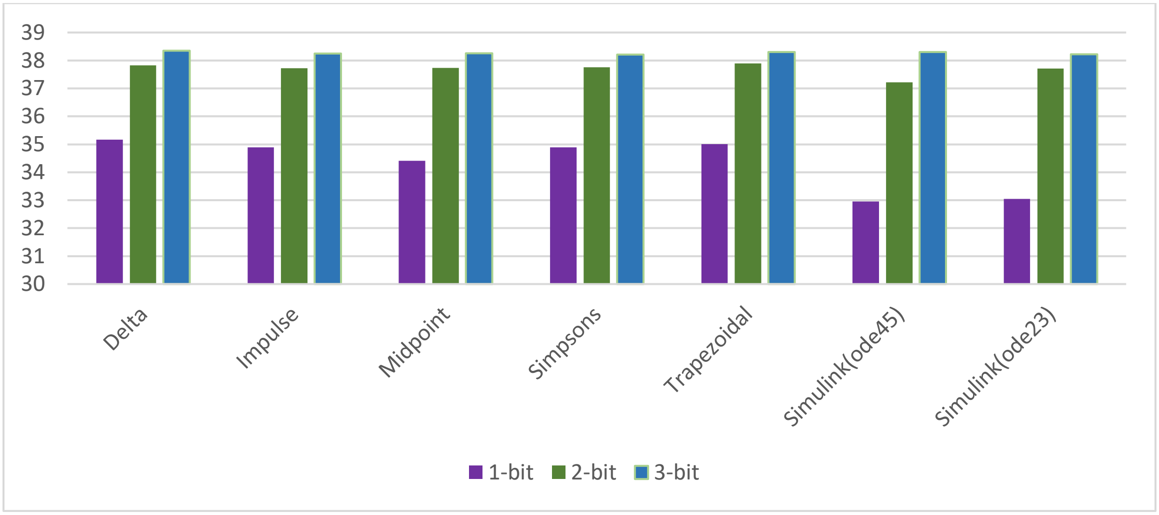


Figure 5.31: Average SNR of five single-bit, two-bit and three-bit first-order CT ∑ΔMs as a function of simulation method

The SNR values obtained from the first-order CT ∑ΔMs simulation methods do not have a noticeable difference. However, there is noticeable difference in the SNR values of single-bit, two-bit and three-bit CT ∑ΔMs. The single-bit first-order CT ∑ΔMs have the lowest SNR values while the three-bit first-order CT ∑ΔMs have the highest SNR values. This is an expected result because the value of signal to noise ratio increases with increasing number of bits. However, the SNR value using CT ∑ΔM is very high than that theoretical SNR value given by (2.15) which for three-bit ADC is around 20 dB.

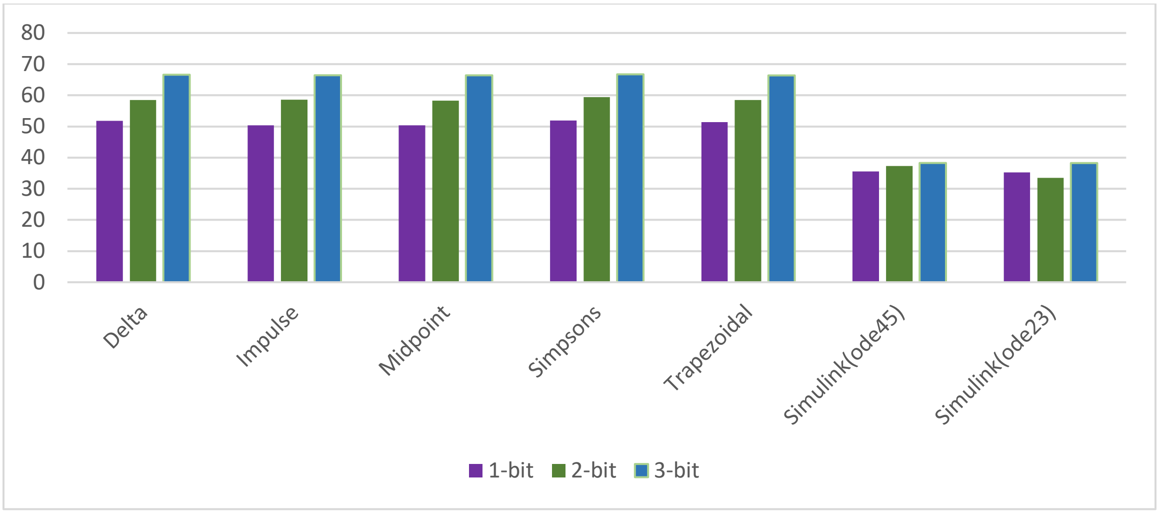


Figure 5.32: Average SNR of five single-bit, two-bit and three-bit second-order CT ∑ΔMs as a function of simulation method

The SNR values obtained from the second-order CT ∑ΔMs simulation methods show that the numerical integration methods have better SNR than both the Simulink solver models (ode45 and ode23). There is noticeable difference in the SNR values of single-bit, two-bit and three-bit CT ∑ΔMs. The single-bit CT ∑ΔMs have the lowest SNR values while the three-bit CT ∑ΔMs have the highest SNR values. This is an expected result because the value of signal to noise ratio increases with increasing number of bits.

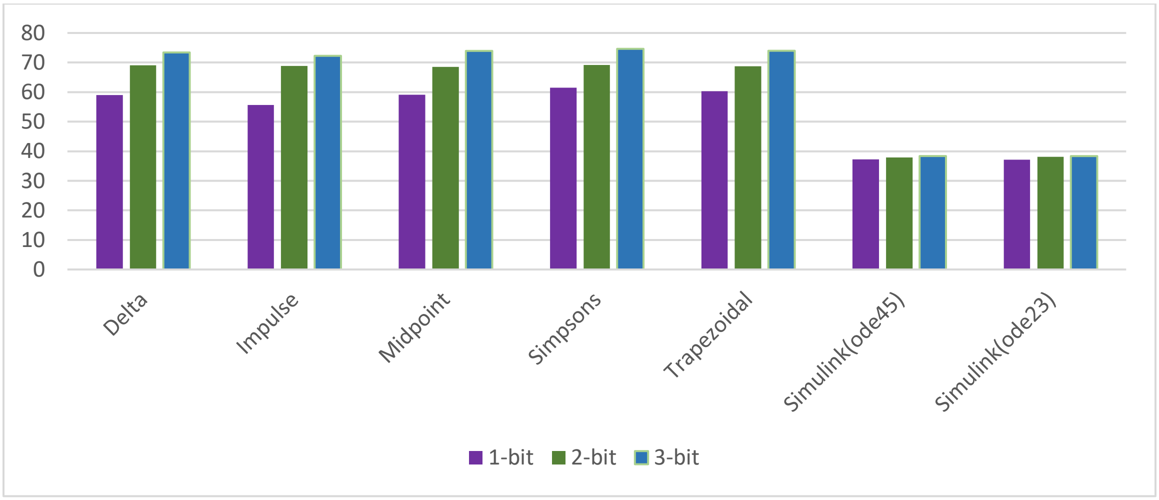


Figure 5.33: Average SNR of five single-bit, two-bit and three-bit third-order CT ∑ΔMs as a function of simulation method

The SNR values obtained from the third-order CT ∑ΔMs simulation methods also show that the numerical integration methods have better SNR than both the Simulink solver models (ode45 and ode23). The single-bit CT ∑ΔMs have the lowest SNR values while the three-bit CT ∑ΔMs have the highest SNR values.

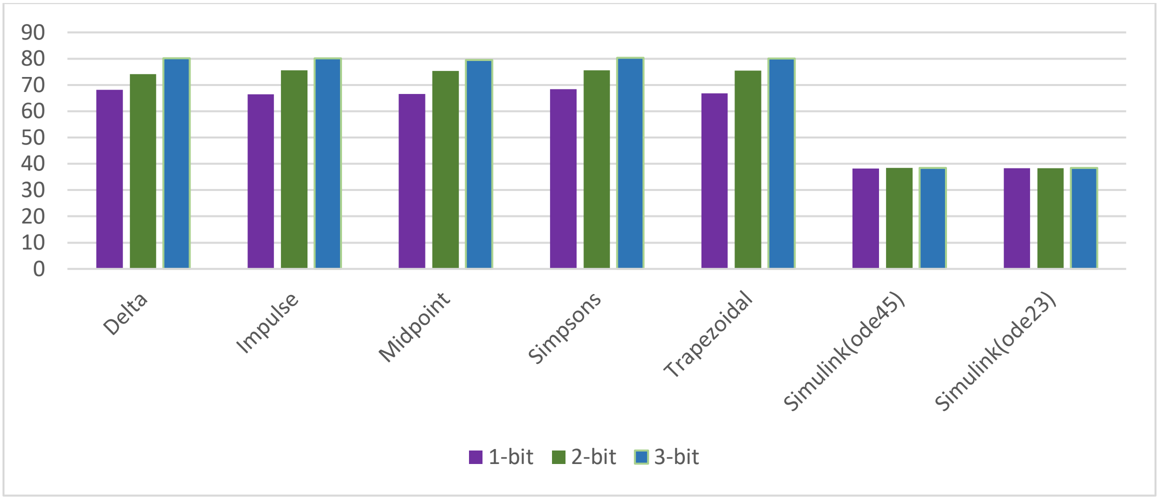


Figure 5.34: Average SNR of five single-bit, two-bit and three-bit fourth-order CT ∑ΔMs as a function of simulation method

The SNR values obtained from the fourth-order CT ∑ΔMs simulation methods also show that the numerical integration methods have better SNR than both the Simulink solver models (ode45 and ode23). The single-bit CT ∑ΔMs have the lowest SNR values while the three-bit CT ∑ΔMs have the highest SNR values.

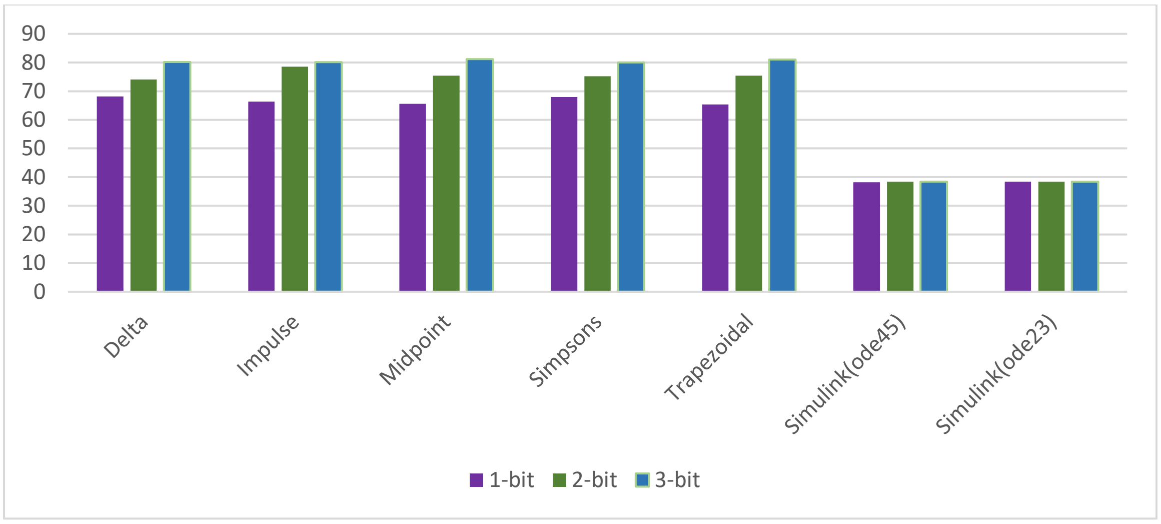


Figure 5.35: Average SNR of five single-bit, two-bit and three-bit fifth-order CT ∑ΔMs as a function of simulation method

The SNR values obtained from the fifth-order CT ∑ΔMs simulation methods also show that the numerical integration methods have better SNR than both the Simulink solver models (ode45 and ode23). The single-bit CT ∑ΔMs have the lowest SNR values while the three-bit CT ∑ΔMs have the highest SNR values.

Fig 5.36 shows the chart showing SNR values versus order of the filter by bits using delta transform.

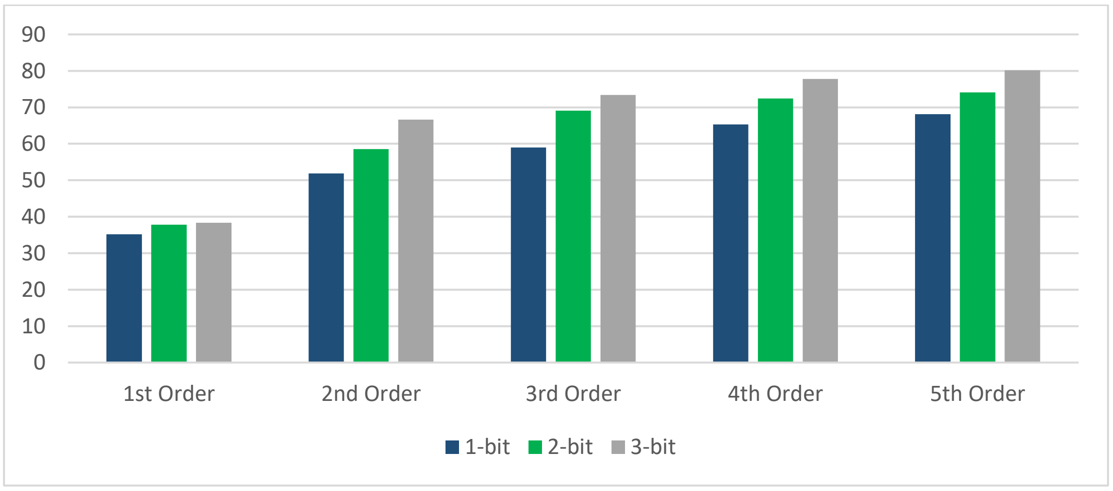


Figure 5.36: SNR vs Filter Order by bits using delta transform.

Fig 5.37 shows the chart showing the SNR values versus order of the filter by bits using impulse invariance method.

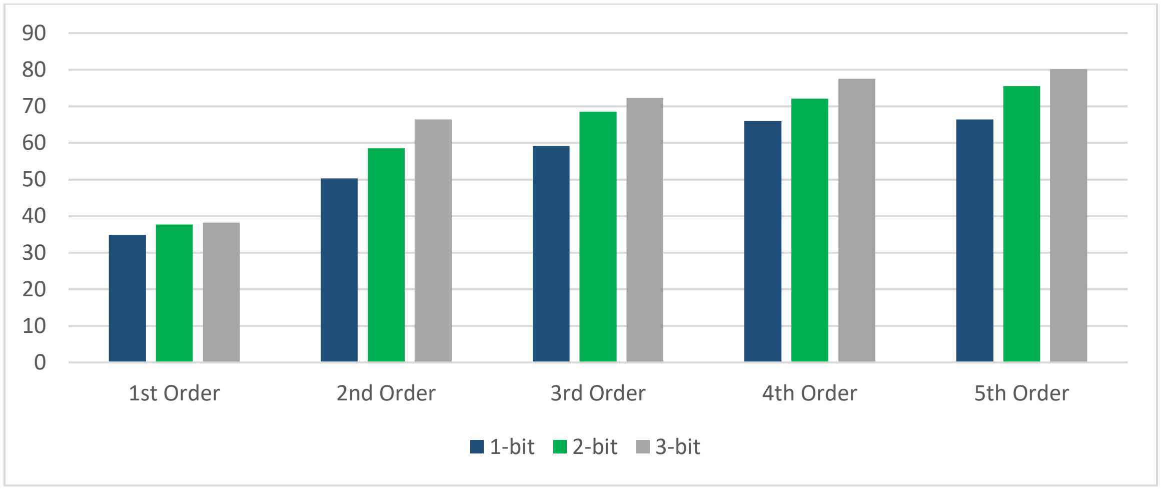


Figure 5.37: SNR vs Filter Order by bits using impulse invariance method

Fig 5.38 shows the chart showing the SNR values versus order of the filter by bits using midpoint integration.

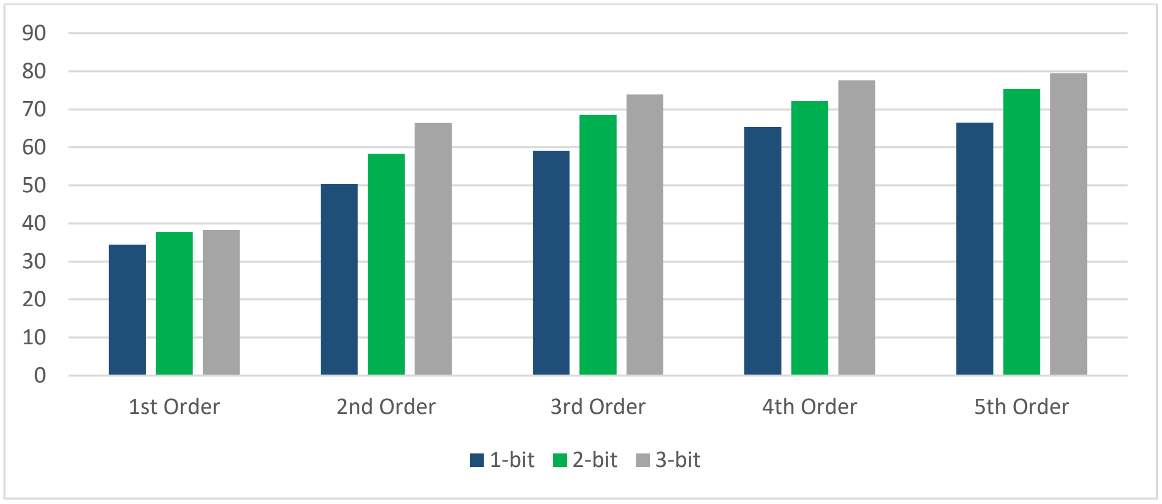


Figure 5.38: SNR vs Filter Order by bits using midpoint integration.

Fig 5.39 shows the chart showing the SNR values versus order of the filter by bits using Simpsons rule.

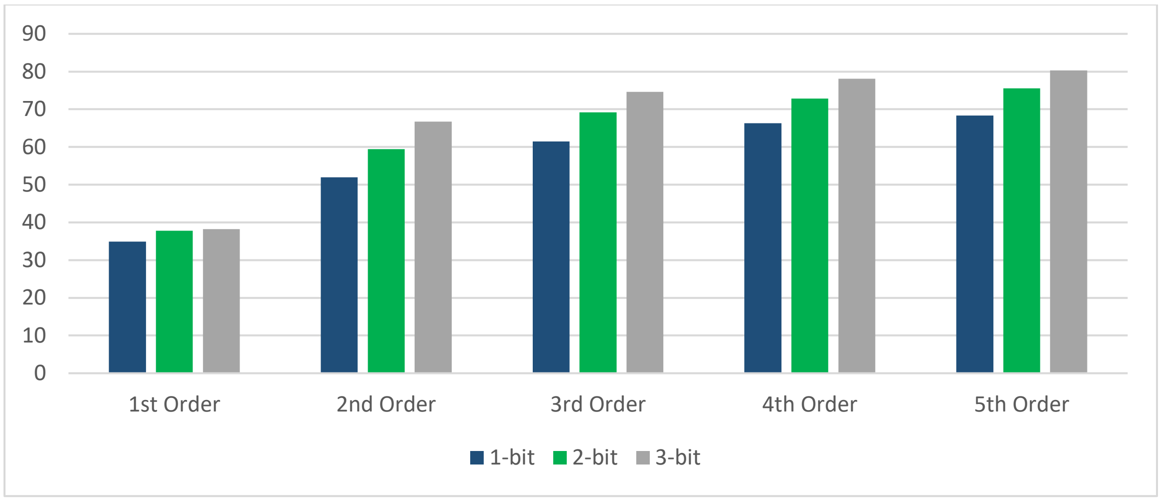


Figure 5.39: SNR vs Filter Order by bits using Simpsons rule

Fig 5.40 shows the chart showing the SNR values versus order of the filter by bits using trapezoidal integration.

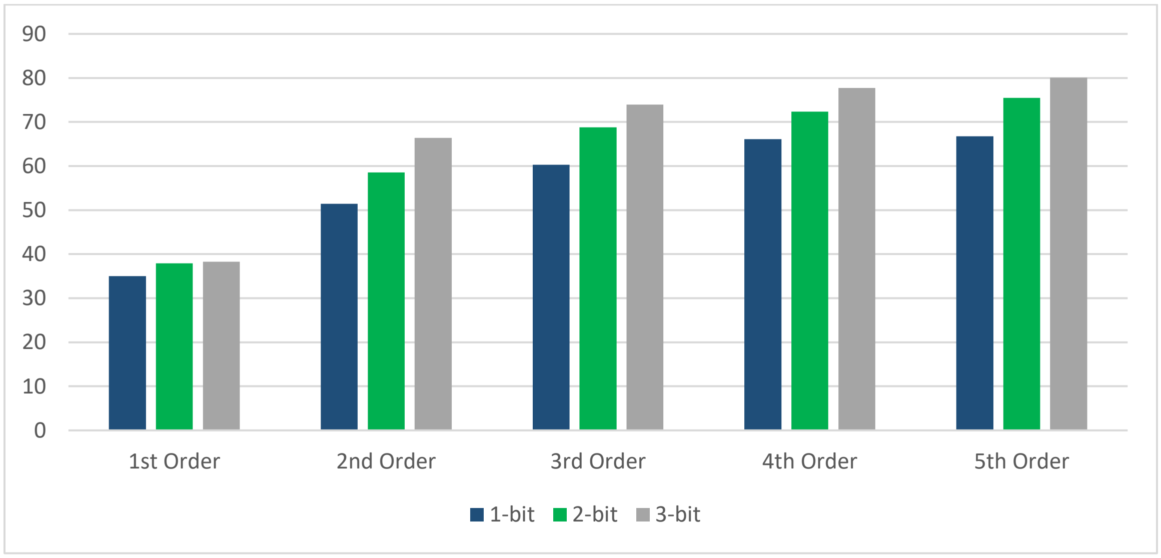


Figure 5.40: SNR vs Filter Order by bits using trapezoidal integration.

Fig 5.41 shows the chart showing the SNR values versus order of the filter by bits using Simulink(ode45) solver.

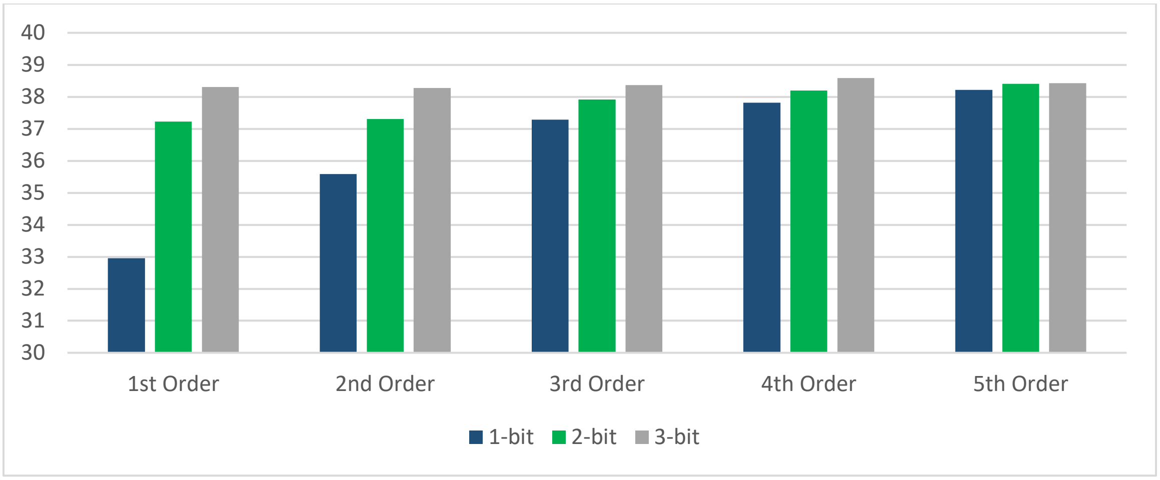


Figure 5.41: SNR vs Filter Order by bits using Simulink(ode45) solver.

Fig 5.42 shows the chart showing the SNR values versus order of the filter by bits using Simulink(ode23) solver.

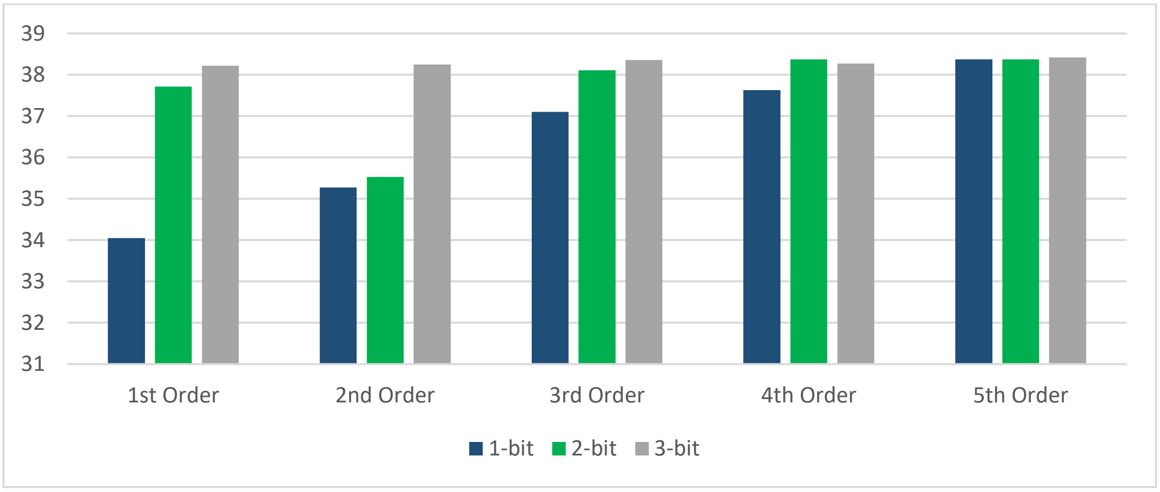


Figure 5.42: SNR vs Filter Order by bits using Simulink(ode23) solver

Fig. 4.43 shows the values of SNR values versus order of filter for one, two and three bit all CT ∑ΔM simulation methods.

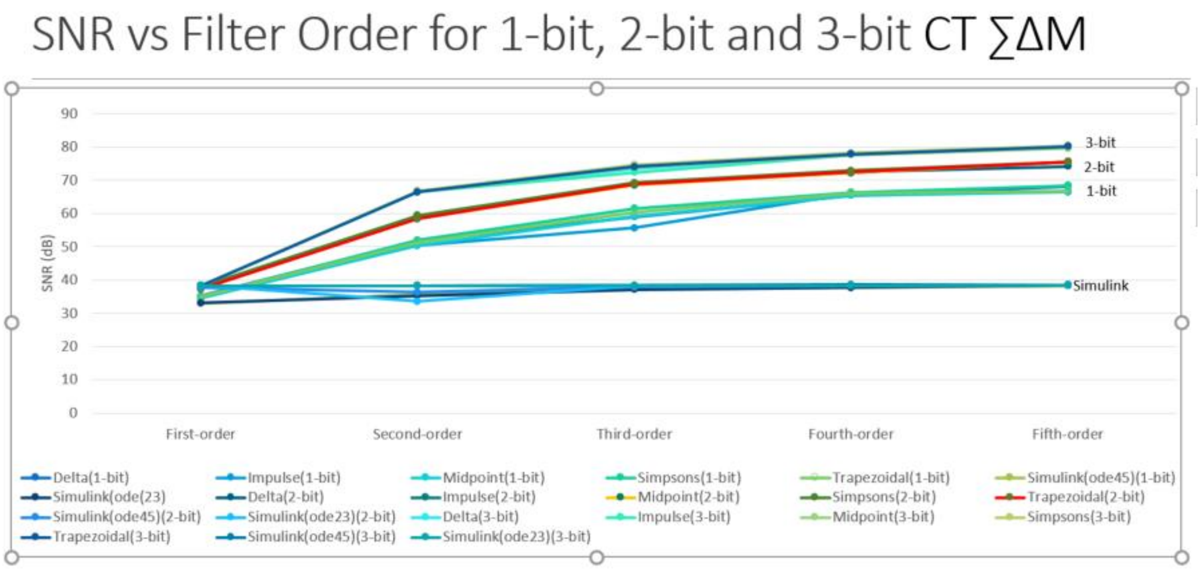


Figure 5.43: SNR vs filter order for one, two and three-bit CT ∑ΔM simulation methods

Overall comparison shows that the SNRs obtained from the numerical integration methods are better than that obtained from Simulink simulation. Among the numerical integration methods, Simpsons has better SNR than other numerical integration methods. Also, the values of SNR increase with increasing number of bits and increasing order of the filter for all the simulation methods.

5.7.6.2 Based on total Simulation Time

In this section, the average calculated simulation time values of five single-bit, two-bit and three-bit (first, second, third, fourth and fifth-order) CT ∑ΔMs as a function of all the simulation methods has been shown in charts and the values have been compared.

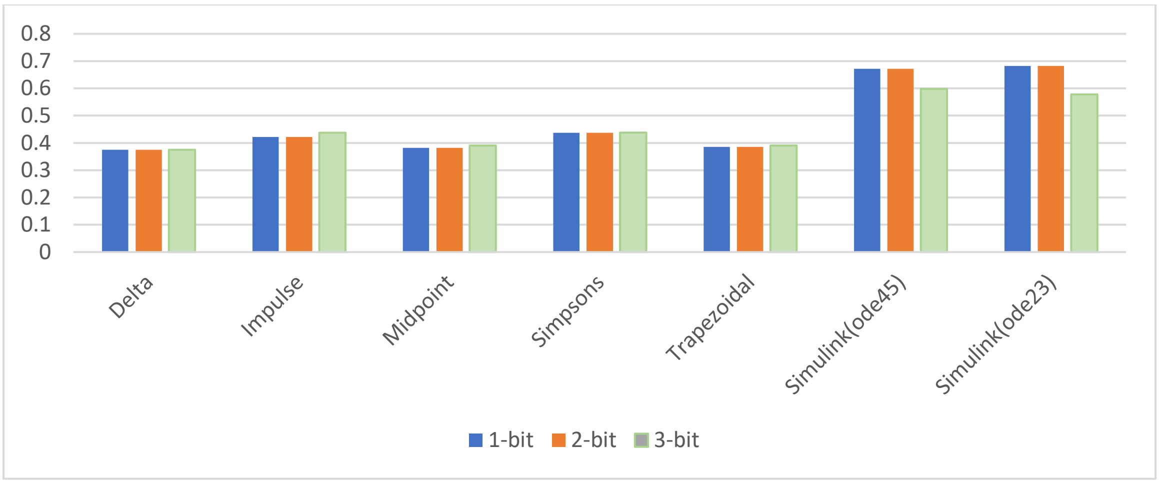


Figure 5.44: Average simulation times of five single-bit, two-bit and three-bit first-order CT ∑ΔMs as a function of simulation method

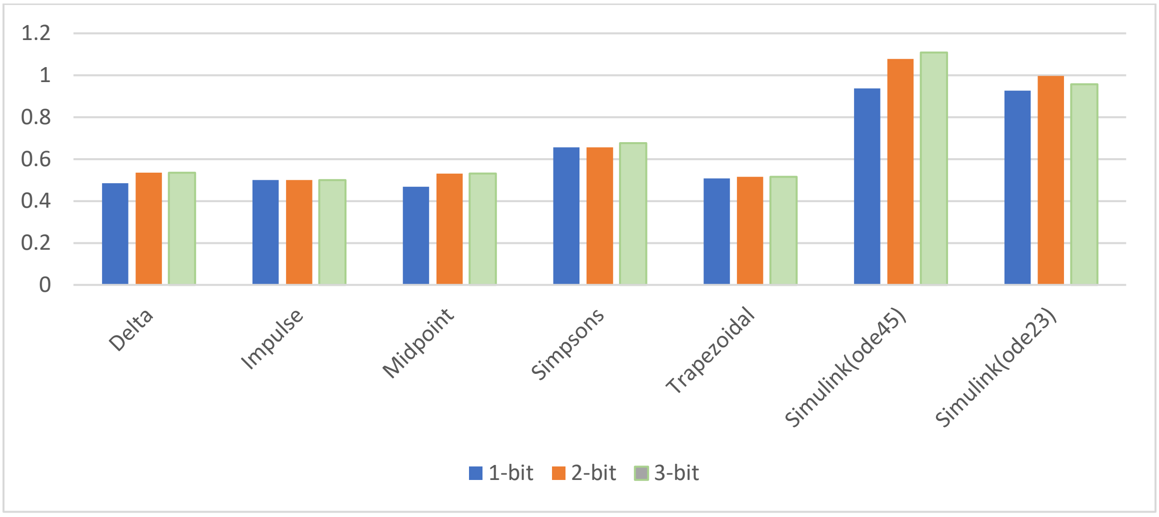


Figure 5.45: Average simulation times of five single-bit, two-bit and three-bit second-order CT ∑ΔMs as a function of simulation method

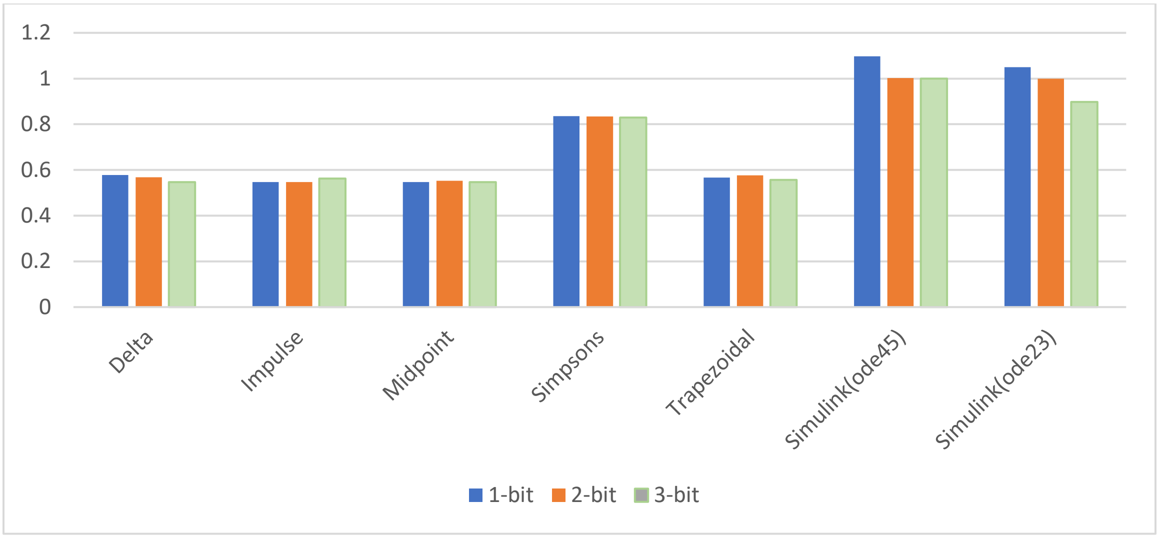


Figure 5.46: Average simulation times of five single-bit, two-bit and three-bit third-order CT ∑ΔMs as a function of simulation method

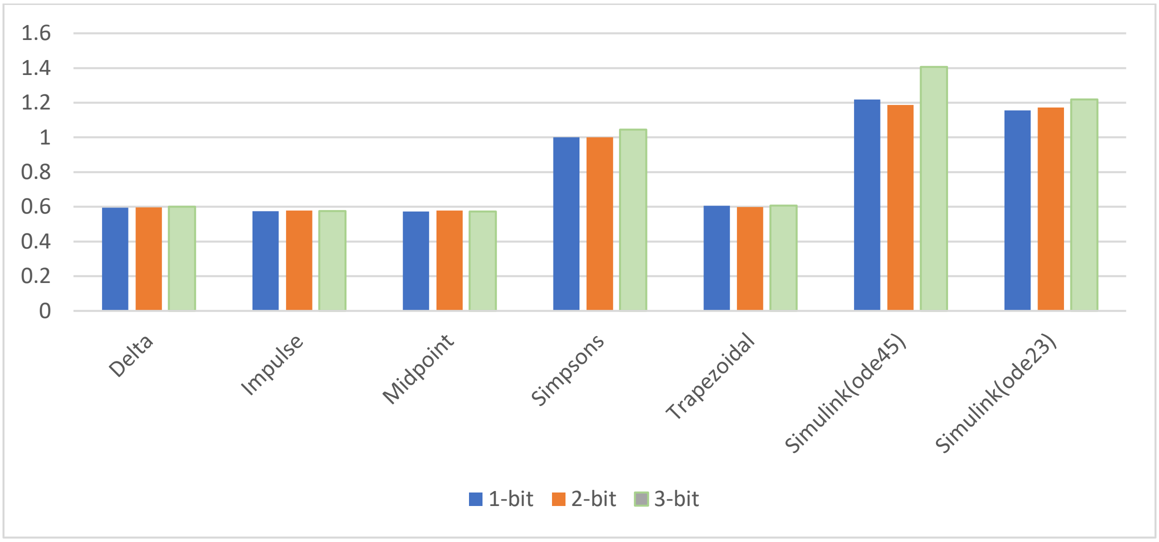


Figure 5.47: Average simulation times of five single-bit, two-bit and three-bit fourth-order CT ∑ΔMs as a function of simulation method

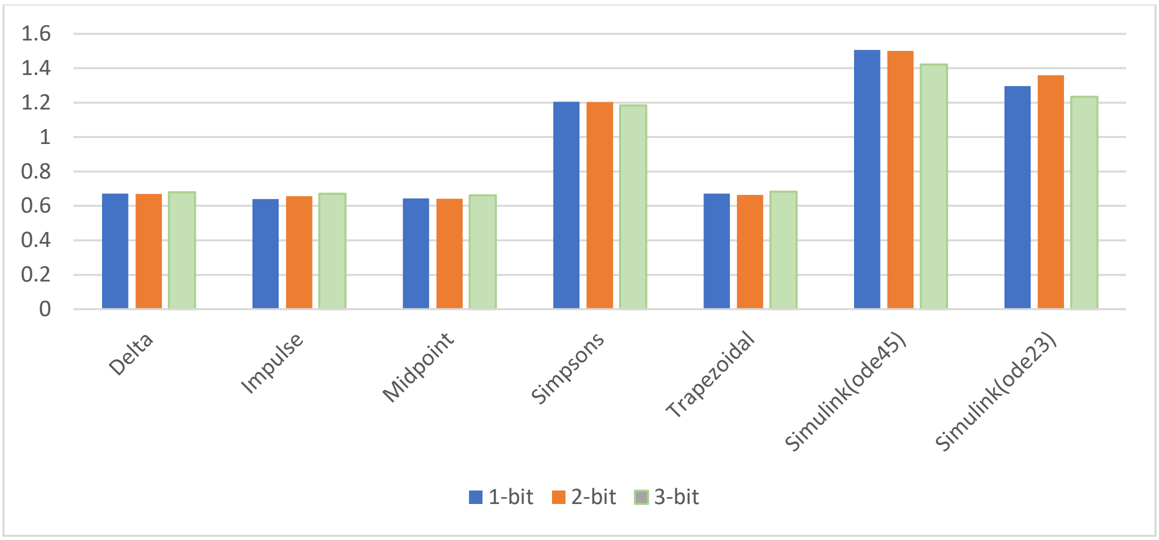


Figure 5.48: Average simulation times of five single-bit, two-bit and three-bit fifth-order CT ∑ΔMs as a function of simulation method

In all the charts, it is seen that almost all the numerical integration methods performed in almost same amount of time, but the Simulink models took quite longer simulation time. Among the numerical integration methods, Simpsons numerical integration method took maximum simulation time. In the Simulink models, Simulink ode45 solver took longer simulation time than simulation time taken by Simulink ode23 solver. Overall simulation time comparison shows that the default ode45 Simulink solver took maximum time for simulation.

5.7.6.3 Based on Simplicity

Table 5.18 shows the ranking of the simulation methods on the basis of simplicity. The simulation method with the lowest number of stars is the most difficult simulation method whereas the simulation method with the highest number of stars is the simplest simulation method.

Simulation Method	**Simplicity**
Delta Transform	***
Impulse Invariance	***
Midpoint Integration	***
Simpsons Rule	***
Trapezoidal Integration	***
Simulink (Ode45)	*****
Simulink (Ode23)	*****

Table 5.18: Comparison of simulation methods on the basis of simplicity

Among the simulation methods, Simulink was the simplest method of simulation because of the availability of required blocks in Simulink. Numerical integration methods were comparatively difficult because they required derivation of their respective *s*-domain to *z*-domain formula and after the formula derivation, they had to be implemented in MATLAB code. The overall simulation process was complicated. Simpsons numerical integration method was the most difficult simulation because it was difficult to derive the difference equations and convert it into MATLAB code.

5.7.7 Frequency domain analysis

To prove the correctness of the numerical integration *s*-domain to *z*-domain transformation formula, frequency domain analysis has been done. MATLAB's Symbolic Math Toolbox has been used to convert the *s*-domain STFs and NTFs to *z*-domain STFs and NTFs and the respective magnitude response sand phase responses of the NTFs have been plotted.

Fig. 5.49 shows the comparison of the magnitude response and phase response of *s*-domain NTF and Delta transform's *z*-domain NTF.

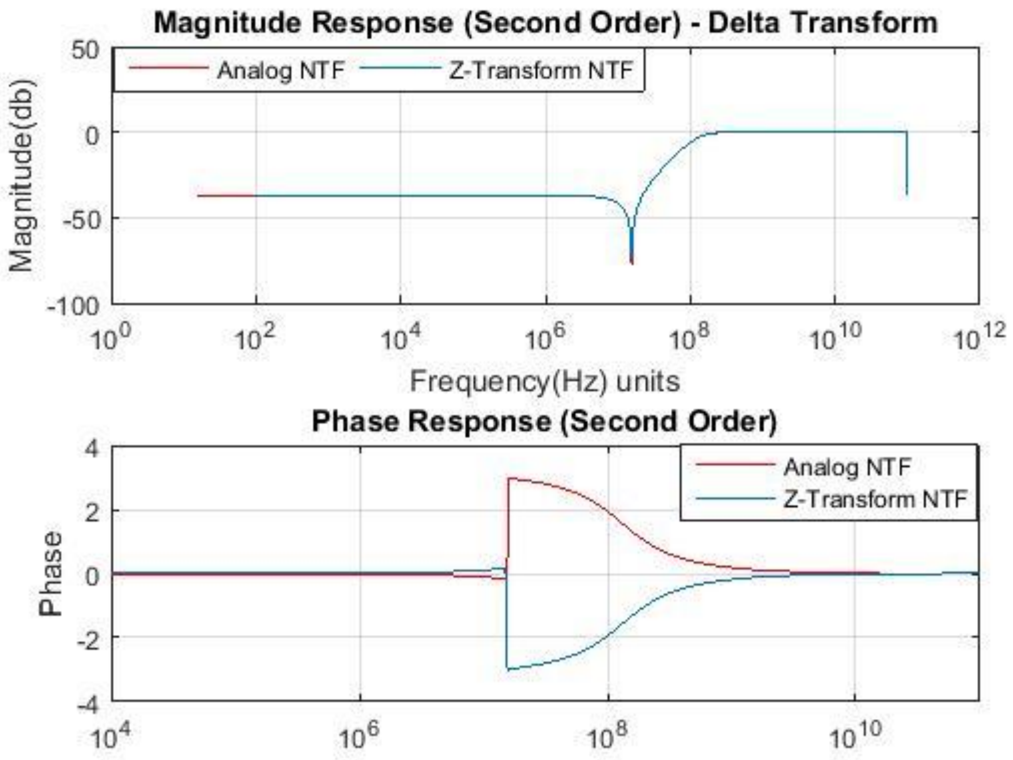


Figure 5.49: *s*-domain NTF and Delta transform's *z*-domain frequency response comparison

In Fig. 5.49, both the magnitude response and phase response of *s*-domain NTF and Delta transform's *z*-domain NTF match and this shows that the formula is correct.

Fig. 5.50 shows the comparison of the magnitude response and phase response of *s*-domain NTF and Bilinear transform's *z*-domain NTF.

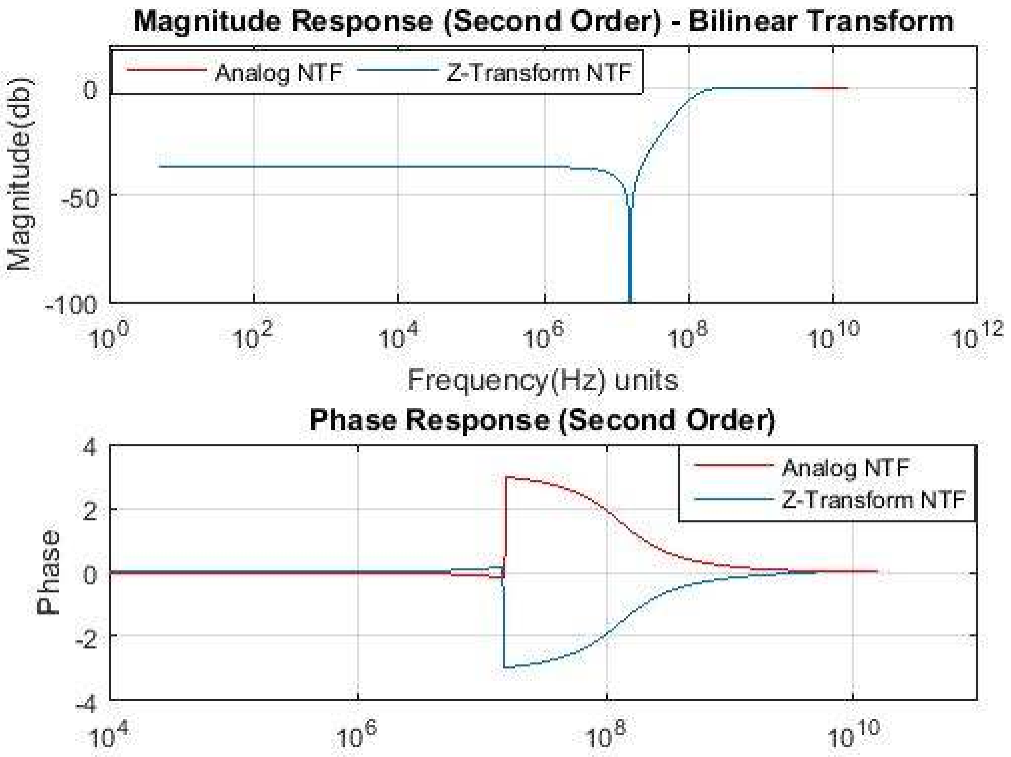


Figure 5.50: *s*-domain NTF and Bilinear transform's *z*-domain frequency response comparison

In Fig. 5.50, both the magnitude and phase response of *s*-domain NTF and Bilienar transform's *z*-domain NTF match and this shows that the formula is correct.

Fig. 5.51 shows the comparison of the magnitude and phase response of *s*-domain NTF and Midpoint Integration's *z*-domain NTF.

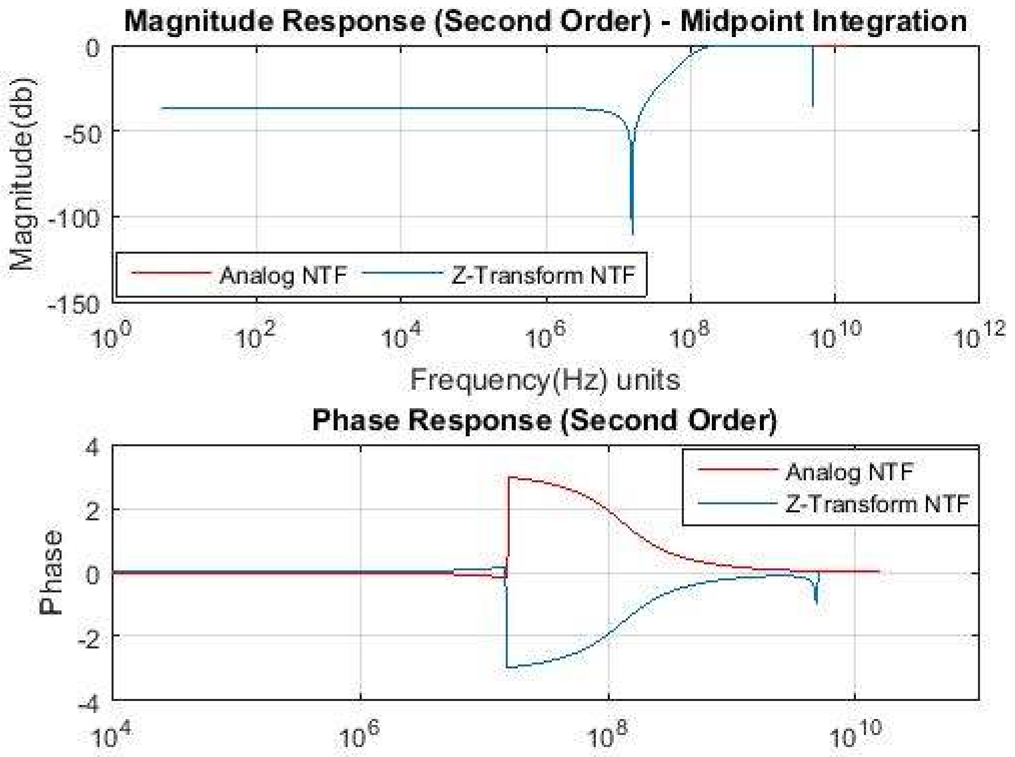


Figure 5.51: *s*-domain NTF and Midpoint integration's *z*-frequency response comparison

In Fig. 5.51, both the magnitude and phase response of *s*-domain NTF and Midpoint integration's *z*-domain NTF match and this shows that the formula is correct.

Fig. 5.52 shows the comparison of the magnitude and phase response of *s*-domain NTF and Simpsons rule's *z*-domain NTF.

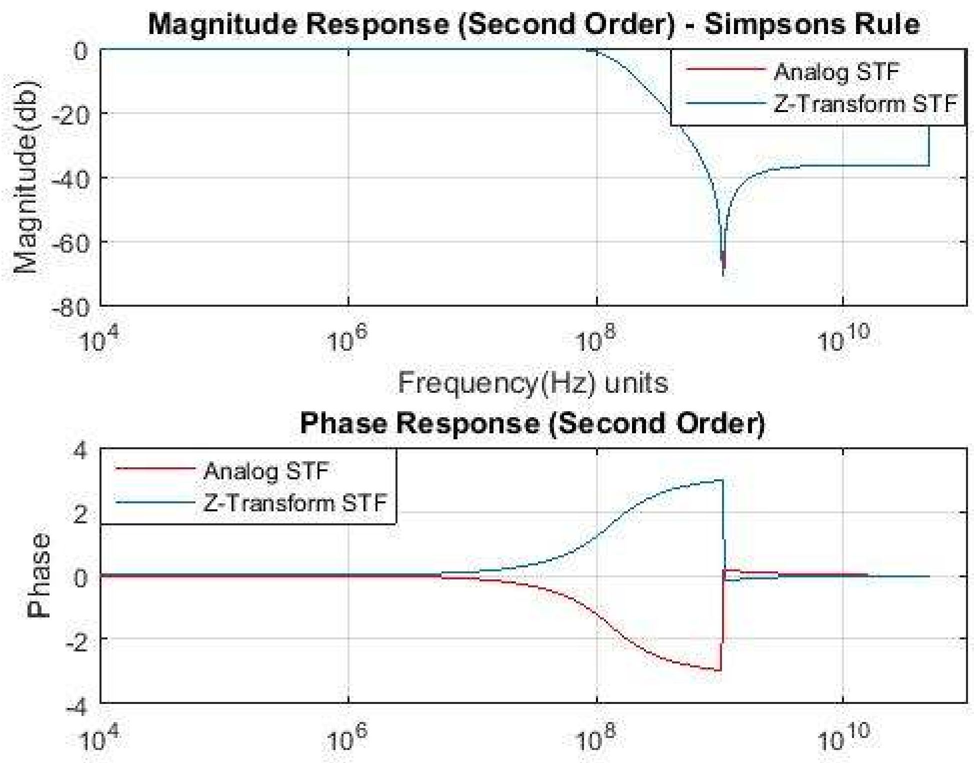


Figure 5.52: *s*-domain NTF and Simpsons integration's *z*-domain frequency response comparison

In Fig. 5.52, both the magnitude and phase response of *s*-domain NTF and Simpsons rule's *z*-domain NTF match and this shows that the formula is correct.

Fig. 5.53 shows the comparison of the magnitude and phase response of *s*-domain NTF and Impulse Invariance transformation's *z*-domain NTF.

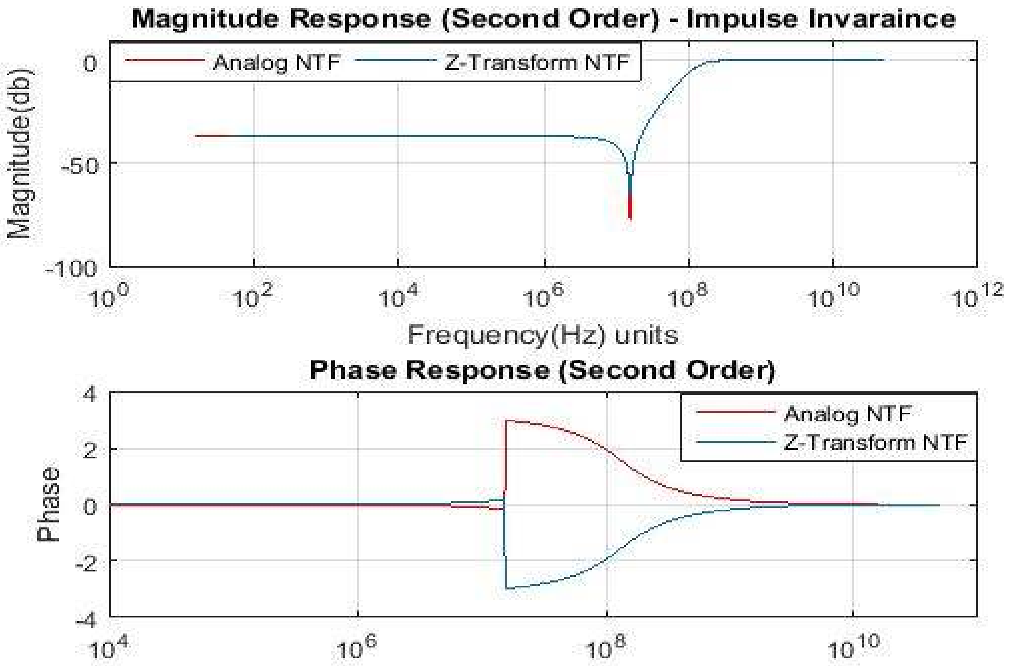


Figure 5.53: *s*-domain NTF and Impulse Invariance Transformation's *z*-domain frequency response comparison

In Fig. 5.53, both the magnitude and phase response of *s*-domain NTF and Impulse Invariance transformation's *z*-domain NTF match and this shows that the formula is correct.

Thus, the above magnitude and phase comparison plots prove the correctness of the derived *s*-domain to *z*-domain transformation formulas.

Chapter 6

CONCLUSION AND FUTURE WORK

In this thesis, methods for simulating single-bit and multi-bit CT ∑ΔMs are developed. The various methods of simulations include the bilinear transform or trapezoidal integration, impulse invariance transform, midpoint integration, Simpson's rule, delta transform or Euler's forward integration rule and Simulink (ode45 and ode23) modeling. These methods are compared with respect to simulated signal to noise ratio, dynamic range, total elapsed time, frequency response and performance which includes accuracy, simplicity, and speed of the simulation method. The CT ∑ΔMs are extended from first order up to fifth order with one, two and three bit quantizers. Also, the frequency domain analysis is done for all the orders of CT ∑ΔMs in order to prove the correctness of the transformation formulas.

This thesis describes how the numerical integration methods and Simulink can be used in CT ∑ΔMs simulations. The correctness of all the numerical integration methods is proved by comparing the magnitude reponses and phase responses of the *s*-domain NTFs and *z*-domain NTFs which is obtained by using the numerical integration *s*-domain to *z*-domain formulas. The numerical integration methods had better SNR than Simulink models (ode45 and ode23). Among the numerical integration method, Simpson's rule had better SNR. Along with that, the numerical integration simulation methods were also faster than Simulink simulations. However, CT ∑ΔM simulation using Simulink was simpler because of the availability of the required blocks. But while comparing the overall advantages, numerical integration methods performed better than Simulink models.

Currently, the proposed work just shows how various numerical integration methods and how Simulink can be used to simulate one-bit, two-bit and three-bit first, second, third, fourth and fifth-order CT ∑ΔMs. It would be interesting to model various non-idealities such as operational amplifier noise, clock jitter, integrator non-idealities, finite DC gain, slew rate, finite bandwidth, amplifier saturation and transconductor nonlinearities that are associated with the CT ∑ΔMs.

Another significant area to work on is the stability of the CT ∑ΔMs. It would be interesting to determine the stability criteria and do the stability analysis of the CT ∑ΔMs using methods such as analytical root locus.

Also, since this thesis is limited to just fifth-order CT ∑ΔMs, it would be interesting to work on even higher orders CT ∑ΔMs.

BIBLIOGRAPHY

[1] Jurij F. Tasic, Mohamed Najim and Michael Ansorge, Intelligent Integrated Media Communication Techniques: COST 254 & COST 276

[2] Benabes, Philippe, Mansour Keramat, and Richard Kielbasa. "A methodology for designing continuous-time sigma-delta modulators." Proceedings of the 1997 European conference on Design and Test. IEEE Computer Society, 1997.

[3] Kang, Kyung. "Simulation, and Overload and Stability Analysis of Continuous Time Sigma Delta Modulator." (2014).

[4] Ortmanns, Maurits, Friedel Gerfers, and Yiannos Manoli. "Compensation of finite gain-bandwidth induced errors in continuous-time sigma-delta modulators." IEEE Transactions on Circuits and Systems I: Regular Papers 51.6 (2004): 1088-1099.

[5] Cherry, James A., and W. Martin Snelgrove. Continuous-time delta-sigma modulators for high-speed A/D conversion: theory, practice and fundamental performance limits. Vol. 521. Springer Science & Business Media, 1999.

[6] Panda, Soumya Shatakshi. "A 15 bit third order Power optimized Continuous time Sigma Delta modulator for audio application."

[7] Jackson, Matthew Edward. Optimal design of discrete-time delta sigma modulators. Diss. University of Nevada, Las Vegas, 2009.

[8] Cho, Soo-Hwan, et al. "Comparison of analog-to-digital filter conversion methods in a digital flickermeter." Electrical Engineering 91.3 (2009): 125..

[9] Rabiner, Lawrence R., and Bernard Gold. "Theory and application of digital signal processing." Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p. (1975)..

[10] Parks, Thomas W., and C. Sidney Burrus. Digital filter design. Wiley-Interscience, 1987.

[11] Gregorian, Roubik, and Gabor C. Temes. "Analog MOS integrated circuits for signal processing." New York, Wiley-Interscience, 1986, 614 p. (1986).

[12] Unbehauen, Rolf, and Andrzej Cichocki. MOS switched-capacitor and continuous-time integrated circuits and systems: analysis and design. Springer Science & Business Media, 2012.

[13] Gerfers, Friedel, and Maurits Ortmanns. Continuous-time sigma-delta A/D conversion: fundamentals, performance limits and robust implementations. Vol. 21. Springer Science & Business Media, 2006.

[14] Balagopal, Sakkarapani, et al. "Design-to-testing: a low-power, 1.25 GHz, single-bit single-loop continuous-time\Delta\Sigma modulator with 15 MHz bandwidth and 60 dB dynamic range." Analog Integrated Circuits and Signal Processing 90.3 (2017): 625-638.

[15] Inose, Hi, Y. Yasuda, and J. Murakami. "A telemetering system by code modulation-δ-σmodulation." IRE Transactions on Space Electronics and Telemetry 3 (1962): 204-209.

[16] Candy, James. "A use of double integration in sigma delta modulation." IEEE Transactions on Communications 33.3 (1985): 249-258.

[17] Breems, Lucien, and Johan Huijsing. Continuous-time sigma-delta modulation for A/D conversion in radio receivers. Vol. 634. Springer Science & Business Media, 2001.

[18] Kang, Kyung, and Peter Stubberud. "A comparison of continuous time sigma delta modulator simulation methods." Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium on. IEEE, 2014.

[19] Radjen, Dejan, et al. "A low-power 2nd-order CT delta–sigma modulator with an asynchronous SAR quantizer." Analog Integrated Circuits and Signal Processing 84.3 (2015): 409-420.

[20] Charmin, Asghar, and Esmaeil Najafi Aghdam. "A low power reconfigurable multi-mode continuous time Delta Sigma modulator for seven different mobile standards with VCO-based quantizer." Analog Integrated Circuits and Signal Processing 90.2 (2017): 321-331.

[21] Li, Yamei, and Lili He. "First-order continuous-time sigma-delta modulator." Quality Electronic Design, 2007. ISQED'07. 8th International Symposium on. IEEE, 2007.

[22] Talebzadeh, Jafar, and Izzet Kale. "A General Formula for Impulse-Invariant Transformation for Continuous-Time Delta-Sigma Modulators." (2017).

[23] Zheng, Geng, Saraju P. Mohanty, and Elias Kougianos. "Design and modeling of a continuous-time delta-sigma modulator for biopotential signal acquisition: Simulink vs. Verilog-AMS perspective." Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on. IEEE, 2012.

[24] Benabes, Philippe, and Catalin-Adrian Tugui. "Effective modeling of CT functions for fast simulations using MATLAB-Simulink and VHDLAMS applied to Sigma-Delta architectures." Circuits and Systems (ISCAS), 2011 IEEE International Symposium on. IEEE, 2011.

[25] Webb, Matthew, and H. U. A. Tang. "System-level simulation for continuous-time delta-sigma modulator in MATLAB Simulink." Proceedings of the 5th WSEAS Int. Conf. on Circuits, Systems, Electronics, Control & Signal Processing, Dallas, USA. 2006.

CURRICULUM VITAE

Graduate College

University of Nevada, Las Vegas

Benju Koirala

benzu.koirala@gmail.com

Degrees:

Bachelor of Electronics and Telecommunications Engineering 2013

Tribhuvan University, Nepal

Thesis Title:

Comparison of Simulation Methods of Single and Multi-bit Continuous Time Sigma Delta Modulators

Thesis Examination Committee:

Chairperson, Dr. Peter Stubberud, Ph.D.

Committee Member, Dr. Ebrahim Saberinia, Ph.D.

Committee Member, Dr. Sahjendra Singh, Ph.D.

Graduate Faculty Representative, Dr. Ajoy K. Datta, Ph.D