E C O L E
P O L Y T E C H N I Q U E
D E T U N I S I E

Engineering internship report

Option SISY

Continuous time simulator of Sigma Delta converter in C langage

Hosting laboratory : ASIM

written by : KAMMOUN ABLA
Supervised by : MR ABOUSHADY HASSAN
MR BEILLEAU NICOLAS

Academic year 2004/2005

Acknowledgements

I would like to thank Mr Alain Greiner, the director of the ASIM department, for the opprtunity he has granted me to work in his laboratory.

I wish also to express my gratitude to my supervisors, Mr Hassan Aboushady and Mr Nicolas Beilleau for their help and availability.

Contents

List of Figures

Introduction

The data transmission and signal processing fields have known a tremendous development due to the progress of the digital circuits technology. As most natural sources of information are analog, the analog-digital conversion has become necessary to transform anlog information into digital format. Analog-digital conversion is performed by discretizing the analog signal first in time then in amplitude.

According to Shannon theorem, The time discretization doesn't entail any loss of information if the sampling frequency is more than twice the nyquist rate. However, the amplitude discretization implies some irreversible loss of information. Therefore, throughout the last decennies, researches have been conducted in order to produce ADC with the least possible quantization noise.

Nowadays, Sigma Delta converters are the most qualified converters to meet this requirement. But the theoretical study of these non linear systems didn't yield satisfactory results for designers. Thats'why simulations are still the only method to ensure sigma delta converters stability.[1]

Sigma Delta modulators are usually simulated using MATLAB/Simulink and the signal processing toolbox. But as Matlab is an interpreted langage, simulations require a lot of time. The objective of my engineering internship is then to elaborate a C library for simulating sigma delta modulators.

After giving a brief presentation of the hosting laboratory, I will begin in the first chapter of this report by giving an overview of the different converters.Then, the second chapter presents the ex-theoretical analysis that had been carried out in order to prove some interesting results about $\Sigma\ \Delta$ converters.After that the files' organiztion of the C simualtor will be presented in the third chapter. Finally, the last chapter compares the simulation results of the C simulator with that of the Matlab simulator.

Presentation of the hosting laboratory

I have worked in my engineering internship in the ASIM Department of the LIP6 Laboratory.

Presentation of the LIP6 laboratory

The LIP6 Laboratory is founded in January 1997 by the union of three laboratories which are:

- LAFORIA
- LITP
- MASI

The activities of the LIP6 Laboratory covers a great variety of ingineering fileds like microelectronics, network and performance analysis, Computer Algebra etc...

The lip6 laboratory has enhanced the creation of 2 centers of research which are:

- CERME(Centre Europen de recherche en Microlctronique)

  The Center for European Research in Micro-Electronics (CERME) has been created by the Universit Pierre et Marie Curie (UPMC) and the Centre National de la Recherche Scientifique (CNRS) in january 2002, to support the research in the field of CAD tools and methods for ”System on Chip” design.

- EURONETLAB

  EuronetLab is a joint research laboratory that brings together LiP6 (and, through it, the Universit Pierre et Marie Curie and the CNRS), the ENST, Thales, and 6WIND.

  It has been created to develop solutions adapted to real-world telecommunication systems.

Presentation of the ASIM department

The ”Architecture des systèmes intégrés et Micro électroniques” departement of the LIP6 laboratory focuses on the design of highly complex integrated circuits, the development of advanced CAD tools for circuits and integrated systems. It has also enhanced the creation of four startups which are:

- Tachys Thechnologies

  (specialized in providing innovative, high-speed interconnect building blocks for data communications equipment.)

- AED(created in 1997 by 3 researchers of the ASIM Department)

  (Design of integrated circuits)

- AVERTEC(created in 1998 by 5 researchers of the ASIM Department)

  (Development of CAO tools)

- Surf technology

  (specialized in multimedia indexation)

Chapter 1

An overview of the different converters

1.1 The simple quantizer

The diagram of the simple quantizer is depicted below. As it is shown in fig 1.1, the quantizer can be modelized as an additive noise in the system.[2]

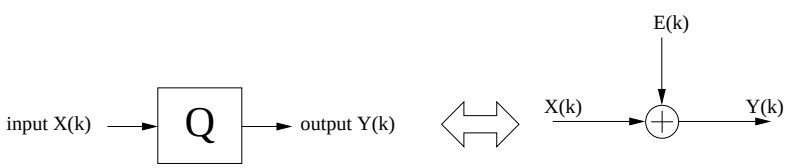


Figure 1.1: the diagram of the simple encoder

According to Bennett, the quantization noise can be assumed to be a a uniform white quantization, under the following conditions:[3]

1-the quantizer doesn't overload

2-the quantizer has a large number of levels

3-the distance between the levels is small

4-the probability distribution of pairs of input samples is given by a smooth probability density function.

Assuming that the quantization noise is uniformly distributed and uncorrelated with the input signal (fig 1.2),the variance of the noise is then equal to:

$$\sigma^2 = \frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} q^2 dq = \frac{\Delta^2}{12} \tag{1.1}$$

where Δ is the quantization step.[4]

The noise power spectrum density is then equal to:

$$E(f) = \frac{\sigma^2}{fe} = \frac{\Delta^2}{12fe} \tag{1.2}$$

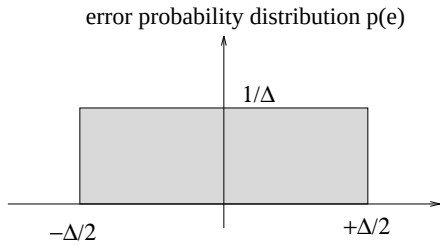


Figure 1.2: the noise distribution

where fe is the sampling frequency. The power in the band of interest is then given by the next formula:

$$P = \int_{-fm}^{fm} E(f)df = \frac{\Delta^2}{12OSR} \tag{1.3}$$

where fm denotes the maximum frequency of the signal and OSR the oversampling ratio.

$$OSR = \frac{fe}{2fm} \tag{1.4}$$

Therefore

$$P(db) = 10 * log_{10}(\frac{\Delta^2}{12OSR})$$

If the encoder samples data at the nyquist rate, adding one bit to the quantizer devises the quantization step Δ by 2 and decreases the quantization noise by 6 dB. In fact

$$P(db) = 10 * log_{10}(\frac{\Delta^2}{12fe})$$

Therefore, the variation of the noise power is

$$\Delta P(db) = -10 * log_{10}(4) = -6db$$

However, when operating at the nyquist rate, we need a very high number of components of high accuracy in order to achieve high resolution ADC. Furthermore, we need to use an anti-aliasing filter with a sharp cutoff in ordre to eliminate all signal components with a frequency higher than the nyquist rate.(fig 1.3)

On the other hand, in oversampled ADCs, the used anti-aliasing filter has in general a smooth slope, which reduces a lot its design complexity. (fig 1.4)

Besides, the reduction of the noise quantization power can be made possible by only increasing the oversampling ratio OSR. For example, when doubling OSR, the quantization noise power is reduced by 3dB. In fact when doubling OSR the variation of the signal noise is

$$P(db) = -10 * log_{10}(2) = -3db$$

1.2 The sigma delta encoder: the predictive encoder

The principle of the sigma delta converter is to encode the error quantification rather than the input signal.[2].

Called also the noise shaping encoders, ΣΔ converters have the advantage of pushing the signal noise outside of the band of interest.(fig1.5)

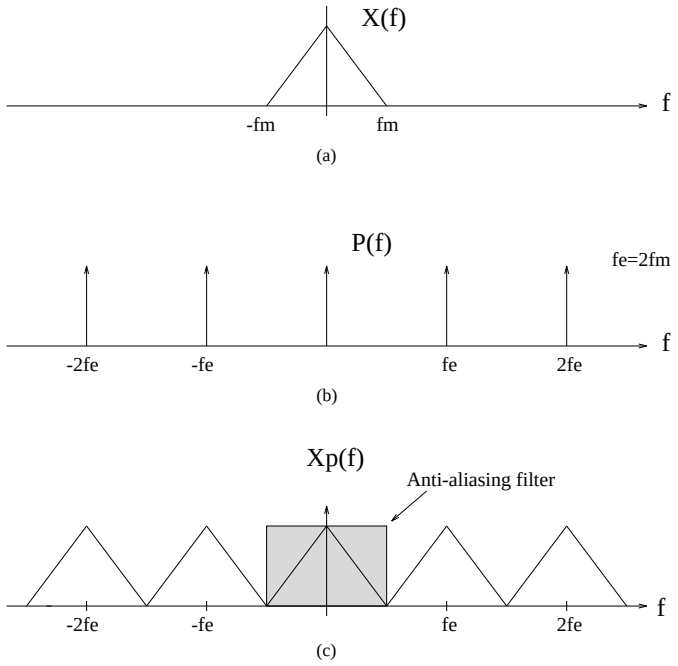


Figure 1.3: Sampling at the nyquist frequency:(a)Signal input spectrum,(b)Spectrum of the sampling function (c)Spectrum of the sampled signal

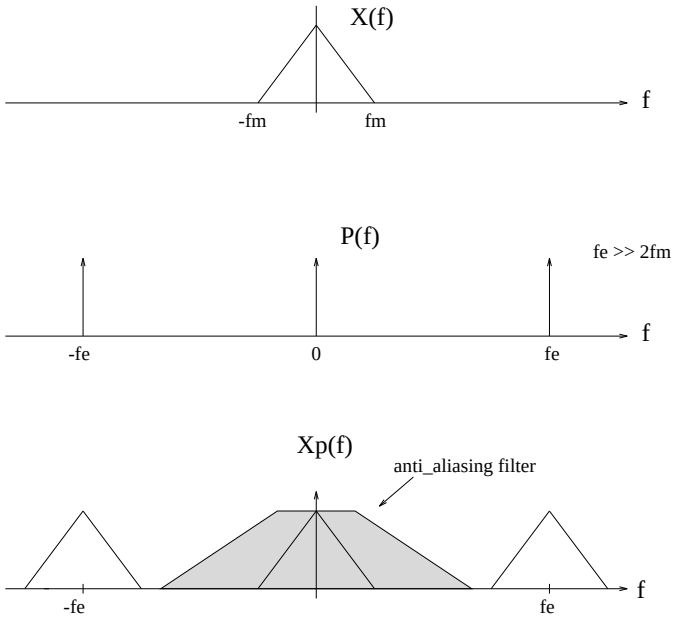


Figure 1.4: Oversampling: the anti-aliasing filter has a smooth slope

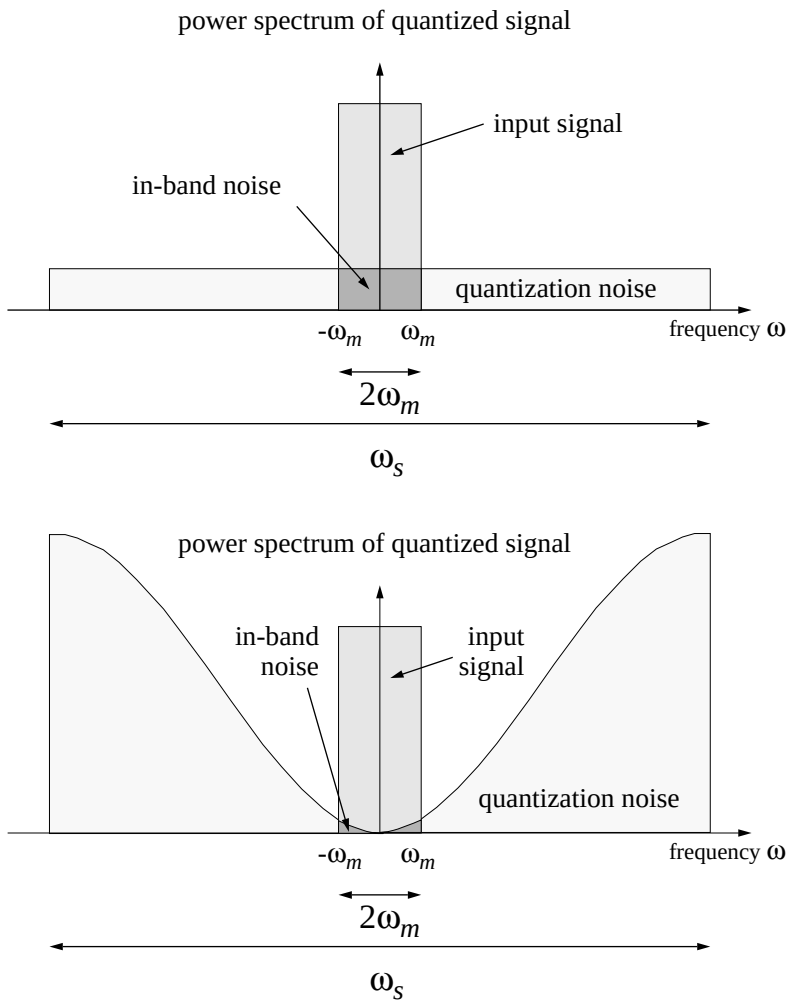


Figure 1.5: Power spectrum of the output of oversampled A/D conversion. (a) simple encoding (b) Σ Δ converter

In this report we will focus on the following ΣΔ configurations:

1. The single loop modulator

   The diagram of the single loop modulator is depicted in (fig1.6)

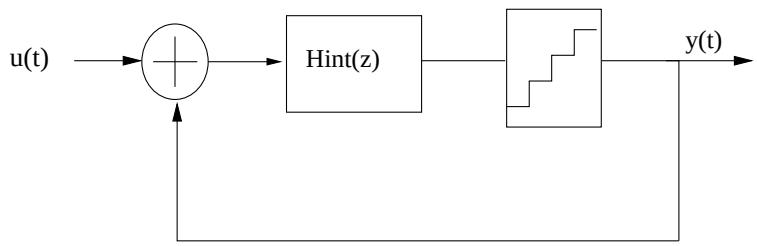


Figure 1.6: the diagram of the single loop converter

Examples:

- CIFF(cascade of integrators Feedforward) architecture:(fig1.7)

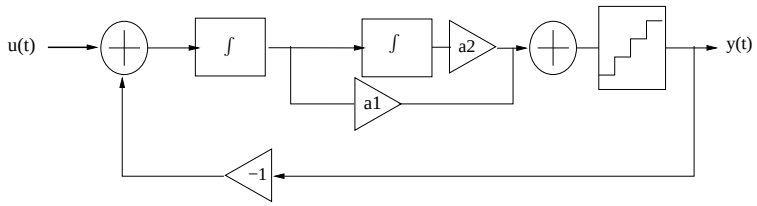


Figure 1.7: a second order CIFF architectured sigma delta converter

- CRFF(cascade of Resonators Feedforward):(fig1.8)

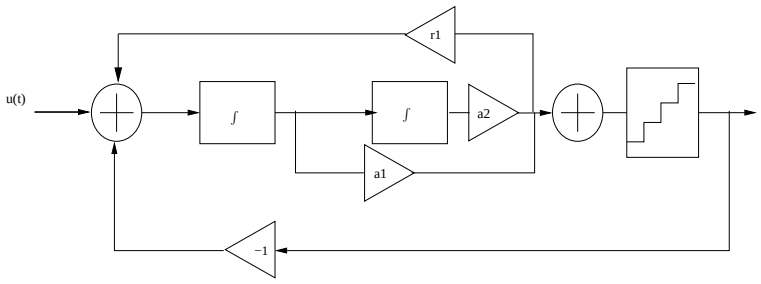


Figure 1.8: a second order CRFF architectured sigma delta converter

2. The multi loop modulator

   The diagram of the multi loop modulator is depicted in (fig1.9)

Examples:

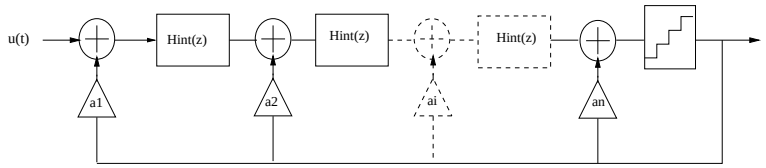


Figure 1.9: General diagram of multiloop converter

- CIFB architecture (fig1.10)

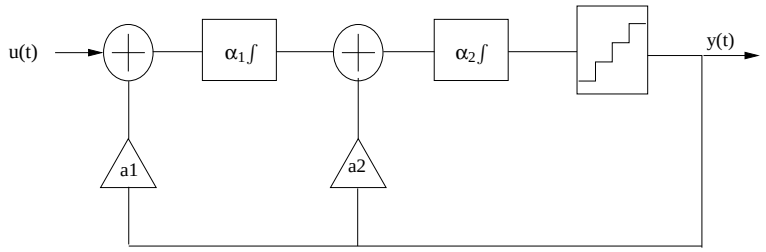


Figure 1.10: A second order CIFB architectured modulator

- CRFB architecture (fig1.11)

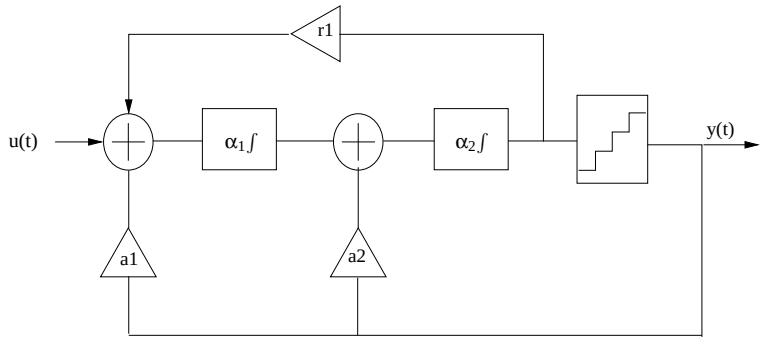


Figure 1.11: A second order CRFB architectured modulator

A $\Sigma\Delta$ converter could be then characterized by:

- its order: the order of a $\Sigma\Delta$ converter is equal to the number of integrators.
- its architecture
- The number of levels of its quantizer

Chapter 2

$\Sigma\,\Delta$ converters: Theoretical analysis and techniques of design

There are two different approachs for the analysis of a sigma delta converter which are

- The linear modelization of the quantizer:

  This analysis is based on linearizing $\Sigma\,\Delta$ modulators and yields some interesting theoretical results like the SNR theoretical expression.

- The linear gain model:

  Modeling the quantizer as a source of an additive noise isn't sufficient to predict the converter stability. Another method based on determining the root locus is rather used in order to determine whether a sigma delta converter is stable or not.

  In the following, we will focus on the description of the first theoretical anlysis.

2.1 The linear modelization of the quantizer

This approach consists in modeling the quantizer as a source of additive noise in the system. According to this appraoch, The general diagram of a $\Sigma\,\Delta$ converter is depicted in (fig2.1)

The output of the modulator can be then expressed as the sum of two terms, one containing the input signal, and the other containing the noise:

$$V(z) = G(z)U(z) + H(z)E(z)$$

where G and H are respectively the signal and noise transfer functions.

As this diagram shows, the main parts of a $\Sigma\,\Delta$ converter are:

- the loop filter: which is a linear system with two inputs U and V, where U denotes the analog input to the system and V denotes the output of the DAC.

- the quantizer: which is a nonlinear system whose input Y comes from the loop filter.

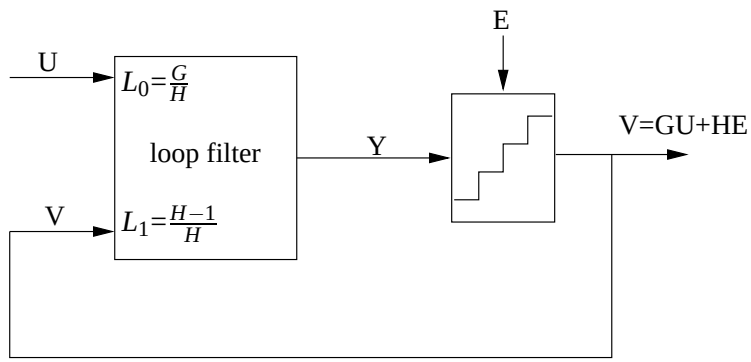


Figure 2.1: a General diagram of single quantizer $\Sigma\,\Delta$ converter

The choice of the quantization noise plays a significant role in the achievable performance of the converter.

While the attenuation of the NTF(H(z):noise transfer function) in the band of interest is provided by its zeros, the out of band gain is controlled by its poles. Reducing the out of band gain improves stability but decreases the signal to noise ratio.

The design requirements for the NTF could be resumed into 3 criteria:

1. system realizability

   The loop around the quantizer (H(z)-1) must be strictly causal. This constraint forces:

   $$lim_{z\rightarrow+\infty}H(z)=1;$$

2. An important in band noise attenuation:

   The Schreier toolbox offers two methods for shaping the quantization noise.

   - a direct method: this method considers coincident zeros.
   - an optimized method: this method consists in spreading the zeros across the band of interest. We have considered in the C simualtor the discrete coefficients returned by Matlab using the direct method.

   (a) low pass converters

   Using the direct method, The NTF has the next form:

   $$NTF=\frac{(z-1)^n}{\prod_{i=1}^{i=n}(z-p_i)}$$

   where p_i is the ith pole of the NTF and n is the degree of the modulator. Thus, for a nth order modulator, we have n degrees of freedom. In order to reduce the number of degree of freedom to 1, we consider that the NTF poles are arranged into an inverse chebychev configuration, so that the single degree of freedom becomes the cutoff frequency.

   (b) pass band converter

   For pass band converter, the NTF has the next formula:

   $$NTF=\frac{(z-2\pi f0)^n}{\prod_{i=1}^{i=n}(z-p_i)}$$

where p_i is the ith pole of the NTF, n is the degree of the modulator and f0 is the center frequency for the modulator.

3. system stability

   Simulations have been proven that having $|H(jw)| < 2$ for monobit converters and 3.5 for multibit converters is necessary to ensure stability.

2.2 Techniques for producing Σ Δ converters

There are two techniques for producing Σ Δ converters:

- The first technique consists in using Discrete time circuits: the converter input is sampled.(fig2.2)

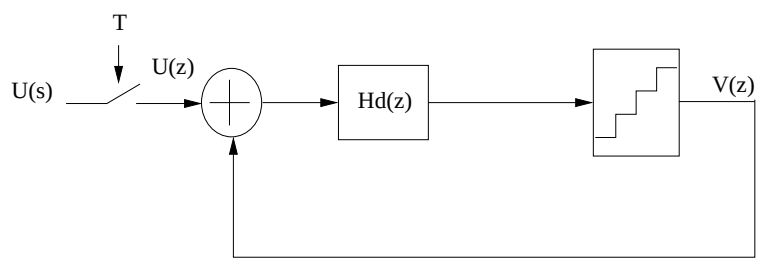


Figure 2.2: a Discrete time Σ Δ converter

- The second technique consists in using continuous time circuits: the sampling operation takes place inside the modulation loop (just before the quantization).(fig2.3)

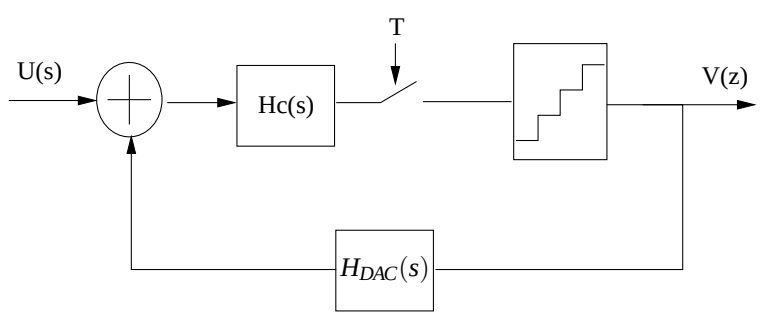


Figure 2.3: a continuous time Σ Δ converter

Discrete time converters are easier to design and simulate, but they are more sensitive to sampling errors. In fact, the sampling errors for CT converters are suppressed by the high gain of the loop in the passband. Therefore, CT converter can operate at higher sampling rate.[5] CT Σ Δ converter coefficients could be determined by applying a transfomation to the DT Σ Δ converter coefficients.[5]

2.3 Determination of the theoretical expression of the noise's power

Let's take the example of a discrete first order modulator. (fig2.4) where E(z) is a

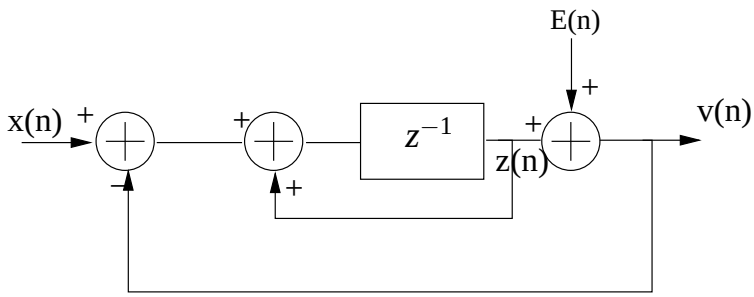


Figure 2.4: The linear model of a first order modulator

uniform white source of noise. Therefore, we have:

$$z(n) = x(n-1) - v(n-1) + z(n-1)$$

Then

$$v(n) = z(n) + E(n) \tag{2.1}$$

$$v(n) = x(n-1) - v(n-1) + z(n-1) + E(n) \tag{2.2}$$

Using 2.1 and 2.2, we obtain

$$v(n) = x(n-1) + E(n) - E(n-1)$$

The quantification noise is then equal to:

$$N(n) = E(n) - E(n-1)$$

Therefore

$$N(z) = E(z)(1 - z^{-1})$$

The expression of N(f) is:

$$N(f) = N(z)_{|e^{\frac{2j\pi f}{fe}}}$$

$$|N(f)| = 2E(f)(1 - e^{\frac{2j\pi f}{fe}})$$

$$|N(f)|^2 = 4|E(f)|^2 sin^2(\frac{\pi f}{2fe})$$

$$|N(f)|^2 = 4\frac{\Delta^2}{12fe} sin^2(\frac{\pi f}{2fe})$$

Consequently, the power noise expression is

$$P = \int_{-fm}^{fm} |N(f)|^2 df = 2\frac{\Delta^2}{12fe}\int_{-fm}^{fm}(1 - cos(\frac{\pi f}{fe}))df$$

Then

$$P = 4\frac{\Delta^2}{12fe}(f_m - \frac{fe}{\pi}sin(\frac{\pi f_m}{fe}))$$

As $f_m \ll$ fe

$$sin(\frac{\pi f_m}{fe}) \simeq \frac{\pi f_m}{fe} - \frac{1}{3}(\frac{\pi f_m}{fe})^3$$

The expression of power is then equal to:

$$P = \frac{\Delta^2}{12}\frac{\pi^2}{3}\frac{1}{OSR^3}$$

where $OSR = \frac{fe}{f_m}$

More generally, it could be proved that the in-band noise for an n loop ΣΔ converter is: [2]

$$P = \frac{\pi^{2n}}{(2n+1)OSR^{2n+1}}\frac{\Delta^2}{12}$$

Chapter 3

Description of the C code

The code source of the C simulator deals with 24 possible configurations of continuous time sigma delta converters partitioned to 4 different architectures which are : CIFB, CIFF, CRFB, CRFF. Each architecture is described in a seperate file, called a configuration file, whose name is composed of two fields which are ordrei and archi, where i $\in\{1,2,3,4,5,6\}$ and archi $\in$ {CIFB,CIFF,CRFB,CRFF}.

3.1 External files and libraries

The simulator uses the fftw library, a library developed at MIT by Matteo Frigo and Steven G. Johnson, for the computation of the discrete fourier transform of signals. It makes also use of an external file (mt19937.c) containing the source code of the Mersenne Twister random variable generator which is developed by Makoto Matsumoto and Takuji Nishimura in 1996/1997. Using this file, only uniform random variables could be generated. So, we have added a new function that returns a vector of guassian random variables. The prototype of this function is:

Vector* randn(int dim);

where dim is the size of the vector.

The function's algorithm is:

1. generating two independent uniform random variables s1 and s2
2. guassien=$cos(2\pi s1)\sqrt{(-2log(1-s2))}$ is then a guassian random variable

3.2 Some particular variables

1. structure variables: The main used variable structures are:
   (a) param: A param: is a structure including many fields for setting initial parameters.
   (b) SigmaC_Delta: This structure is defined in sigma_continu.h. It completely caracterizes the modulator as it defines the integrator gain, the feedback, feedforward and resonators coefficients.
   (c) Vector: A vector is a structure defined by its dimension and a pointer to a double type variable. This structure is defined in utils.h. It is used by almost all functions.

(d) Matrix: This structure has three fields which are the number of rows, the number of columns and a vector containing the value of each coefficient. This structure is defined in utils.h. It is used by many functions.

2. external variables

   Some variables need to be visible to many files.So we have declared them as external variables in the file sigma_continu.h.

   Examples:

   (a) S, a variable of type Vector whose components design the variables' state of the simulator, is declared as an external variable in the file sigma_continu.h so as to be visible in each of the 24 configuration files.

   (b) Amax_inti, i$\in\{1,2,3,4,5,6\}$ are variables of type double and designs the limit excursions of the output integrators. They are declared in the sigma_continu.h and set in the simul.c file.

   (c) Number is an integer that designs the position of the first elements to be selected at the next time. This variable is declared in the sigma_continu.h, set to zero by the function init() in the file sigma_continu.c, and modified by the function D2A_DWA in the file D2A_DWA.c.

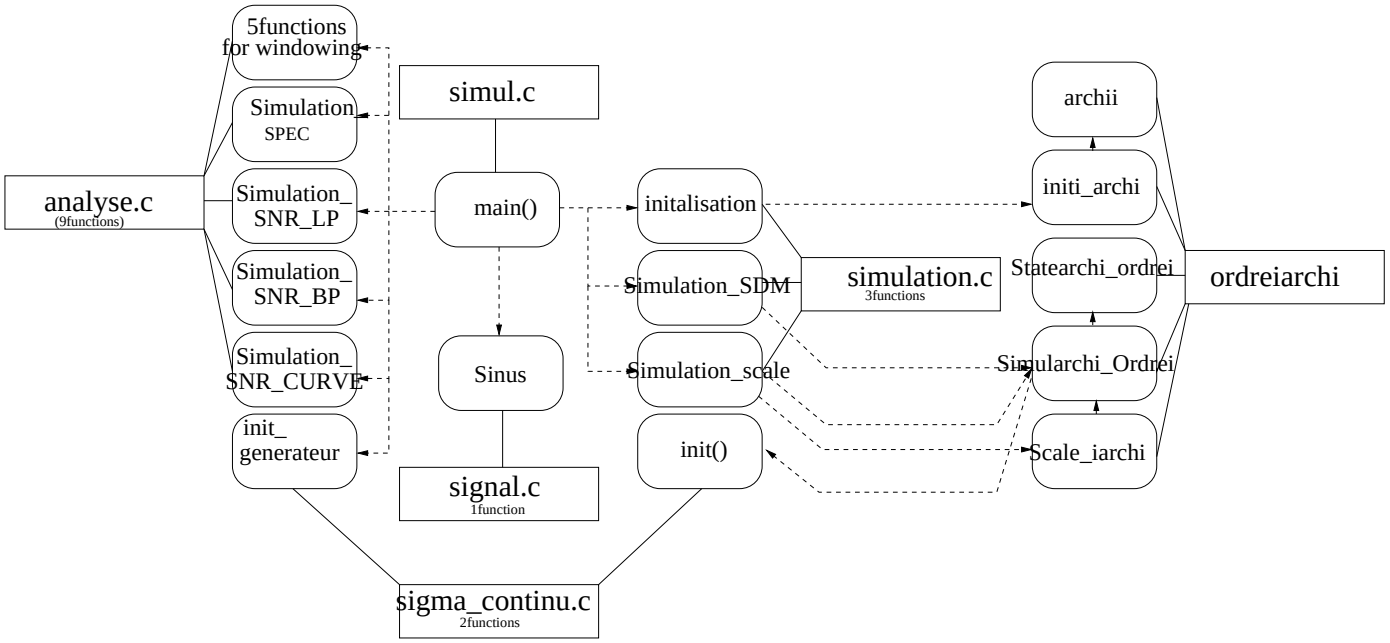
3.3 Description of the general flowchart

The general flowchart describing the links between the main source code functions is depicted in (fig 3.1)

3.3.1 simul.c file

This file contains the main function whose outlines are:

1. setting of the required parameters
2. Call for the function sinus from the file signal.c to compute the input signal.
3. Call for the function initalisation from the file simulation.c to set the modulator with the optimized coefficients
4. Call for the function init_generateur from the file sigma_continu.c in order to set the random variable generator
5. Call for the function Simulation_SDM from the file simulation.c to execute the simulation of the modulator
6. Call for the function Simulation_SPEC from the file analyse.c in order to compute the power spectrum of the output signal
7. Computing the value of the snr by either calling the Simulation_SNR_LP or Simulation_SNR_BP from the file analyse.c
8. Call for the function Simulation_scale from the file simulation.c in order to simulate the modulator having scaled coefficients.
9. Call for the function Simulation_SNR_CURVE in order to determine the snr curve for the scaled modulator



In this diagram:

i denotes an integer from 1 to 6

archi could have the following values: CIFB,CIFF,CRFB,CRFF

the box designs a file

the box with rounded corners designs a function

In this diagram we define the following relations:

F — f the file F contains the function f

f - - g the function f calls the function g

Figure 3.1: Description of the main simulator's functions

3.3.2 Description of the configuration file organisations

For each configuration, we have written a file that contains 5 functions.

1. The first function computes the optimized coefficients of the modulator. Its name is given by archii, where archi $\in$ {CIFB,CIFF,CRFB,CRFF} and i $\in$ {1,2,3,4,5,6}
2. The second function sets the modulator with the optimized coefficients. Its name is given by initi_archi
3. The third function returns for a given sigma delta converter the constants for the state presentation of the modulator. Its name is given by Statearchi_ordrei
4. The fourth function simulates the converter and returns its output. Its name is given by Simularchi_Ordrei
5. The fifth function determines the scaling coefficients and simulates the scaled modulator

The flowchart describing a configuration file is depicted in (fig 3.2)

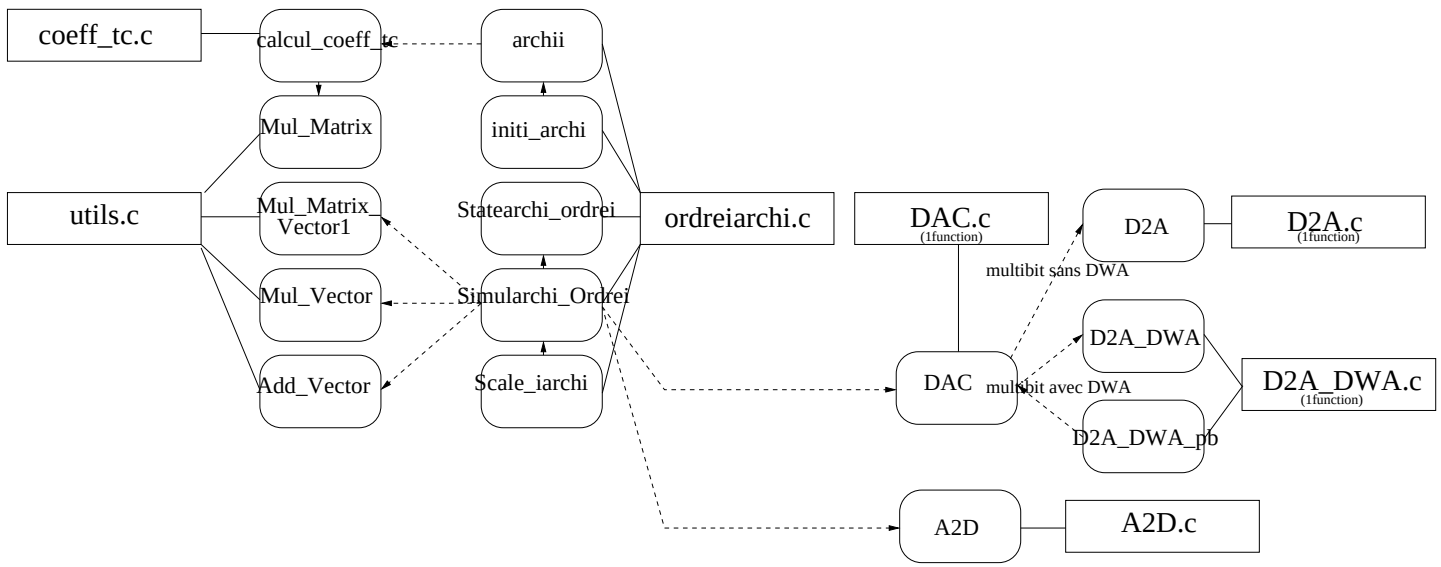


Figure 3.2:

In this diagram:

i denotes an integer from 1 to 6

archi could have the following values: CIFB,CIFF,CRFB,CRFF

the box designs a file

the box with rounded corners designs a function

In this diagram we define the following relations:

F — f the file F contains the function f

f - - g the function f calls the function g

Chapter 4

Simulation of $\Sigma\,\Delta$ converters

4.1 Determination of optimized coefficients for continuous Sigma delta converter

It has proved that the optimized coefficients for a given continuous Sigma delta converter could be determined by applying a transformation to the discrete time $\Sigma\,\Delta$ converter coefficients.[5]

1. Determination of CT resonator feedback coefficients:

   CT resonator feedback could be determined using the next formula:

   $$g_{ci} = \cos^{-1}(1 - \frac{g_{di}^2}{2});$$

   where g_{ci} is the CT feedback resonator and g_{di} is the DT feedback resonator.

2. Determination of CT feedback and feedforward coefficients:

   CT feedback and feedforward coefficients could be determined using the next formula:

   $$A = C^{-1}DB$$

   where

   (a) A is the CT Vector coefficients.

   (b) B is the DT Vector coefficients

   (c) C is a constant matrix that depends on the DAC transfer function.

   (d) D is a constant matrix.

The computation of these coefficients are achieved by the function archii from the file ordreiarchii.

4.2 State representation of the $\Sigma\Delta$ modulator

4.2.1 integrator model

To compute the output of a continuous time integrator, we need to know the value of the input over time. But in the case of sigma delta converters, the input of integrators

depends on the previous outputs. In order to describe the system continuously in time, we have chosen to modelize the integrator by using a discrete time model with an infinitesimal time step.

$$H(z) = \frac{Aintz^{-}1}{1-z^{-}1} \tag{4.1}$$

where Aint is the gain of the discrete time integrator.
Verification of the model:

$$H(z) = \frac{Aintz^{-}1}{1-z^{-}1} = \frac{Aintz^{\frac{-1}{2}}}{z^{\frac{1}{2}} - z^{\frac{-1}{2}}}$$

Then

$$H(f) = H(z)_{|e^{2j\pi fT}} = \frac{Ainte^{-j\pi fT}}{jsin(\pi fT)}$$

where T is the step size and f is the input signal frequency.
when fT $\rightarrow$ 0, the expression of H(f) become:

$$H(f) = \frac{e^{\frac{-j\pi fT}{2}}}{j\pi fT}$$

It turns out that for a small infinetisimal step, the response of the disrete time integrator is similar to that of a continuous time integrator whose gain is equal to A'int=$\frac{Aint}{T}$ For instance, a continuous time integrator with a gain equal to 1 is modelized by a discrete time one with a gain equal to T.

4.2.2 State representation of a $\Sigma\,\Delta$ converter

Lets Y_i and X_i denotes respectively the output and the input of the ith discrete time integrator with a gain of α_i

$$\frac{Y_i(z)}{X_i(z)} = \alpha_i \frac{z^{-}1}{1-z^{-}1}$$

So, for each integrator, the dynamical equation is:
$y_i(n) = y_i(n-1) + \alpha_i * x_i(n-1)$
As $x_i(n-1)$ is a linear function of the different outputs, a matrix representation which makes use of the variable states of the system could be elaborated. Let's take an example of a second order CRFB converter. Lets S_n denotes the output of the integra-

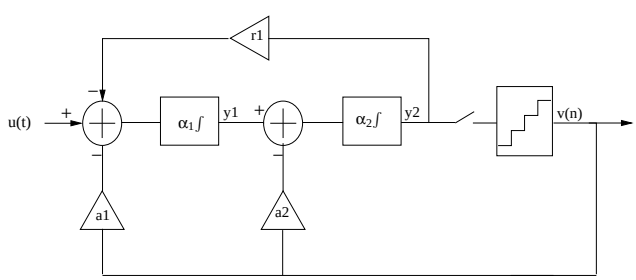


Figure 4.1: a second order CRFB converter

tors at the $n^t h$ step of time.

$$S_n = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}_n$$

The state representation of the system is then:

$$S_{n+1} = \begin{pmatrix} 1 & -r_1\alpha_1 \\ \alpha_2 & 1 \end{pmatrix} S_n + u_n \begin{pmatrix} 1 \\ 0 \end{pmatrix} - v_n \begin{pmatrix} \alpha_1 a_1 \\ \alpha_2 a_2 \end{pmatrix}$$

where v_n is the nth output of the converter. More generally, the dynamical equations of a $\Sigma\,\Delta$ converter could be written as

$$S(n+1) = AS(n) + Bu(n) - Cv(n)$$

where

- S(n+1) denotes the integrator output at the time (n+1) and y(n) denotes thes output of the modulator.
- A is a constant vector given by the next formula:

$$A = \begin{pmatrix} 1 & -r_{1,2}g_1 & 0 & \cdots & & \cdots & 0 \\ g_2 & 1 & -r_{2,3}g_2 & \ddots & & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & g_i & 1 & -r_{i,i+1}g_i & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & & & \vdots \\ 0 & \cdots & \cdots & & 0 & g_n & 1 \end{pmatrix}$$

  where $r_{i,i+1}$ is different from zero iff there exists a resonator coefficient between the ith and the $(i+1)^{th}$ integrators, and g_i is the gain of the ith integrator

- B is a constant vector equal to:

$$B = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- C is a constant vector given by the next expression:

$$C = \begin{bmatrix} c_1 g_1 \\ c_2 g_2 \\ \vdots \\ c_n g_n \end{bmatrix}$$

  where c_i and g_i are respectively the feedback coefficients and the integrators gains.

The determination of the constants of the representation state of a given $\Sigma\,\Delta$ converter is achieved by the function Statearchi_ordrei.

4.3 Simulation of monobit converters

4.3.1 Simulation

Simulation is achieved by the functions whose names are given by Simularchi_Ordrei. It consists in computing the output of the converter at each step of time by using the dynamical equations and the model of the DAC and the quantizer. So for each step of time, Simulation consists in:

- computing the state variables
- computing the output of the DAC
- computing the output of the ADC

Several kinds of DACs have been considered. The computation of their outputs is achieved by the function DAC in the file DAC.c. This function has three parameters which are:

- i: is the number of time steps since the beginning of the simulation
- k: the number of periods since the beginning of the simulation
- v: is a Vector that contains the output values since the beginning of the simulation.

1. RZ DAC for monobit converters

   The delay of the quantizer response has a negative effect on the performance of the $\Sigma\,\Delta$ converter and may sometimes entail its instability. To overcome this problem, the NRZ DAC should be substituted by a RZ one. As it is shown in (fig 4.2), the RZ feedback signal is applied after a delay time, td and lasts for a period equal to τ.

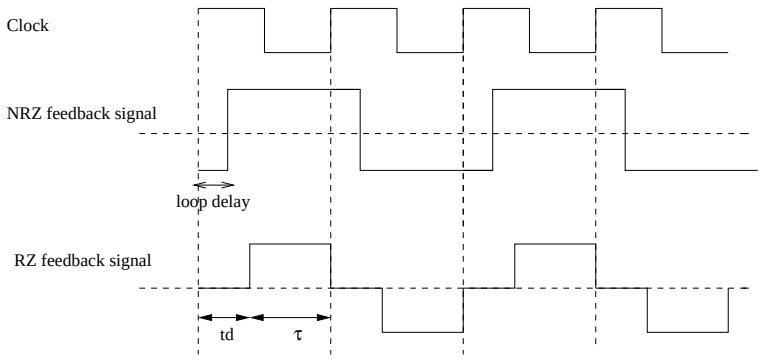


Figure 4.2: comparision between RZ and NRZ signal feedback

As the method of the determination of the CT coefficients takes into account the shape of the DAC transfer function, using a RZ signal feedback didn't degrade the SNR level. To compute the output of the DAC, we need to determine delay times td and τ in terms of number of time steps. We have declared then two

variables index_tdk and index_tauk which denote respectively the number of time steps for td and τ. These variables are given by the next expression:

$$index\_tdk = E(\frac{td}{pas}); index\_tauk = E(\frac{\tau}{pas});$$

where pas denotes the time step of the simulation. For RZ DACs, the response of the DAC is different from zero iff

$$mod(i, \frac{1}{pas}) <= index\_tdk + index\_tauk$$

and

$$mod(i, \frac{1}{pas}) >= index\_tdk$$

The (fig4.3) shows that the return to zero DAC didn't degrade the performance of the CT modulator.

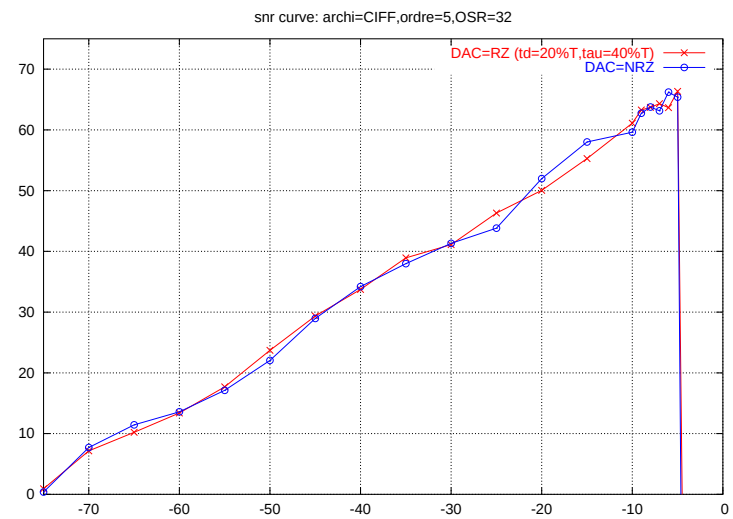


Figure 4.3: snr curves of a fifth order CIFF architectured modulator

2. Sinsodal DAC for monobit converters

   The response of the sinusodal DAC is:

   $$y(t) = 1 - cos(wt)$$

   if the input is positive

   $$y(t) = -1 + cos(wt)$$

   otherwise.

   We have simulate converters having a sinusodal DAC for a second and a third order $\Sigma\,\Delta$ architectured converter. We have obtained the following results:

   - Second order CIFF architectured modulator (fig4.4)

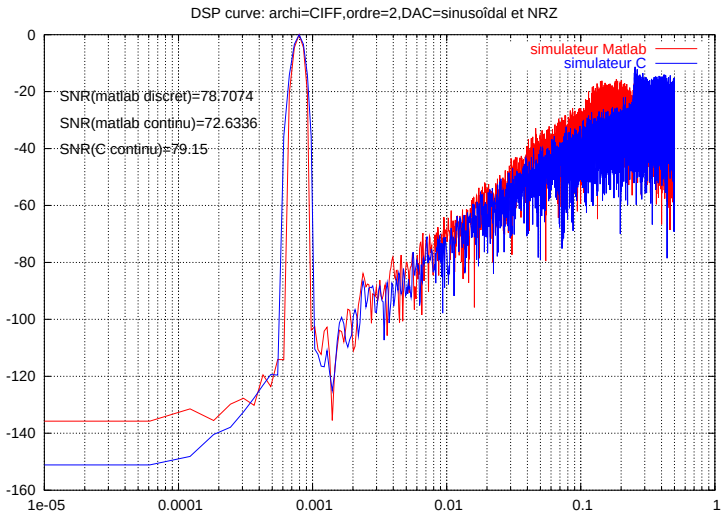


Figure 4.4: DSP curve of a second order CIFF architectured modulator

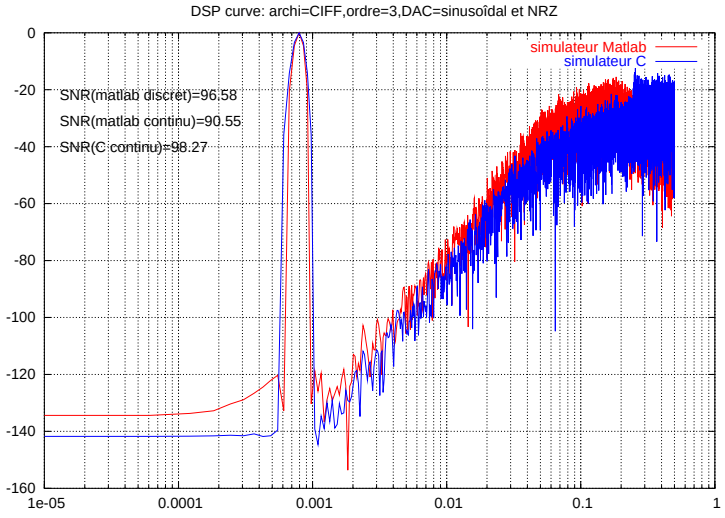


Figure 4.5: DSP curve of a third order CIFF architectured modulator

- Third order CIFF architectured modulator (fig4.5)

The advantage of the sinusodal DAC is that it can attenuate the negative effect of the jitter noising as its response tends to zero at each period. (fig4.6)

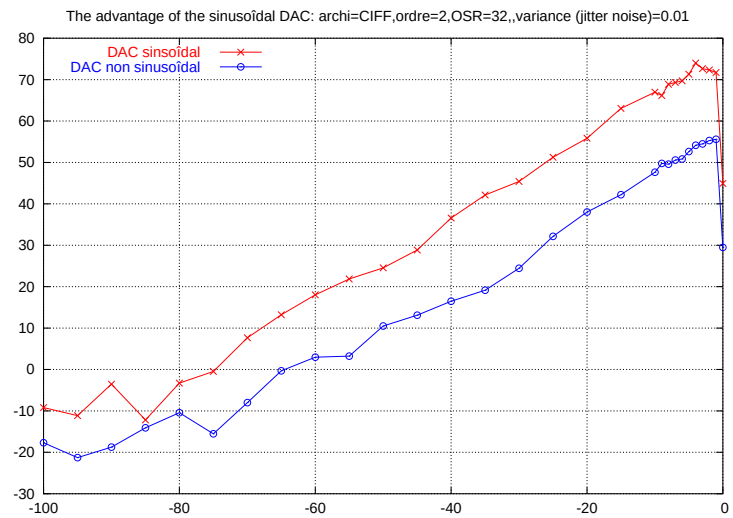


Figure 4.6: the advantage of the sinsodal DAC

4.3.2 Power spectrum density estimation

The power spectrum density estimate of a discrete signal x is given by the next expression

$$S_{xx}(f_k) = \frac{|X_N(f_k)|^2}{N}; k \in 0,..,N-1 and\, f_k = \frac{k f_s}{N}$$

where N is the number of data points available, $|X_N(f)|^2$ is the discrete fourier transform of x. In order to avoid spectral leakage, the C simulator make use of 4 types of windows whose characteristics are depicted in tab4.1

Window	-3db Main Lobe Width(bins)	-6db Main Lobe Width(bins)	Maximum Side Lobe level(db)	Side Lobe Roll-off rate (db/decade)
Hanning	1.44	2	-32	60
Hamming	1.3	1.81	-43	20
Blackman	1.68	2.35	-58	18
Blackman Harris	1.9	2.72	-92	6

Table 4.1: Characteristics of the used window functions

The (fig 4.7) shows the frequency spectrum characteristics of a window in more detail.

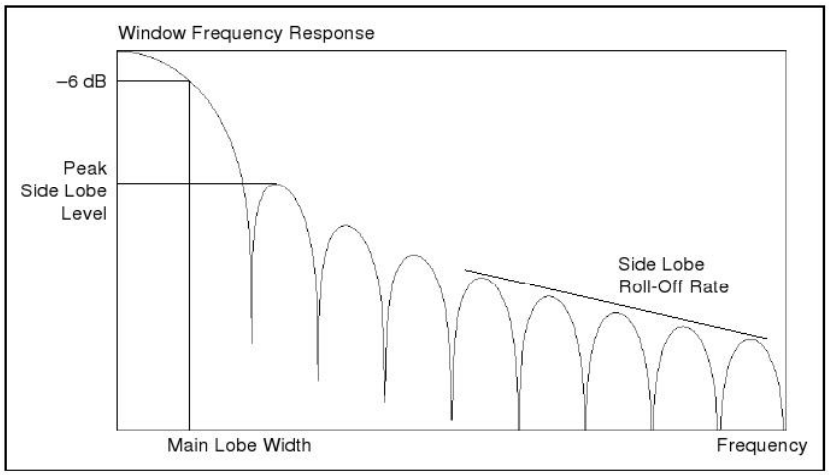


Figure 4.7: Frequency response of a window

4.3.3 SNR estimation

The SNR is defined by the next formula:

$$SNR(db) = 10log(\frac{P\_signal}{P\_bruit})$$

Both the signal and noise power should be computed by considering the frequency bins that lies in the band of interest.

The width of the band of interest is equal to $\frac{fe}{2OSR}$, where OSR is the oversampling ratio. The signal power lies in an interval centered on the input frequency. The width of the interval depend on the main lobe window wide.

Table4.2 shows the number of frequency bins that should be used for the computation of the signal power.

window	number of frequency bins
Blackman window	11
Blackman harris window	11
Hann window	5
hamming window	5

Table 4.2: number of frequency bins used for each window

4.3.3.1 low pass converters

For low pass converters, the signal frequency input is a fraction of the sampling frequency. We have chosen for example, an input signal frequency equal to

$$f_{in} = \frac{13fs}{N}$$

where N is the number of data points available. We have defined then next variables which are:

- Ns: the number of the frequency bins that corresponds to the input signal frequency
- Nbw: is the number of frequency bins in the band of interest.
- nbrepoints_snr: is the number of frequency bins to be considered while computing the signal power.

The power signal Ps and the power noise Pb signal are estimated by using the following expressions:

$$Ps = \sum_{k=Ns-\frac{nbrepoints_snr}{2}}^{k=Ns+\frac{nbrepoints_snr}{2}} S_{xx}(f_k)$$

$$Pb = \sum_{k=0}^{k=Ns-\frac{nbrepoints_snr}{2}} S_{xx}(f_k) + \sum_{k=Ns+\frac{nbrepoints_snr}{2}}^{Nbw} S_{xx}(f_k)$$

4.3.3.2 pass band converters

For pass band converters, the input signal is equal to a fraction of the sampling frequency added by the value of the central frequency. We have chosen for example an input signal frequency equal to:

$$f_{in} = fo + \frac{13fs}{N}$$

The estimation of the signal power and the noise power are computed by using the same strategy as above with the single difference that the band of interest lies now between

$No - \frac{Nbw}{2}$ and $No + \frac{Nbw}{2}$, where No is the number of the frequency bin that correspond to the central frequency. Consequently, the expression of the signal power remains the same as above, while that of the noise power is now given by the following expression:

$$Pb = \sum_{k=No-\frac{Nbw}{2}}^{k=Ns-\frac{nbrepointsssnr}{2}} S_{xx}(f_k) + \sum_{k=Ns+\frac{nbrepointsssnr}{2}}^{No+\frac{Nbw}{2}} S_{xx}(f_k)$$

4.3.3.3 Scaling

The objective of scaling is to limit the integrators output swing. A systematic approach has been presented in [6], where it has been proved that this scaling method preserves the NTF and the STF functions and therefore the SNR level. This approach consists in:

- Simulate the ΣΔ converter with the original coefficients.
- For each integrator, we compute the scaling factor fi, where

$$fi = \frac{max(outputintegrator)}{desiredi^t hintegrator}$$

- distribute f_i in order to preserve the same NTF and STF functions.(fig4.8)

(fig4.9) represents the output integrators for both original and scaled coefficients.

4.3.3.4 Jitter noising

1. Modelization of the jitter noising

   The jitter noising is caused by the delay made by the clock when transiting from upper value to lower value or vice versa. As this delay is very small, we have considered to modelize its effect rather than modelizing it directly. In fact, An ideal DAC has its response x verify the following condition:

$$\frac{1}{T}\int_{td}^{td+\tau} x(t) = \frac{1}{T}\int_{td}^{td+\tau} \frac{T}{\tau} = 1$$

   But with jitter noising, we have

$$\frac{1}{T}\int_{td}^{td+\tau+\delta\tau} \frac{T}{\tau} = 1 + \frac{\delta\tau}{\tau} \tag{4.2}$$

   where $\delta\tau$ denotes a gaussian distributed noise. Then the jitter noising could be modelized by a DAC whose reponse's amplitude is equal to $\frac{T}{\tau} + \delta_A$, where δ_A is a gaussian distributed noise. We have then

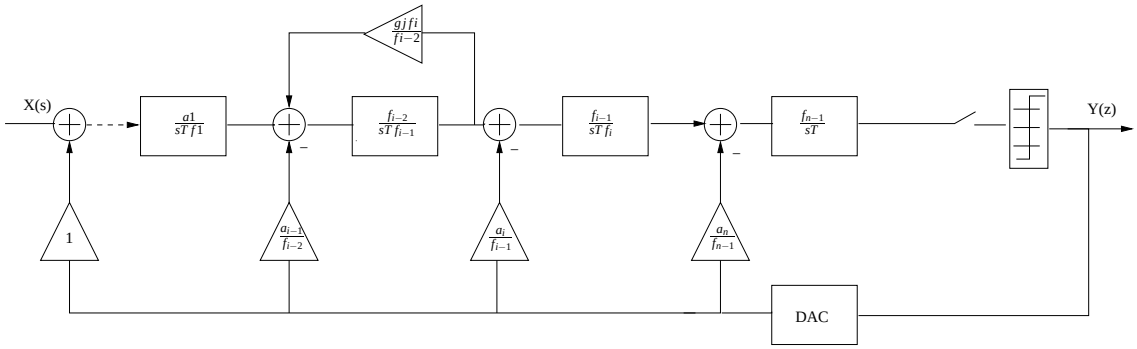
$$\frac{1}{T}\int_{td}^{td+\tau} (\frac{T}{\tau} + \delta_A)dt = 1 + \frac{\delta_A \tau}{T}$$

   As the DAC's response must verify (4.2)

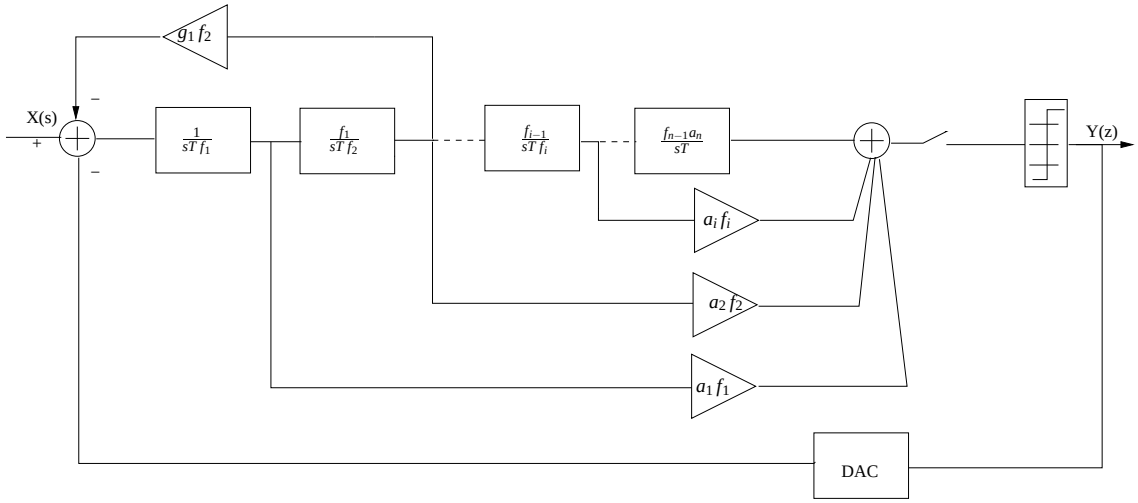
$$\delta_A = \frac{\delta\tau T}{\tau^2}$$

2. evaluation of simulation results

   The snr curve vs the power of the jitter noising is depicted in (fig 4.10.) We can notice that the snr level goes down when the power of the jitter noising increases.

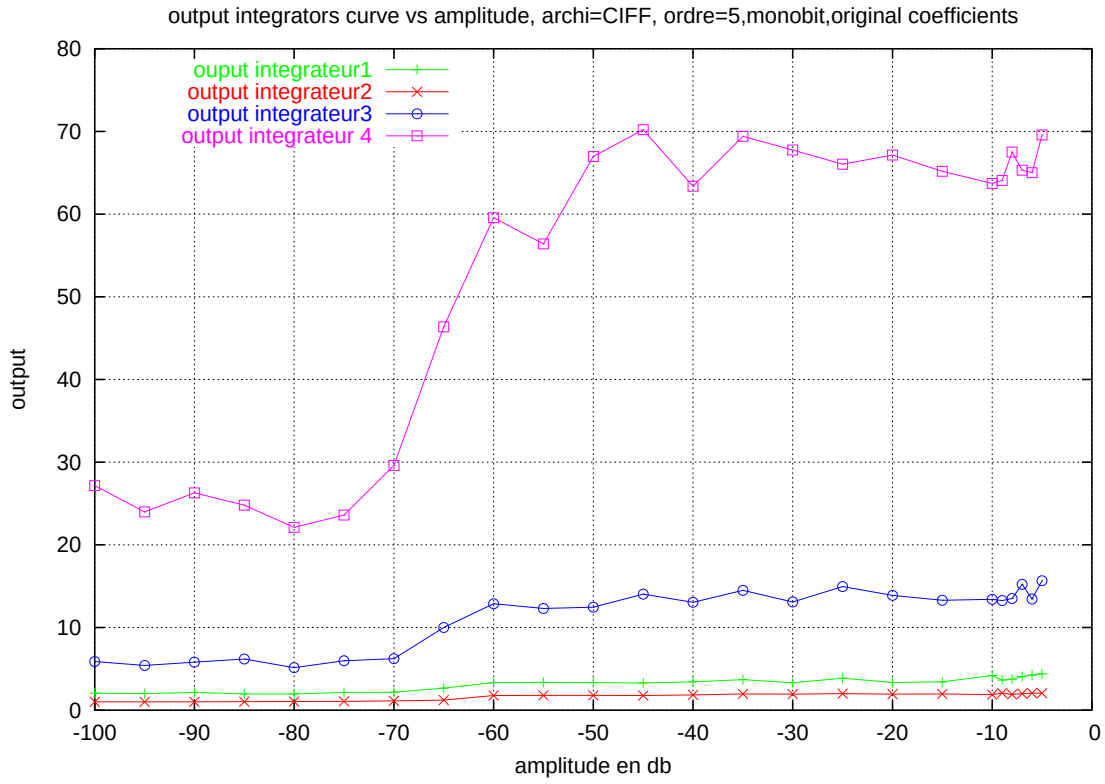


(a) Feedback architectures

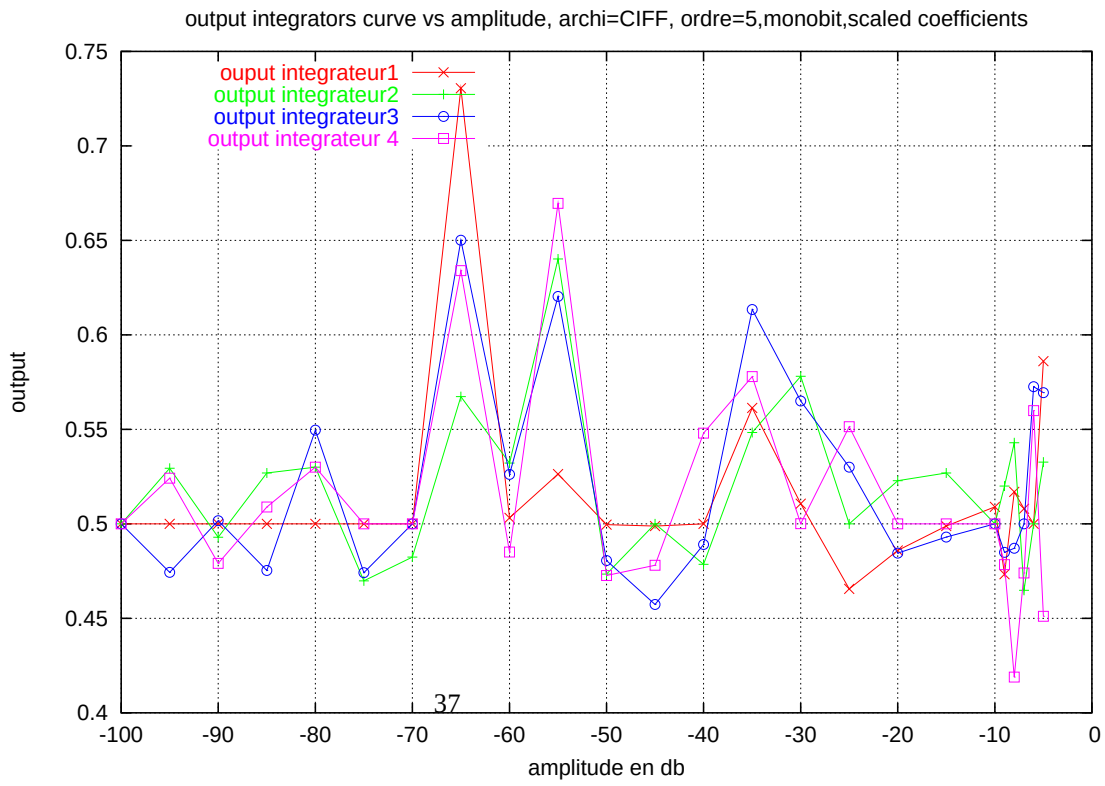


(b) Feedforward architectures

Figure 4.8: Scaling method for feedback(a) and feedforward(b) Σ Δ architectures



(a) Original coefficients



(b) Scaled architectures

Figure 4.9: Integrators swings decrease

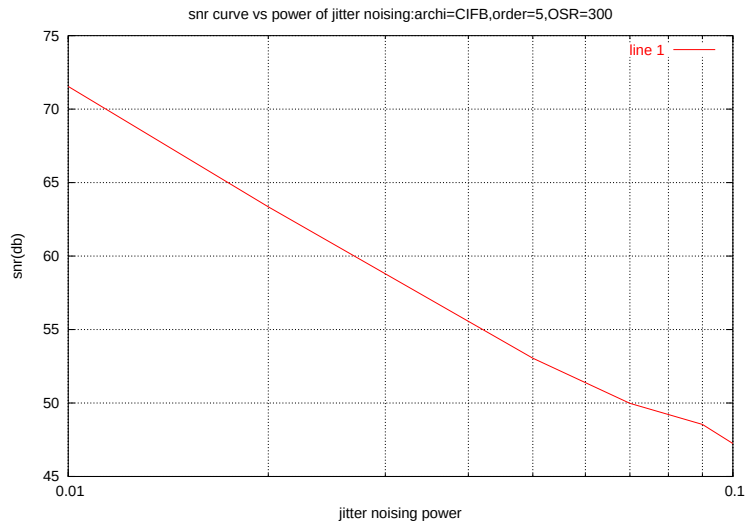


Figure 4.10: SNR curve vs power of jitter noising

4.4 multi bit converters

Multi bit converters have two main advantages over the signle bit one.

1. A higher performance: In fact, having the same OSR and the same order as a monobit converter, the multibit converter could achieve a higher signal to noise ration for two reasons:

   - According the linear analysis of a $\Sigma\ \Delta$ converter, the adding of a single bit to the quantizer is equivalent to an SNR improvement of 6 dB.
   - Besides, as we have seen before, the NTF constraint is less limiting for multibit converters.

2. A better resolution: In fact, the output of the converter is closer to the desired output. However, the main inconvenients of multibit converters is that the DAC cannot be made perfectly linear since it is impossible to manufacture DAC's element sources with exactly the same step size.

The problem is made worse by the fact that any error introduced into the feedback signal will translate into errors for the overall converter. To illustrate this, lets'consider the linear multibit $\Sigma\ \Delta$ converter: (fig4.11) The output of the converter is therefore equal to

$$V(z) = G(z)U(z) + H(z)E(z) + (H(z) - 1)E_{DAC}(z)$$

In the band of interest, H(z) is attenuated, so H(z)-1=$\simeq$ -$E_{DAC}(z)$. The in-band noise will be then dominated by the DAC noise. To correct the DAC nonlinearity, several algortihms have been used. In the following, we will consider the most common one which is Data Weighted averaging.

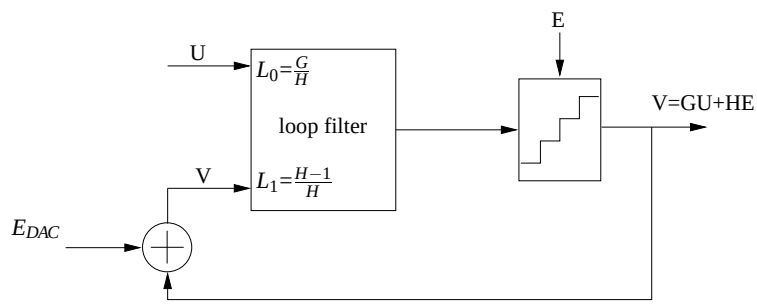


Figure 4.11: a general diagram of multibit Σ Δ converter

4.4.1 the multibit quantizer

The response of a 8 level quantizer is depicted in (fig4.12) where

$$\Delta = \frac{2}{2^{NN}-1}$$

We have represented with dashed points the translation of the response curve by 0.5Δ along the y axis. Let's y and x denote respectively the output and the input of the quantizer,and f(x) denotes the analytical expression of the dashed curve. Then

$$f(x) = (E(\frac{x}{\Delta})+1)\Delta = round(\frac{x}{\Delta})+0.5)\Delta$$

where round(x) returns the nearest integer to x. The expression of y vs x is then

$$y(x) = (round((\frac{x}{\Delta})+0.5)-0.5)\Delta$$

Each level is then represented by a single code which is equal to:

$$code = round(\frac{y(x)}{\Delta}+1)$$

The determination of the output of the multibit quantizer is achieved by the A2D function in the file A2D.c

4.4.2 DAC architecture

The DAC is composed of current sources whose number is equal to the number of the quantizer levels.(fig4.13) For each time, the number of elements selectioned is equal to code, where code is the output of the ADC. A gaussian distributed error with zero mean and standard deviation equal to σ is added to each current source.That's mean that each current source provides a DC signal equal to 1 added by a certain value of error which is constant over time. The idea of the DWA algorithm is to introduce a transfer function in the feedforward path,denoted the MTF(mismatch transfer function), that eliminates the noise caused by the DAC in the band of interest.

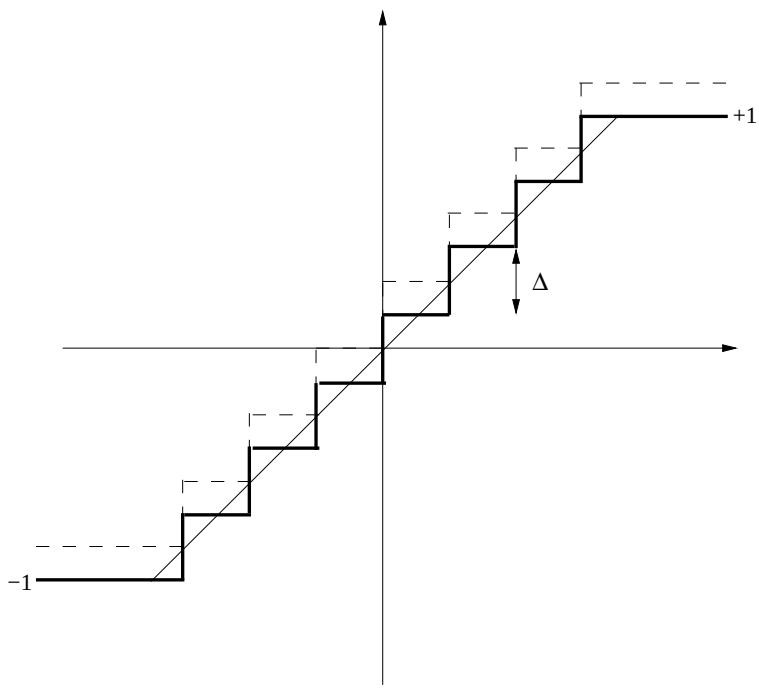


Figure 4.12: A 8 level quantizer response

4.4.3 Data Weighted Averaging algorithm for low pass converters

1. Theoretical analysis

   For low pass converters, the MTF of the DWA algorithm is equal to

   $$H(z) = 1 - z^{-1}$$

   The thoretical model of this algorithm is depicted in 4.14, where ptr(n) is the position of the unit element selected at time n. [7][8] lets V(z) and dV(z) denote respectively the input and the output of the DAC. Then,

   $$dV(z) = V(z) + (1 - z^{-1})E_{DAC}$$

   $$|H(f)| = H(z)_{|e^{2j\pi fT}} = 2|sin(\pi fT)|$$

   for low frequencies, fT→ 0, —H(f)—→ 0, and then the error of the DAC is attenuated.

2. implementation

   According to the theoretical analysis:

   $$V(n) = ptr(n) - ptr(n-1)$$

   therefore

   $$ptr(n) = V(n) + ptr(n-1)$$

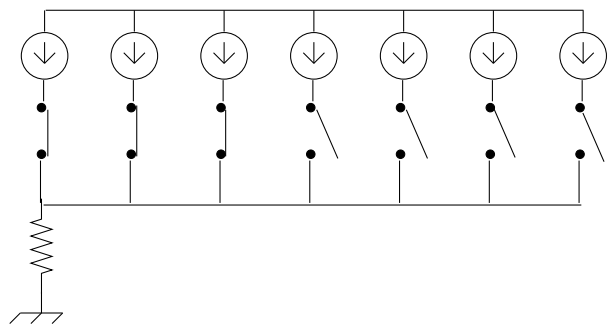


Figure 4.13: A 8 level DAC architecture

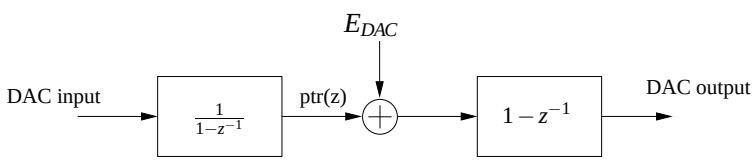


Figure 4.14: A theoretical model of DWA algorithm

The DWA algorithm consists then in selecting adjacent elements from an array beginning with the next unused element. Consequently, all the DAC elements are used at the maximul possible rate. The element selection pattern is depicted in (fig 4.15). The DWA algorithm make use of an external variable, number, which points on the next selected element, and is updated at each step of time. For low pass converters,the DWA algorithm is achieved by the D2A_DWA function in the D2A_DWA.c file.

3. Evaluation of the results

   - DSP curve The fig4.16 shows simulation results for a fifth order low pass modulator. With 1% of mismatch, the SNR drops to 41,34 db if no mismatch shaping algorithm is used, but rises to 99.7 db if the element selection strategy described above is employed.
   - SNR curves The fig4.17 shows that, using the mismatch shaping algorithm, we could obtain the same level of SNR as that of the ideal DAC.

4.4.4 DWA algorithm for pass band converters

1. Theoretical analysis

   For pass band converters, the band of interest has a width equal to $\frac{fs}{2OSR}$ and lies on an interval centered on the central frequency denoted by f_o which is equal to $\frac{fs}{4}$(where f_s is the sampling frequency). In reference to [7], the MTF that cut off the signal in the band of interest is equal to:

$$H(z) = 1 + z^{-2}$$

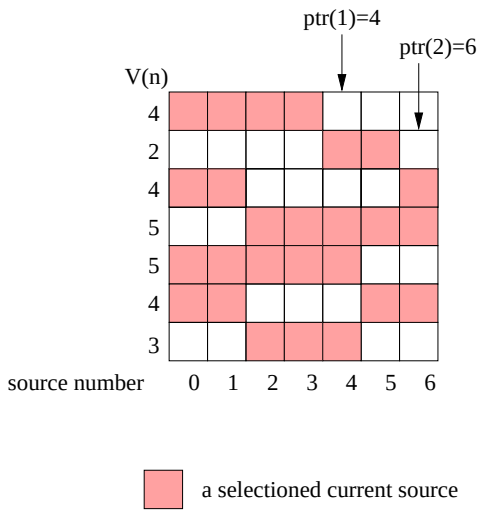


Figure 4.15: The element selection pattern of an 8 level DAC

$$|H(f)| = H(z)_{|e^{2j\pi fT}} = 2|sin(2\pi fT)| = 2|sin(2\pi\frac{f}{f_s})|$$

We can verify easily that by introducing in the feedforward path H(f) the signal is attenuated in the band of interest.

2. Implementation

   The element selection pattern is depicted in (fig4.18)

   where sens denotes the direction of the rotation into the selected elements. For pass band converters, the DWA algorithm makes use of 4 external variables which are number, number 1, number 2 and sens.

   - number: number denotes the position of the current selected element at time n.
   - number1: number 1 denotes the position of the selected element at time n-2.
   - number2: number 2 denotes the position of the selected element at time n-1. The values of number 1 and number 2 are updated each period of time. (not as the previous algorithm where the value of number is updated each step of time).
   - sens:denotes the direction of the rotation into selected elments. The value of sens changed each two period of time.

   The DWA algorithm for pass band converters is achieved by the D2A_DWApb function in the file D2A_DWA.c

3. Evaluation of the results

   - DSP curves

     The (fig4.19) shows simulation results for a fourth pass band CRFB architectured modulator. With 20% of mismatch, the snr drops to 36 db if

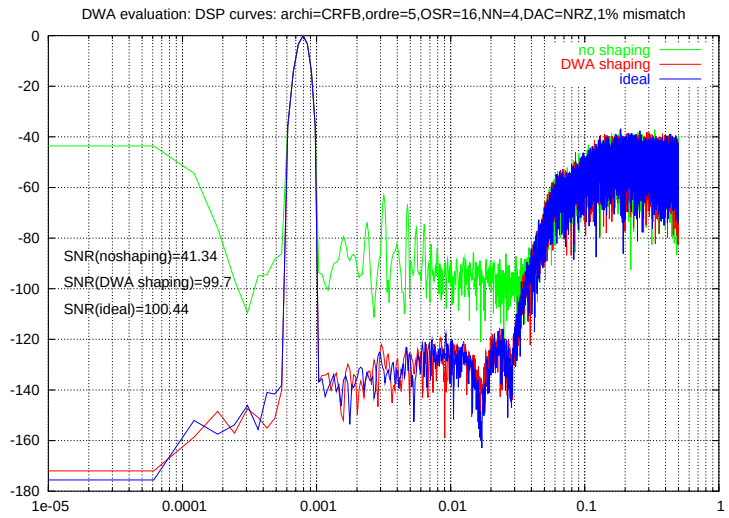


Figure 4.16: Evaluation of the DWA algorithm

no misamtch shaping algorithm is used, but rises to 51db if the proposed algorithm is employed.

- SNR curves

  The (fig4.20) shows that the DWA algorithm is more efficient for higher amplitudes.

4.5 Validation of the results

In order to validate the results, we compare the SNR Curve vs the amplitude and the power spectrum curve provided by the Matlab simualtor with that of the C simualtor. We have represented in the annexe chapter all the DSP and SNR curves for both the C and the Matlab simulators for all possible configurations. These curves prove the validity of the results obtained by the C simulator. Besides, simulations have been done on a PC-2Ghz (Ram=512 Octets) and have showed that the time execution needed for simulation using the Matlab simulator is much higher than that of the C simulator. (See Annexe tables)

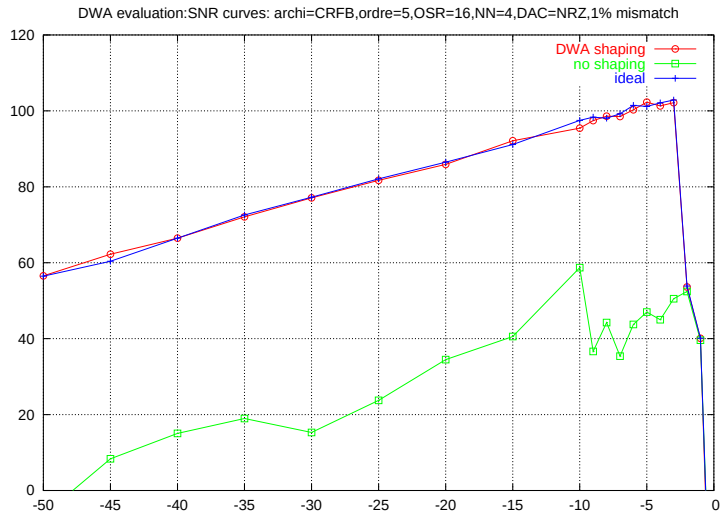


Figure 4.17: Evaluation of the DWA algorithm

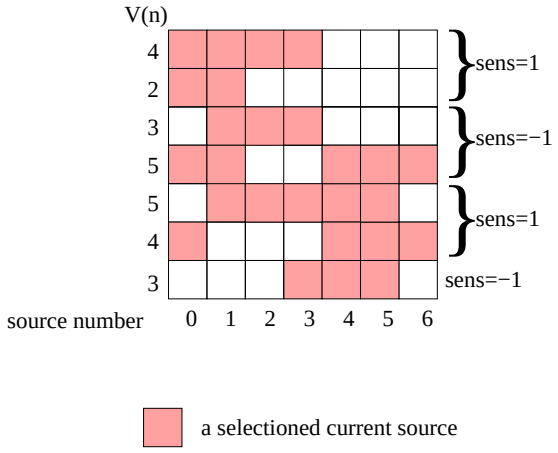


Figure 4.18: The element selection pattern for pass band converters

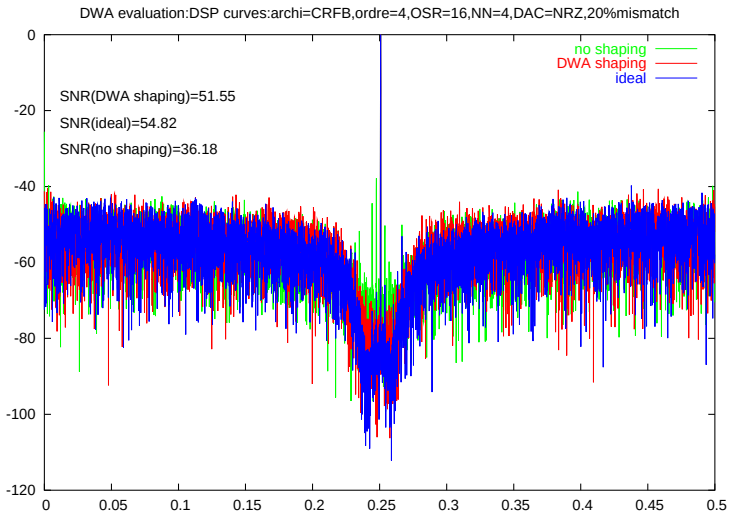


Figure 4.19: Evaluation of the DWA algorithm for passband converters

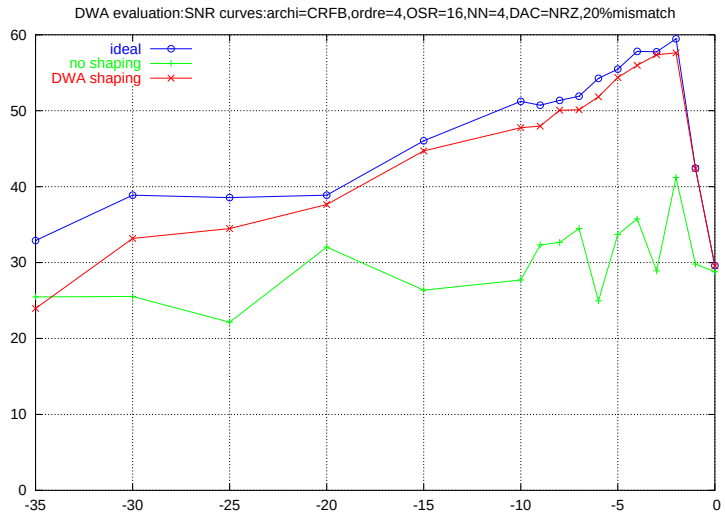


Figure 4.20: Evaluation of the DWA algorithm for pass band converters

Chapter 5

Conclusion

In my engineering internship, I had to develop a C library for simulating continous time $\Sigma\,\Delta$ converters.

In this report, I have presented first the theoretical studies that had been carried out about $\Sigma\,\Delta$ converters. Then, I have presented the results' simulations in order to prove the validity of the C simulator.

However, this library could be improved by

- considering complex architectures composed of multi stage converters.
- Adding an algorithm that computes the invariant sets of the converter,thus verifying the system stability
- Considering the error induced by the delay time error before turning on or off the DAC's element sources and implementing the algorithm that resolves this problem.

Chapter 6

annexes

6.1 DSP Curves

1. CIFB architecture

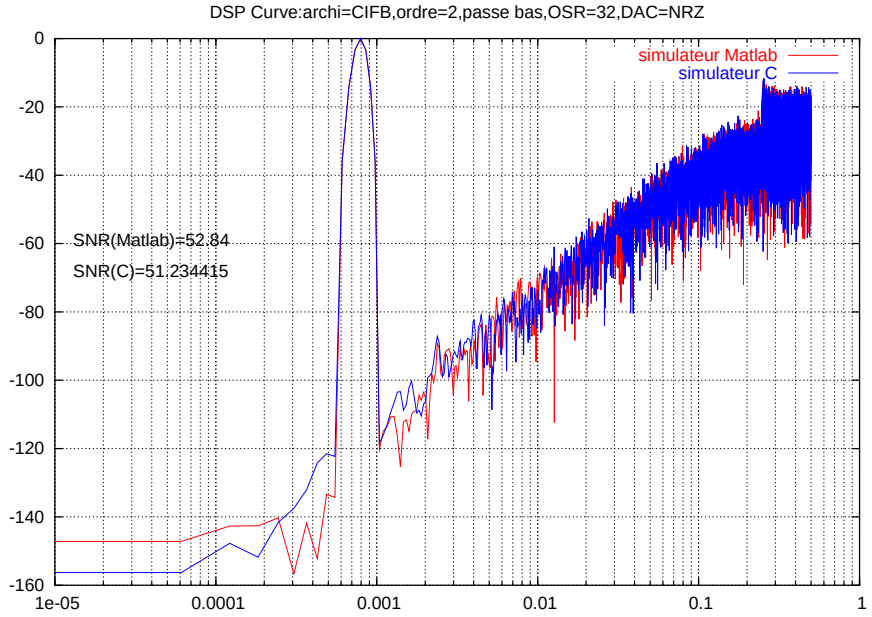


Figure 6.1: DSP_curve of a second order Σ Δ CIFB architectured modulator

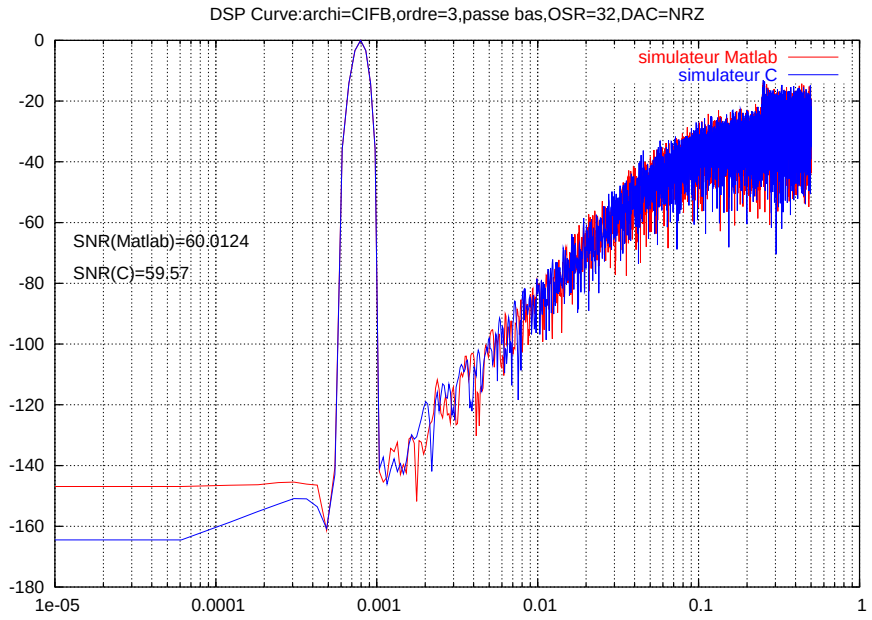


Figure 6.2: DSP_curve of a third order Σ Δ CIFB architectured modulator

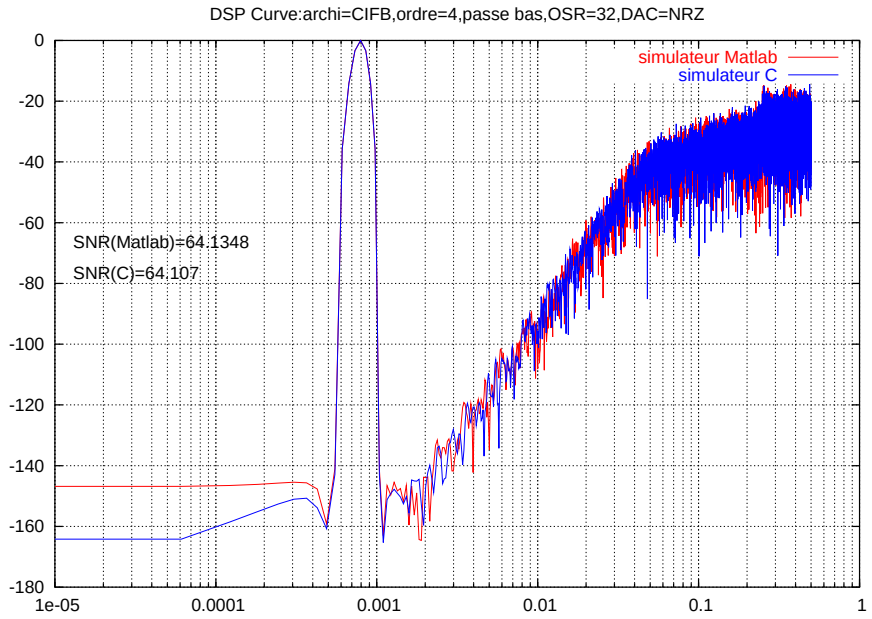


Figure 6.3: DSP_curve of a fourth order Σ Δ CIFB architectured modulator

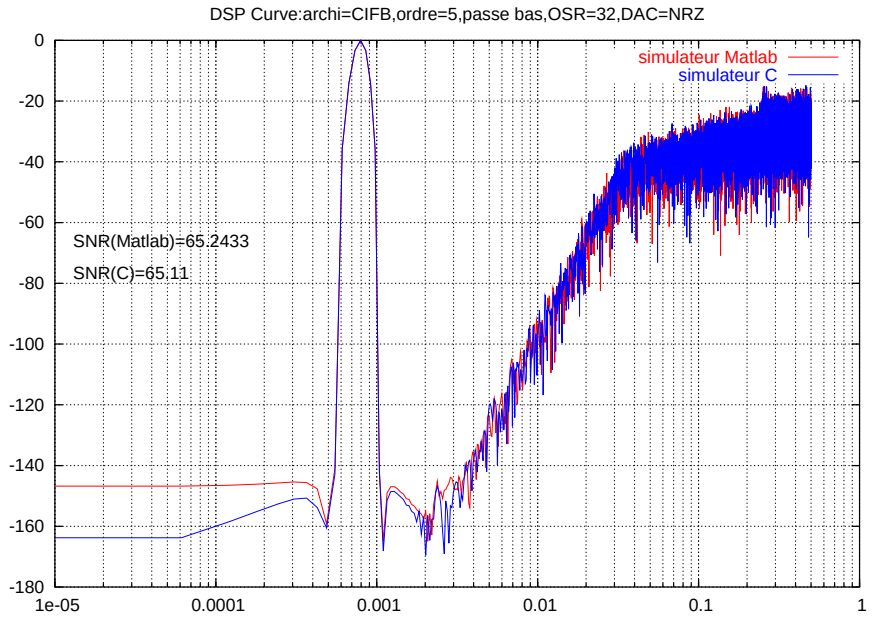


Figure 6.4: DSP_curve of a fifth order $\Sigma\Delta$ CIFB architectured modulator

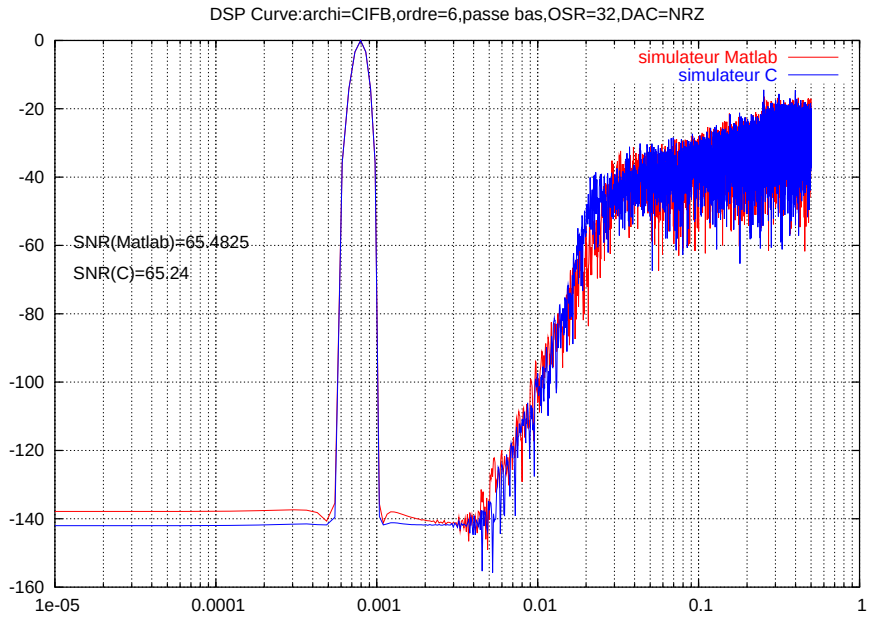


Figure 6.5: DSP_curve of a sixth order $\Sigma\Delta$ CIFB architectured modulator

2. CRFB architecture

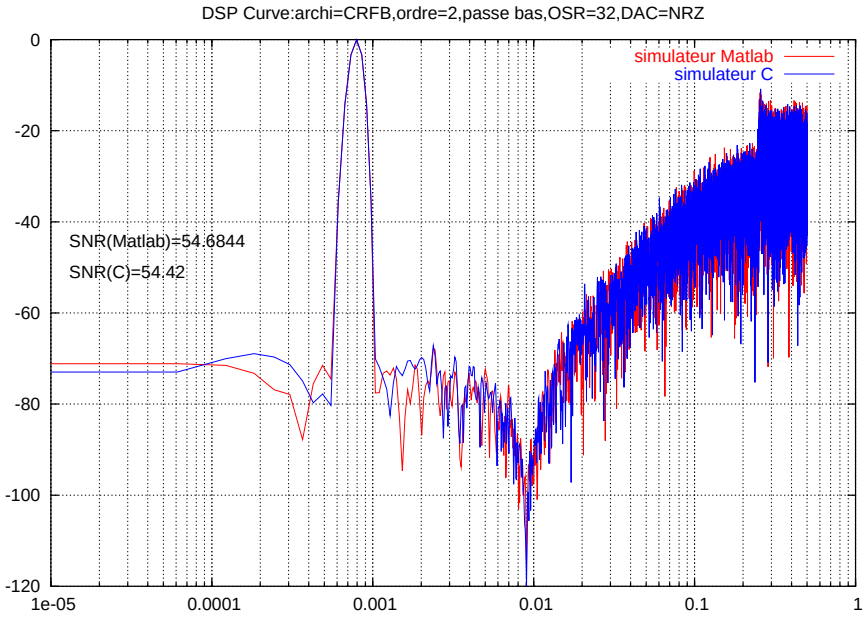


Figure 6.6: DSP_curve of a second order Σ Δ CRFB architectured modulator

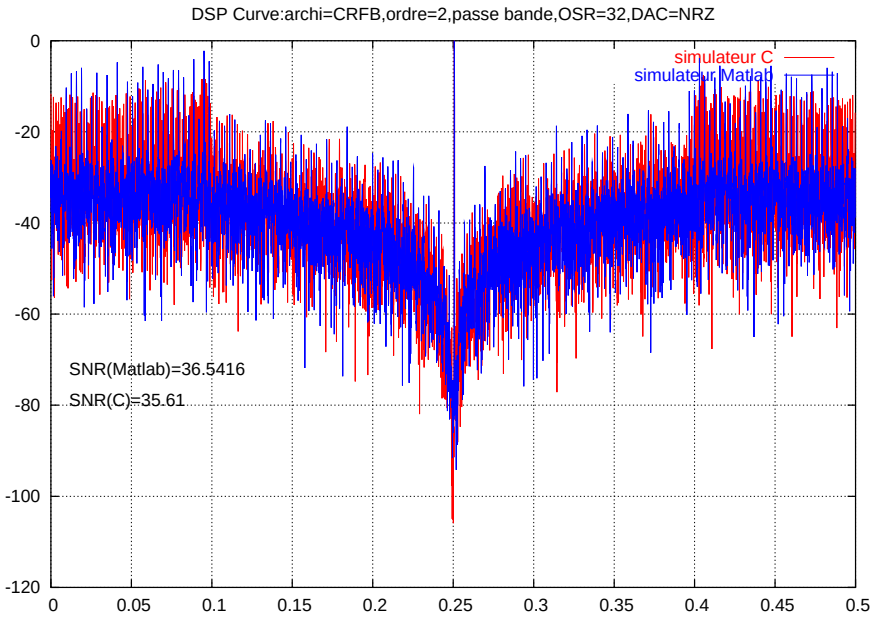


Figure 6.7: DSP_curve of a second order pass band Σ Δ CRFB architectured modulator

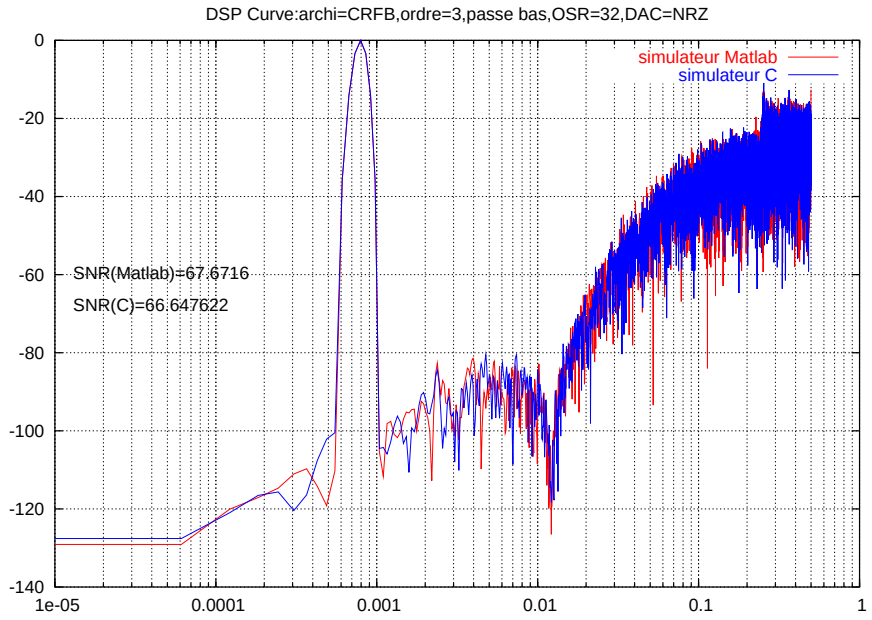


Figure 6.8: DSP_curve of a third order $\Sigma\Delta$ CRFB architectured modulator

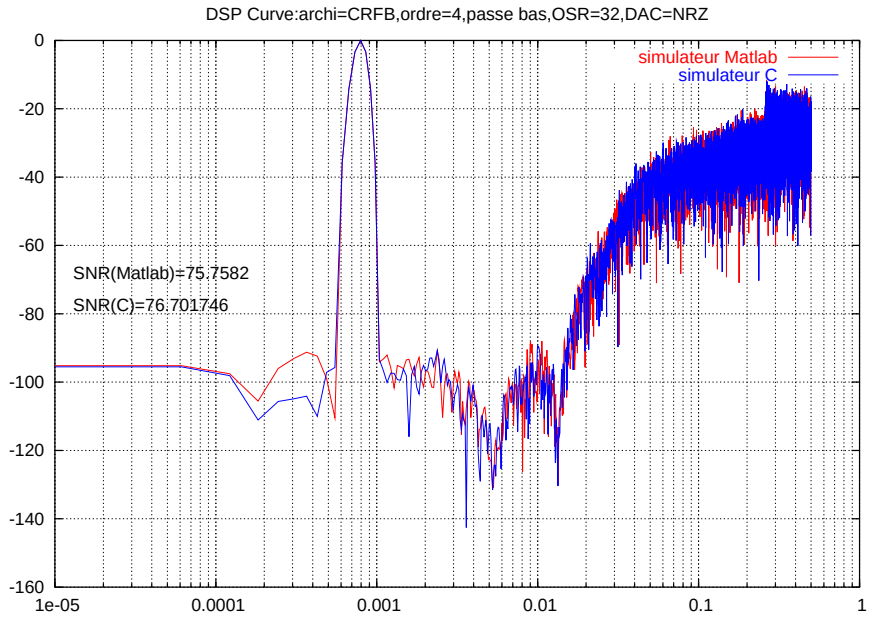


Figure 6.9: DSP_curve of a fourth order $\Sigma\Delta$ CRFB architectured modulator

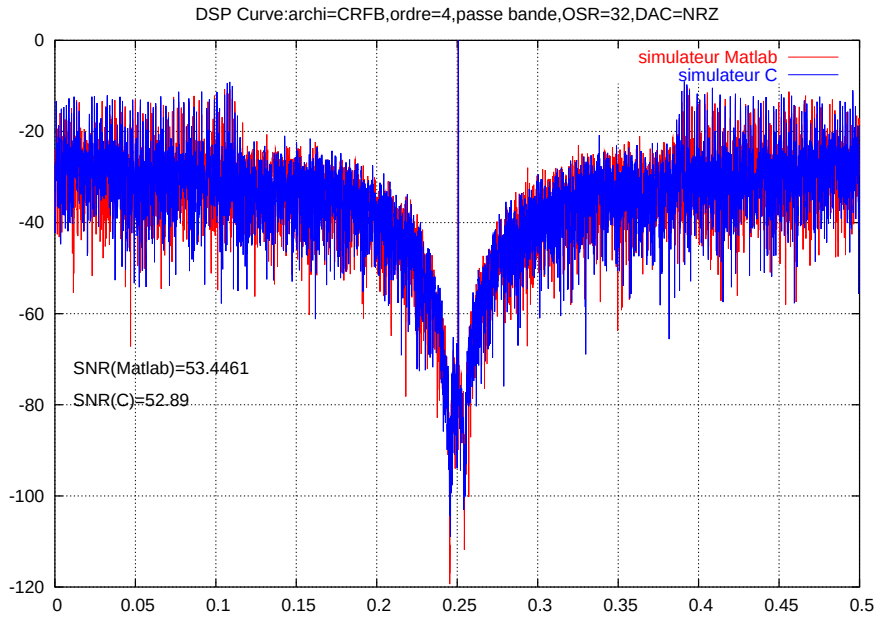


Figure 6.10: DSP_curve of a fourth order pass band Σ Δ CRFB architectured modulator

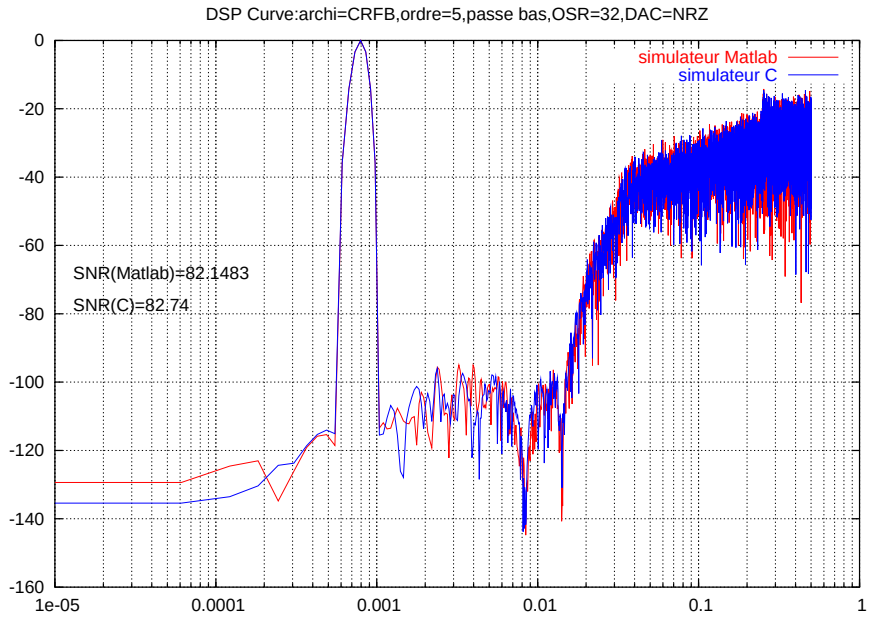


Figure 6.11: DSP_curve of a fifth order Σ Δ CRFB architectured modulator

3. CIFF architecture

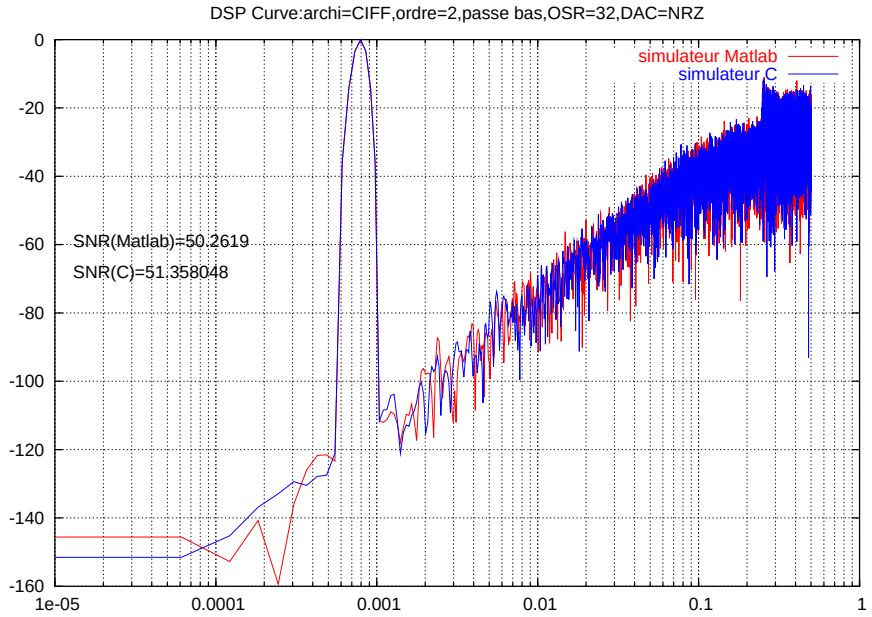


Figure 6.12: DSP_curve of a second order Σ Δ CIFF architectured modulator

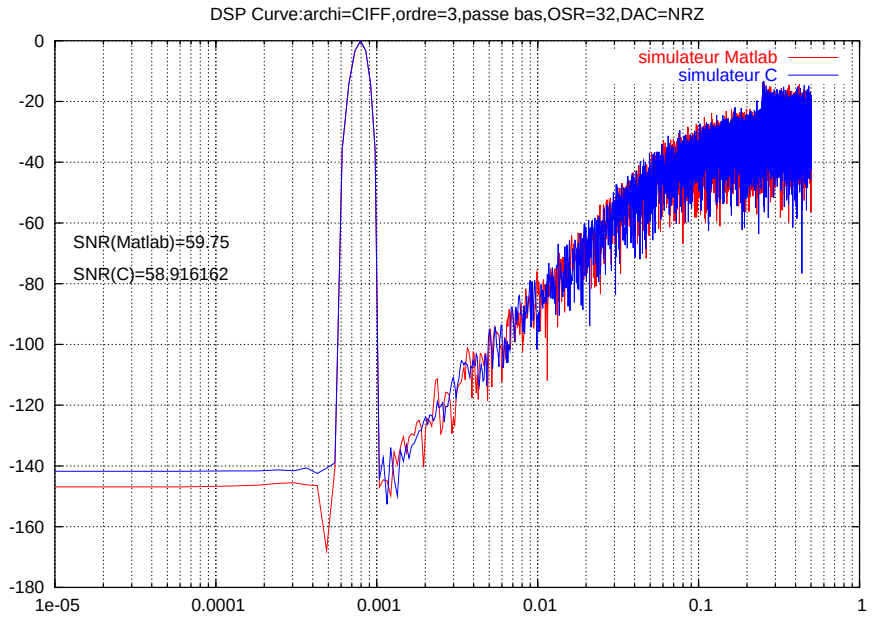


Figure 6.13: DSP_curve of a third order Σ Δ CIFF architectured modulator

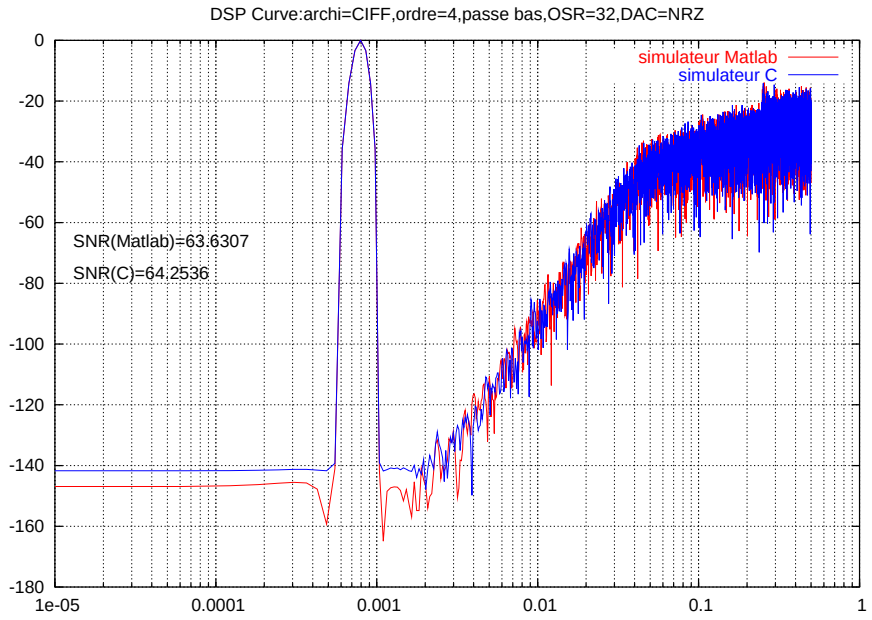


Figure 6.14: DSP_curve of a fourth order $\Sigma\Delta$ CIFF architectured modulator

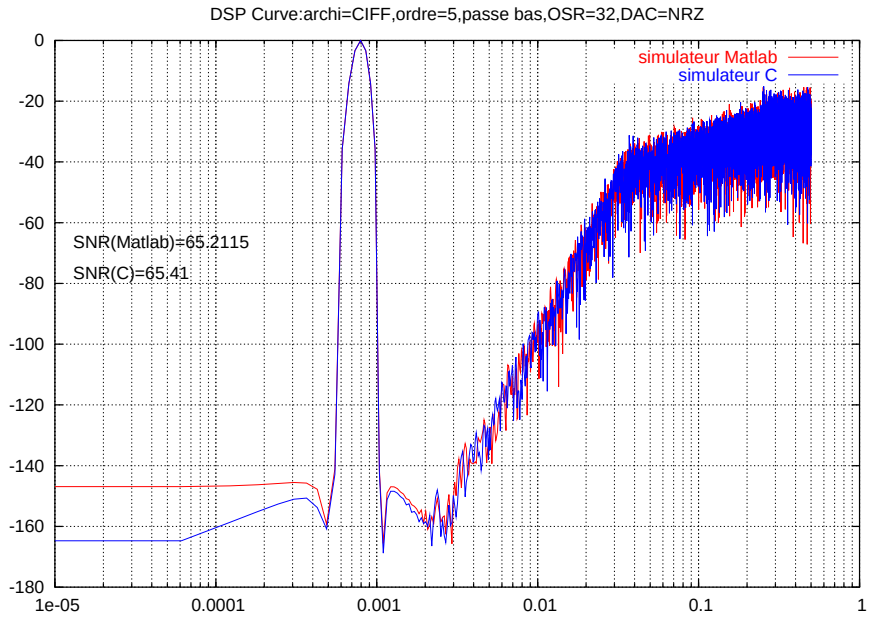


Figure 6.15: DSP_curve of a fifth order $\Sigma\Delta$ CIFF architectured modulator

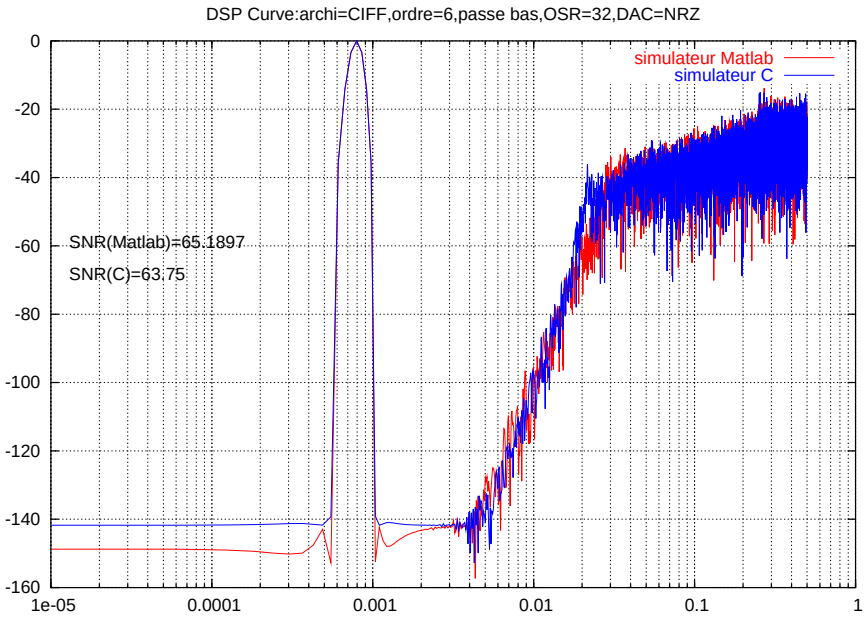


Figure 6.16: DSP_curve of a sixth order Σ Δ CIFF architectured modulator

4. CRFF architecture

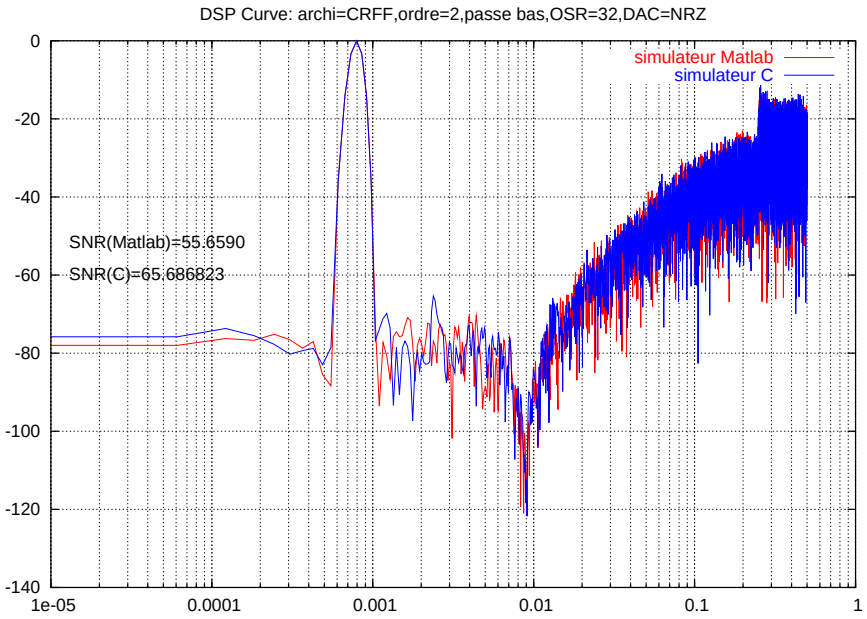


Figure 6.17: DSP_curve of a second order $\Sigma\Delta$ CRFF architectured modulator

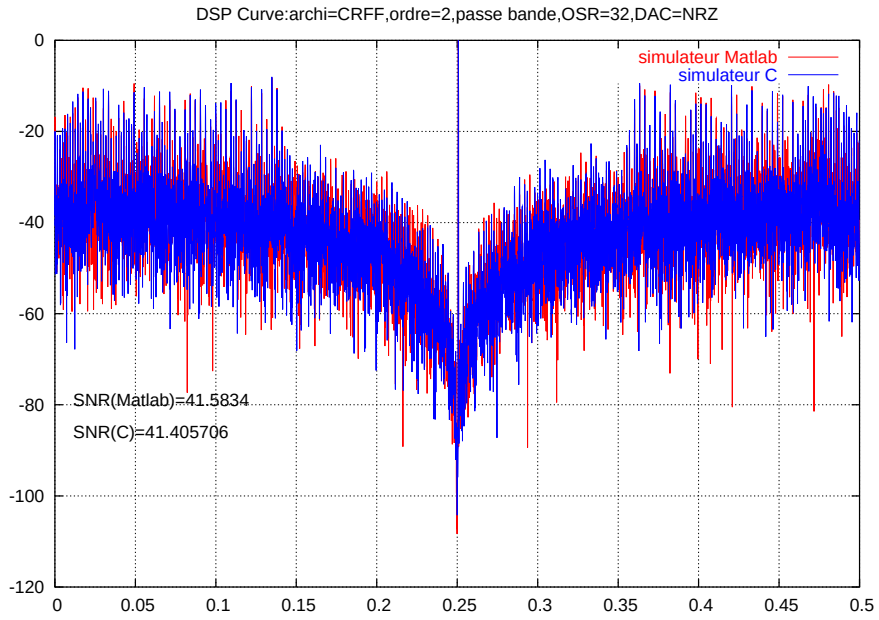


Figure 6.18: DSP curve of a second order pass band Σ Δ CRFF architectured modulator

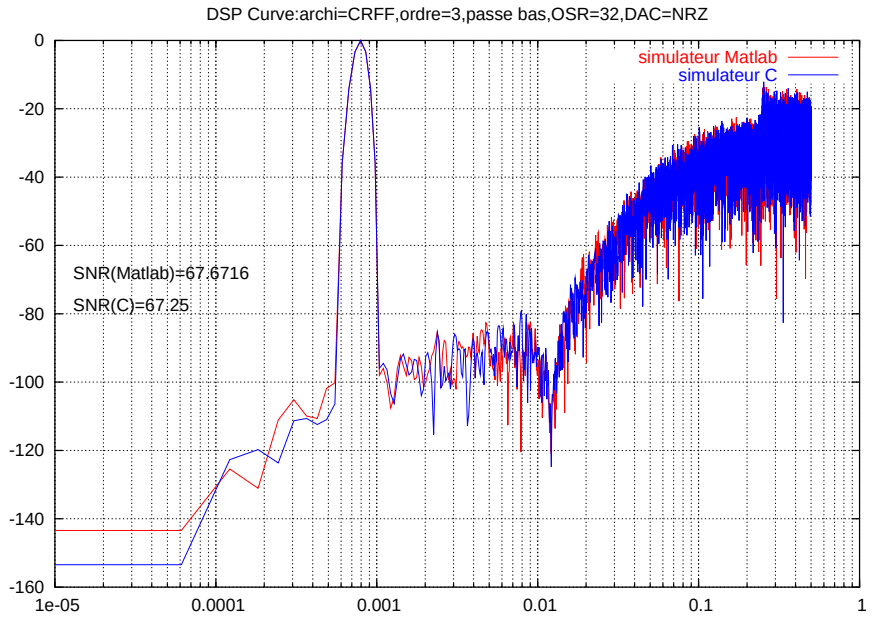


Figure 6.19: DSP curve of a third order Σ Δ CRFF architectured modulator

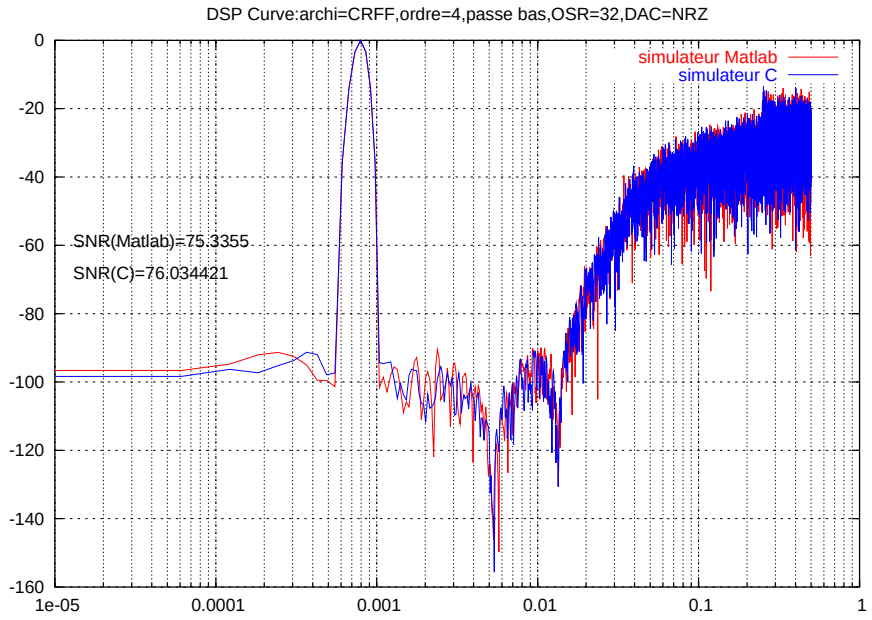


Figure 6.20: DSP curve of a fourth order Σ Δ CRFF architectured modulator

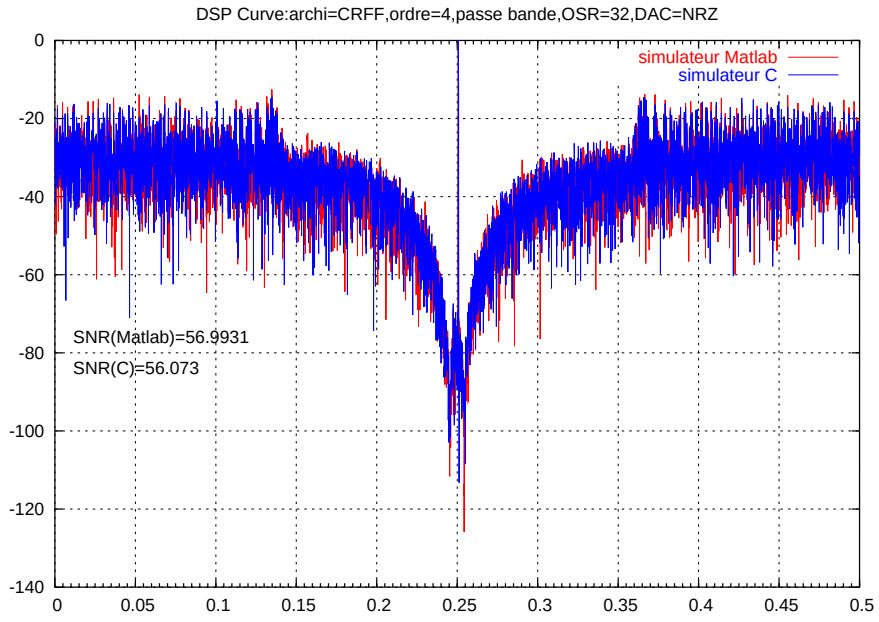


Figure 6.21: DSP curve of a fourth order pass bandΣ Δ CRFF architectured modulator

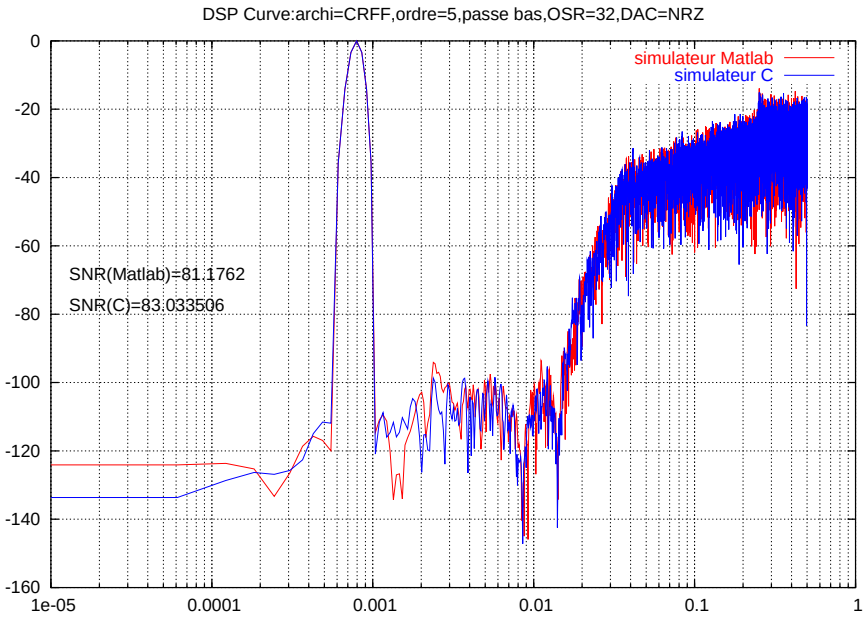


Figure 6.22: DSP curve of a fifth order $\Sigma\,\Delta$ CRFF architectured modulator

6.2 SNR Curves

1. CIFB architecture

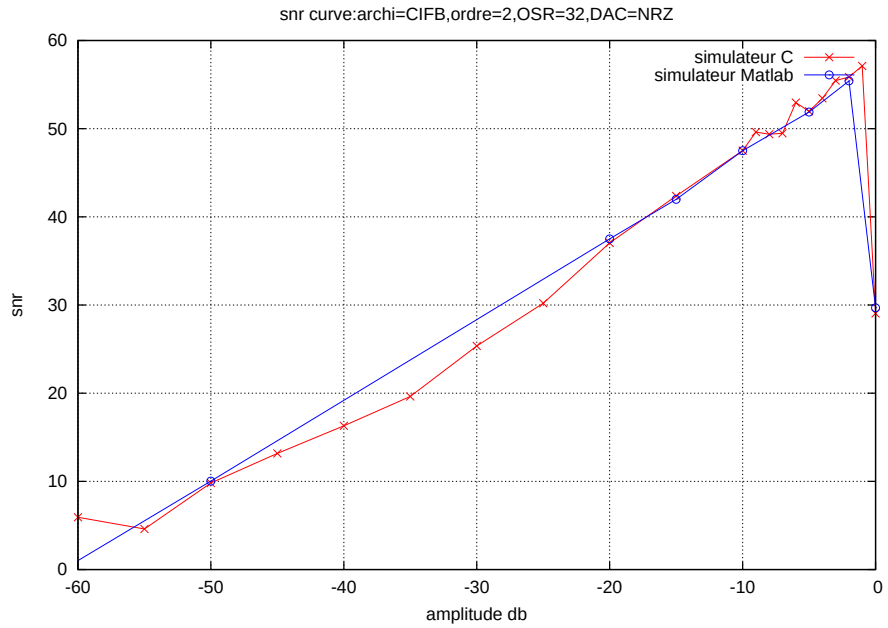


Figure 6.23: SNR curve of a second order Σ Δ CIFB architectured modulator

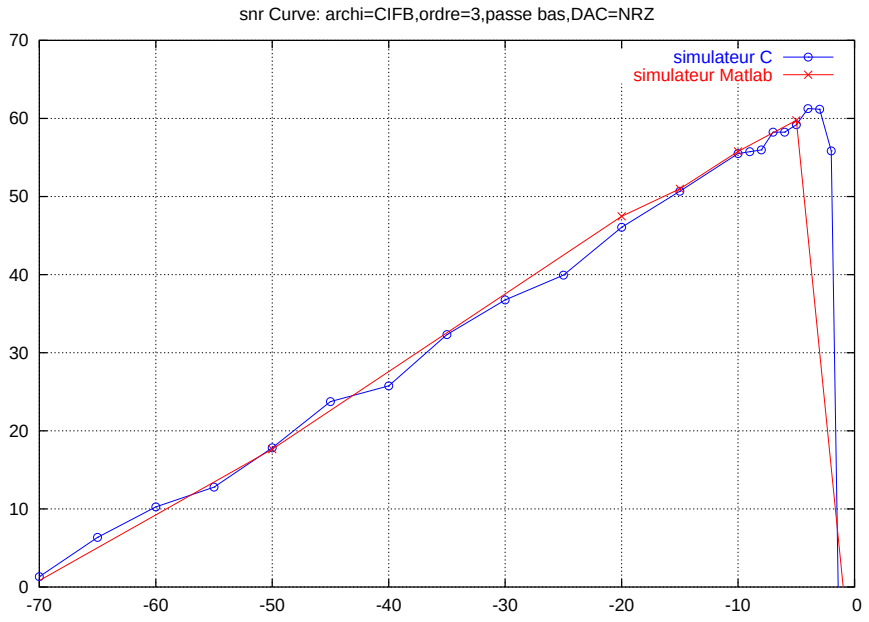


Figure 6.24: SNR curve of a third order Σ Δ CIFB architectured modulator

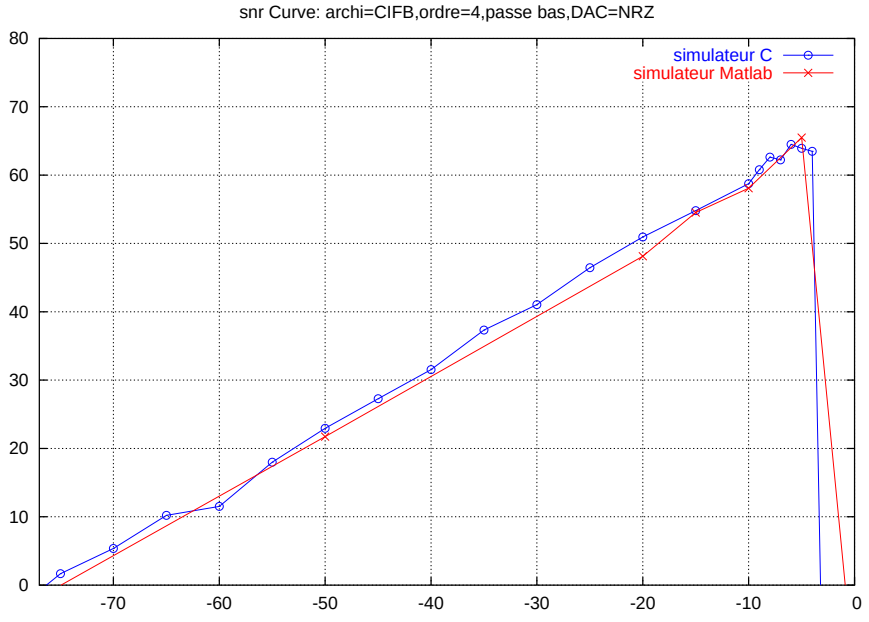


Figure 6.25: SNR curve of a fourth order Σ Δ CIFB architectured modulator

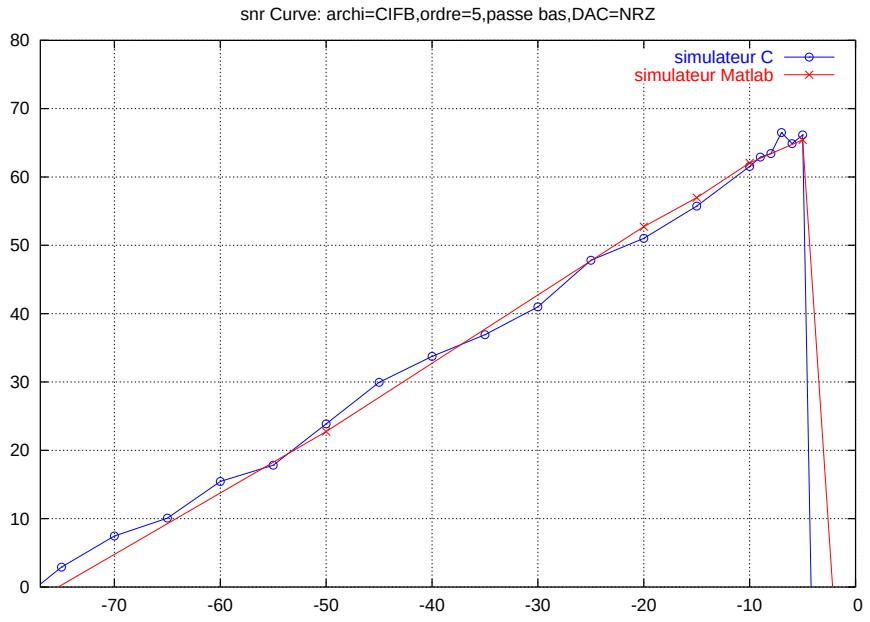


Figure 6.26: SNR curve of a fifth order Σ Δ CIFB architectured modulator

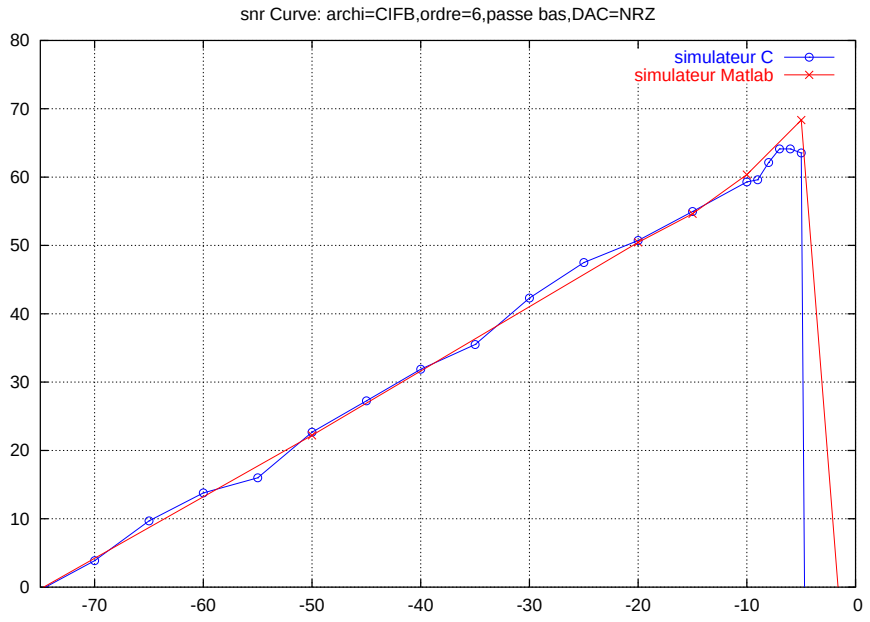


Figure 6.27: SNR curve of a sixth order Σ Δ CIFB architectured modulator

2. CRFB architecture

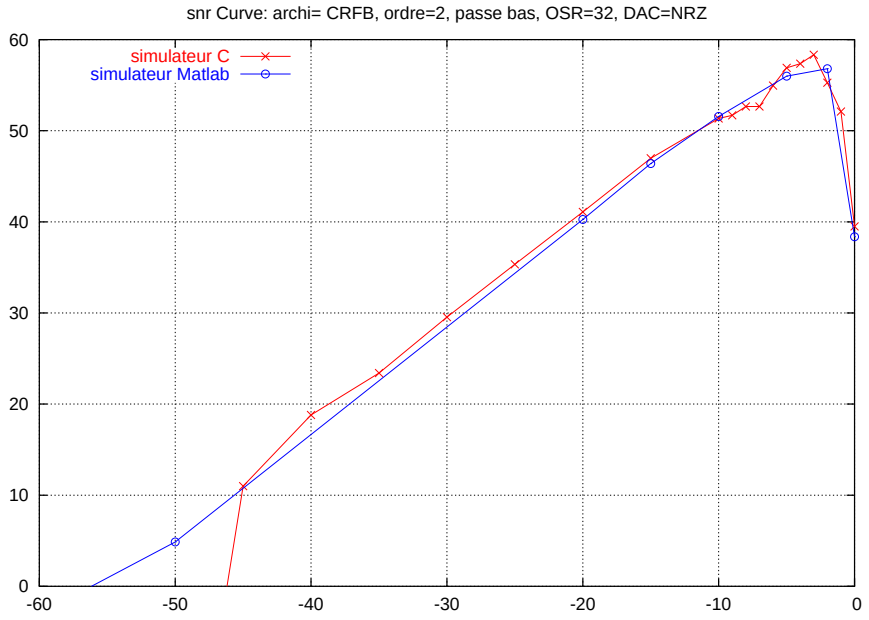


Figure 6.28: SNR curve of a second order Σ Δ CRFB architectured modulator

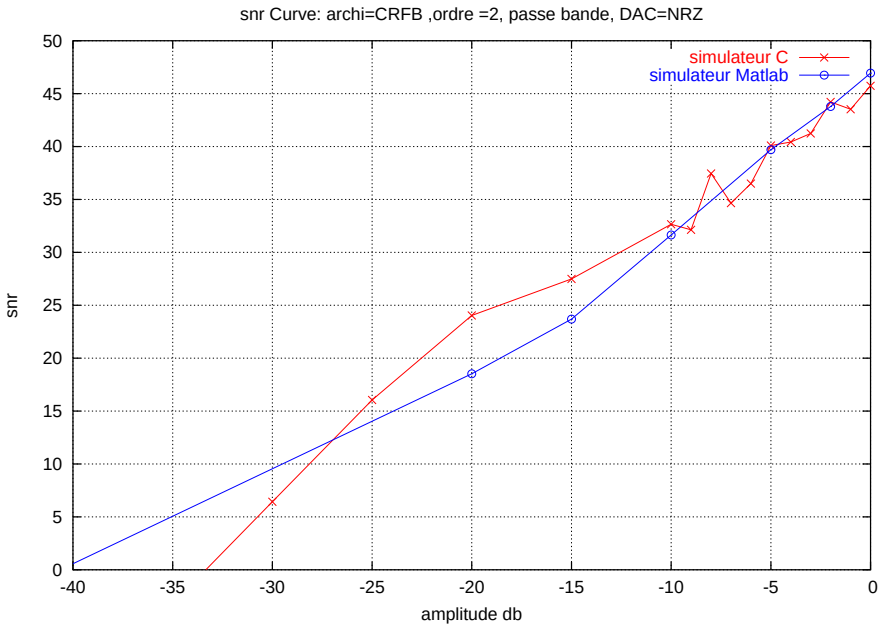


Figure 6.29: SNR curve of a second order pass band Σ Δ CRFB architectured modulator

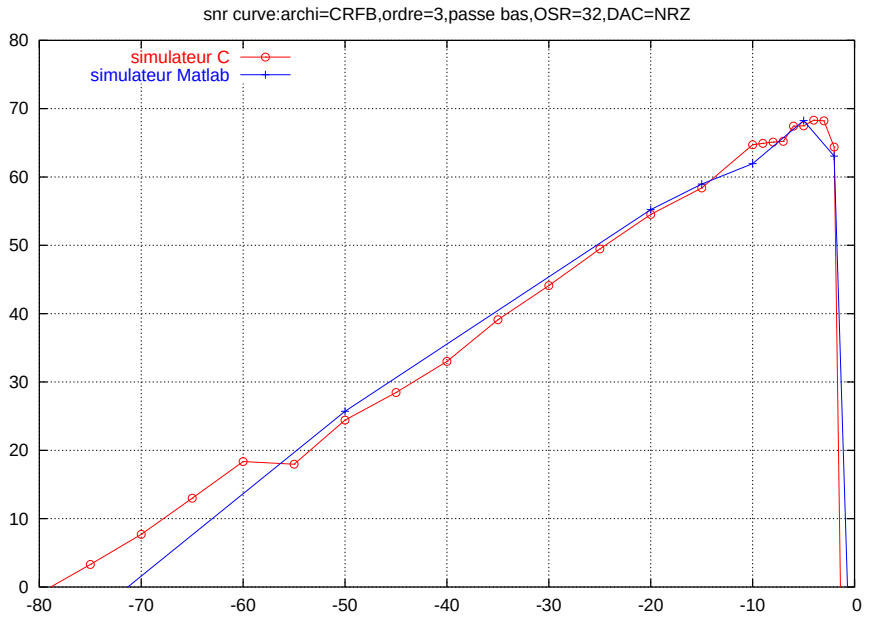


Figure 6.30: SNR curve of a third order $\Sigma\,\Delta$ CRFB architectured modulator

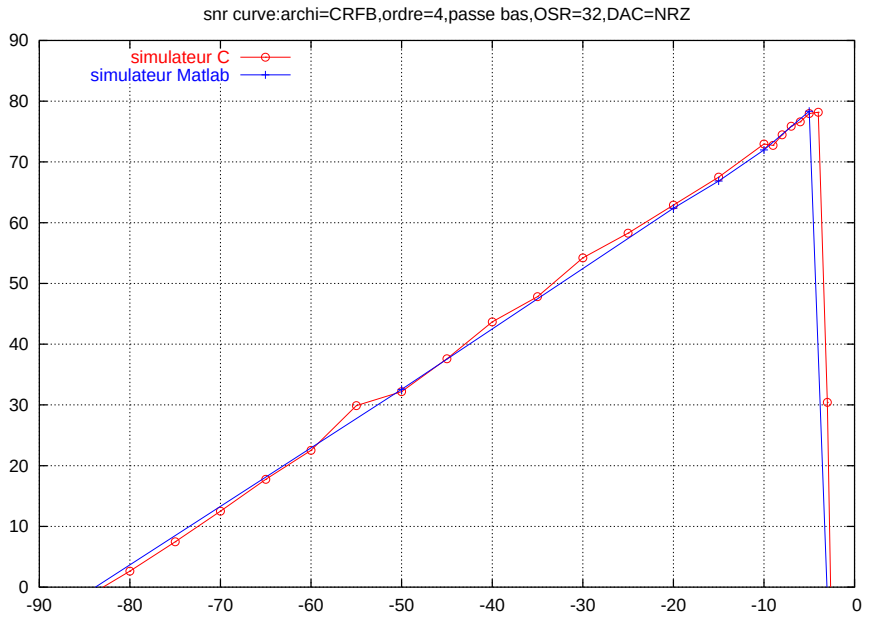


Figure 6.31: SNR curve of a fourth order $\Sigma\,\Delta$ CRFB architectured modulator

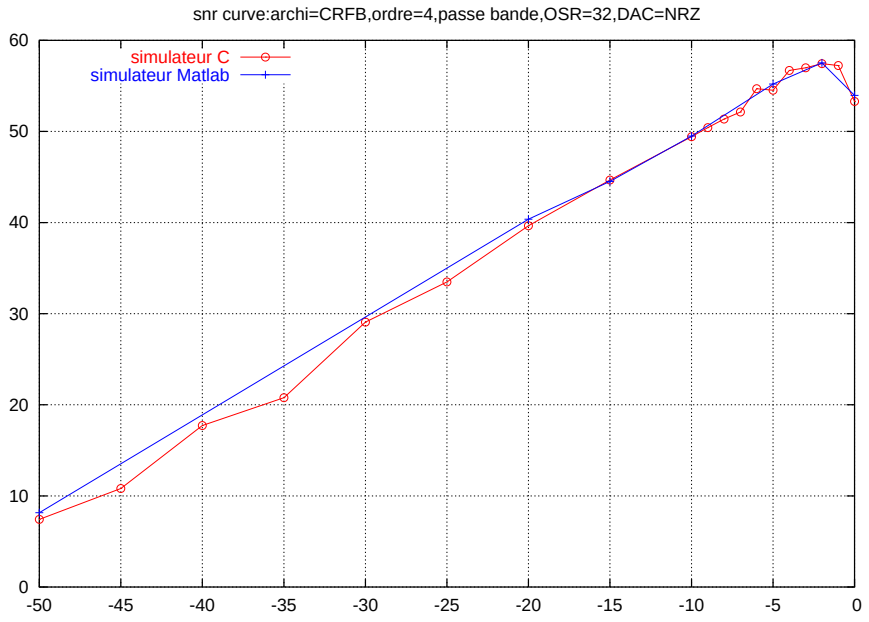


Figure 6.32: SNR curve of a fourth order pass band Σ Δ CRFB architectured modulator

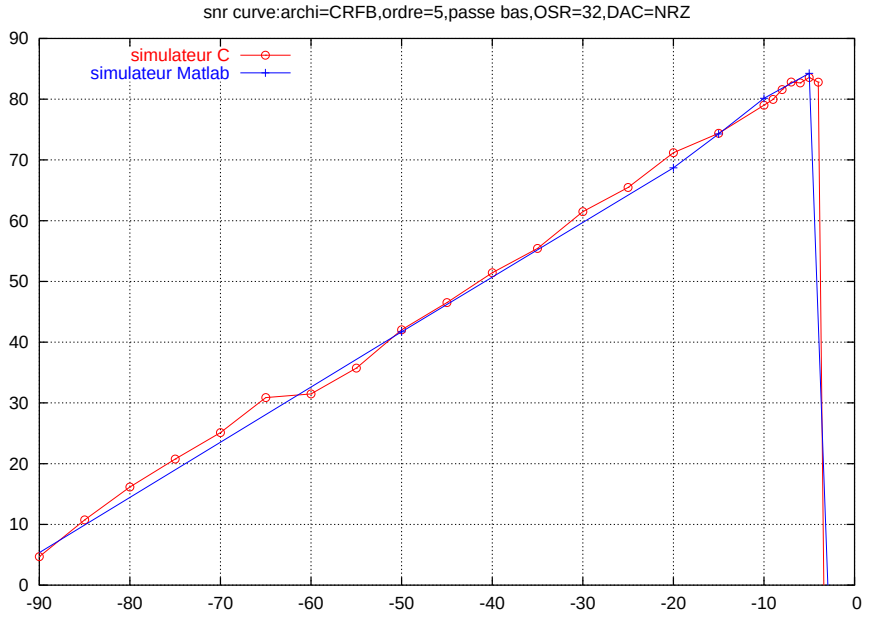


Figure 6.33: SNR curve of a fifth order Σ Δ CRFB architectured modulator

3. CIFF architecture

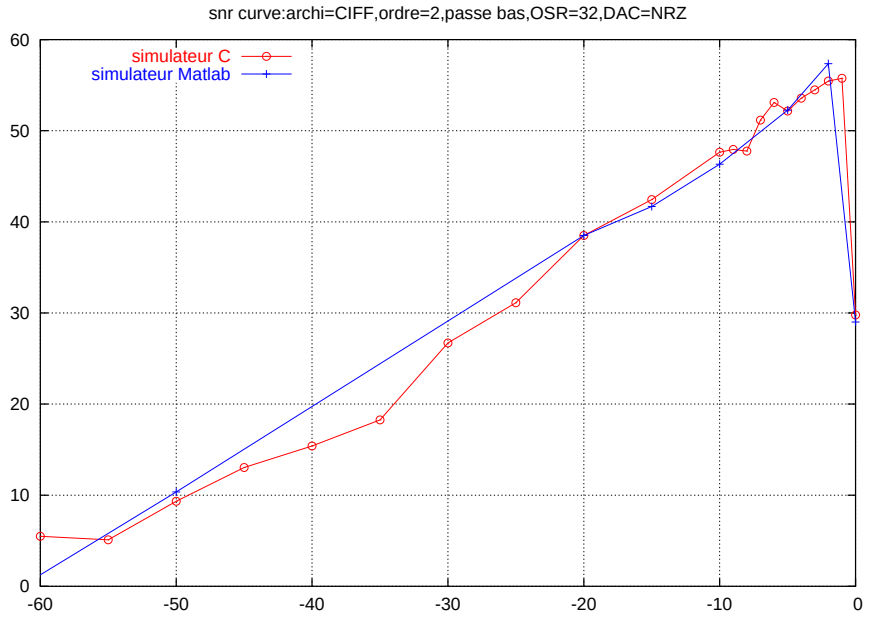


Figure 6.34: SNR curve of a second order Σ Δ CIFF architectured modulator

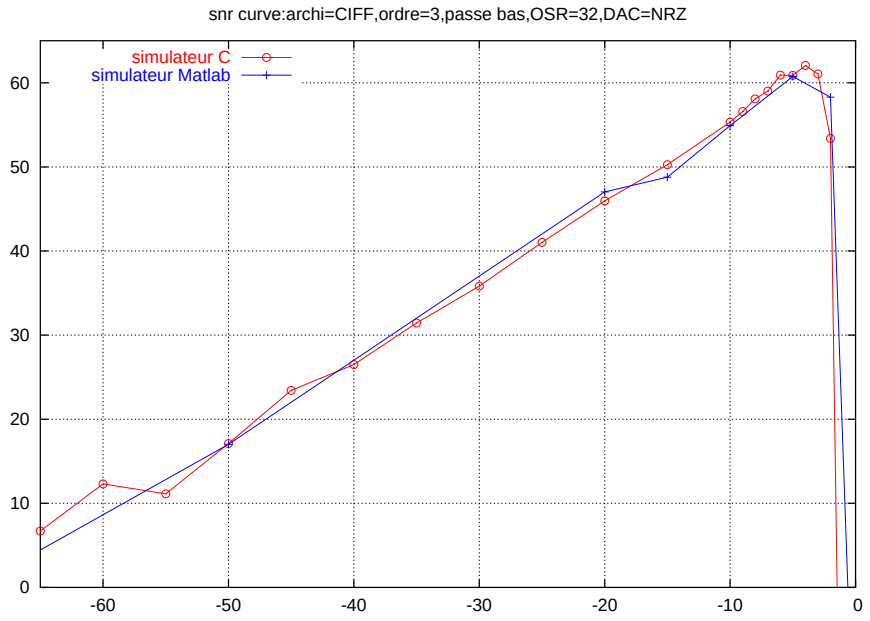


Figure 6.35: SNR curve of a third order Σ Δ CIFF architectured modulator

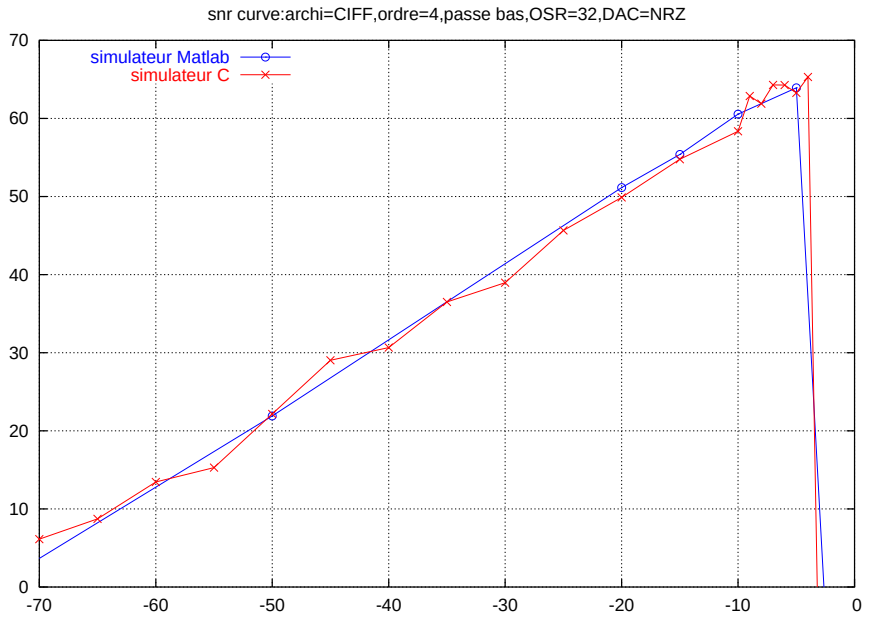


Figure 6.36: SNR curve of a fourth order $\Sigma\Delta$ CIFF architectured modulator

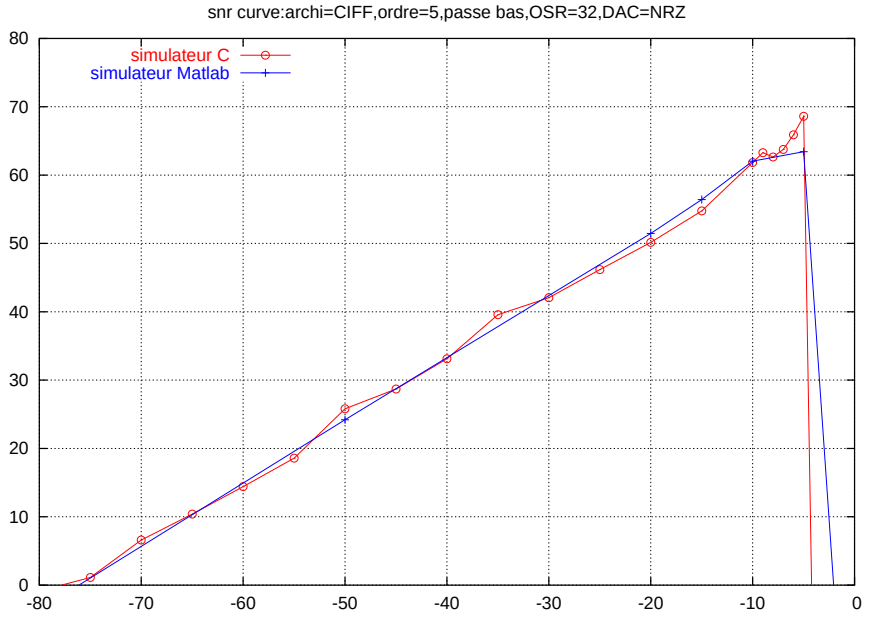


Figure 6.37: SNR curve of a fifth order $\Sigma\Delta$ CIFF architectured modulator

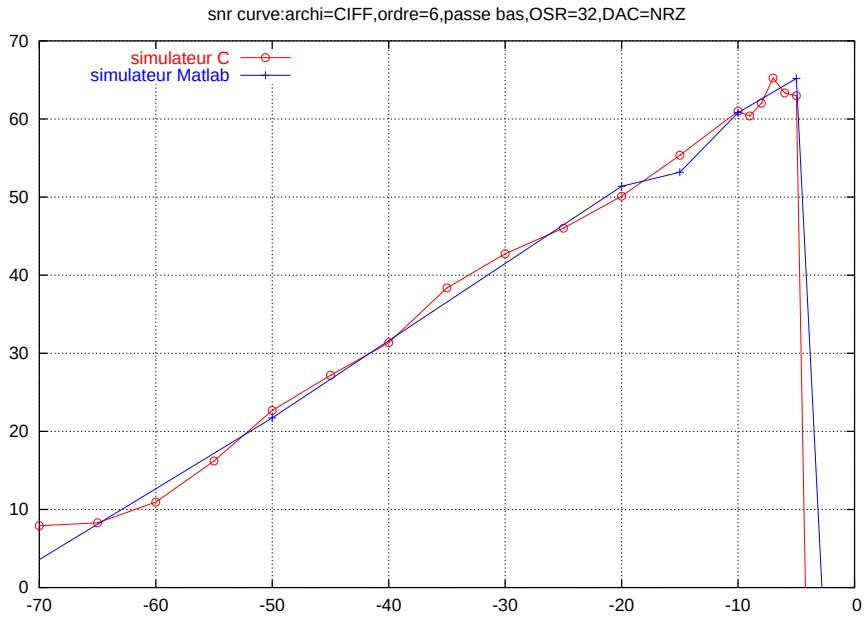


Figure 6.38: SNR curve of a sixth order $\Sigma\,\Delta$ CIFF architectured modulator

4. CRFF architecture

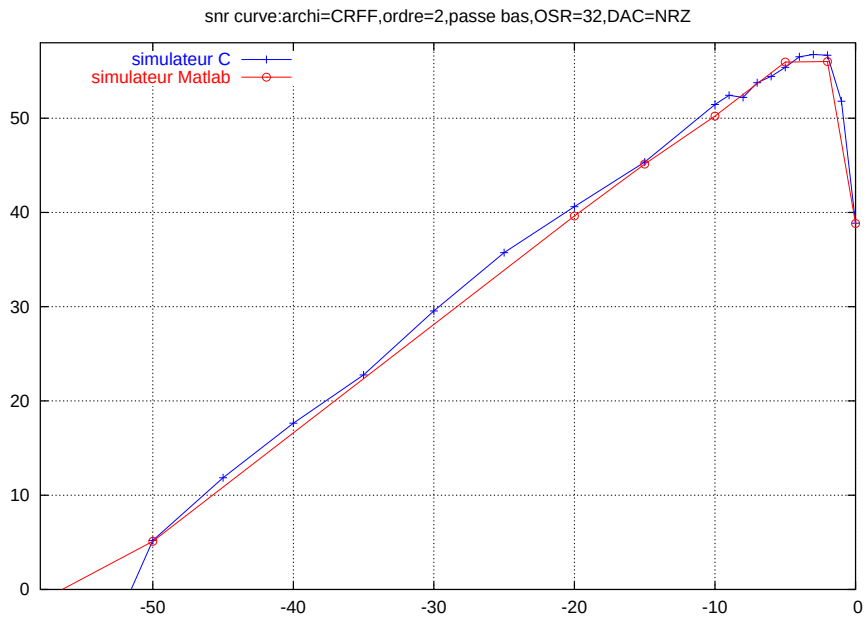


Figure 6.39: SNR curve of a second order $\Sigma\,\Delta$ CRFF architectured modulator

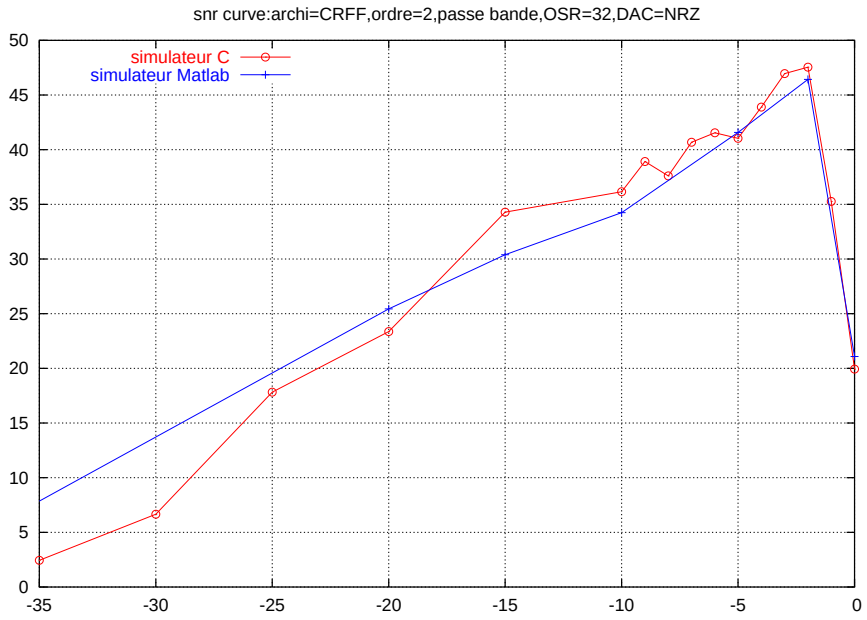


Figure 6.40: SNR curve of a second order pass band Σ Δ CRFF architectured modulator

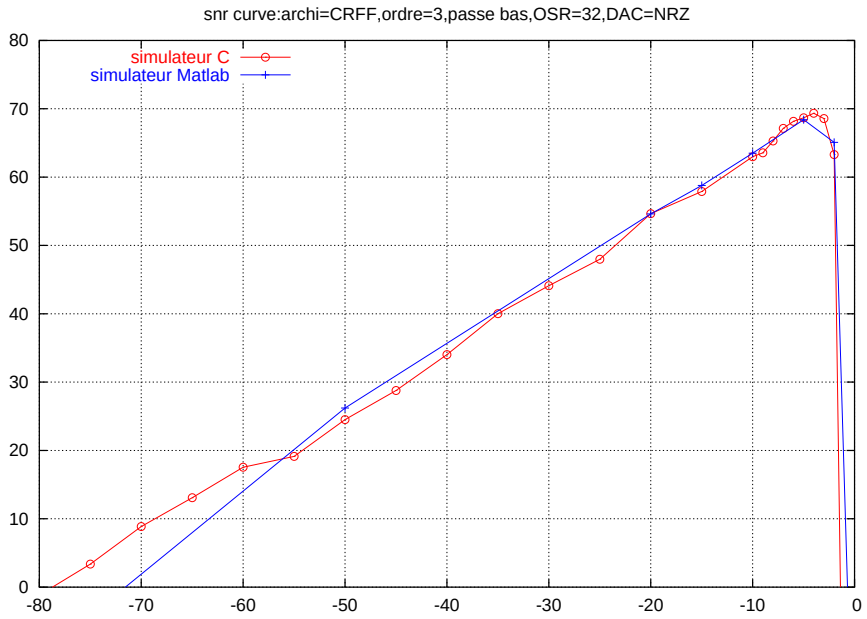


Figure 6.41: SNR curve of a third order Σ Δ CRFF architectured modulator

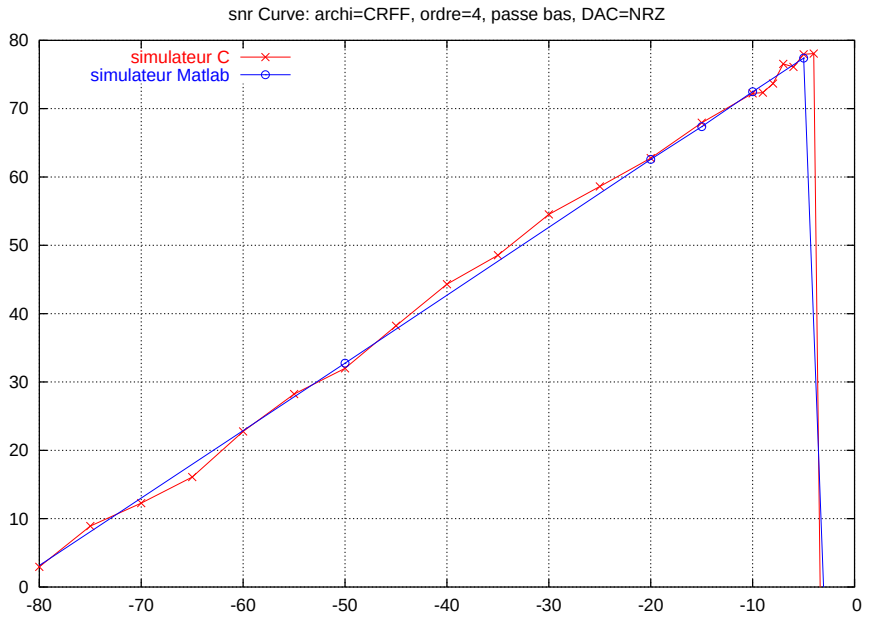


Figure 6.42: SNR curve of a fourth order Σ Δ CRFF architectured modulator

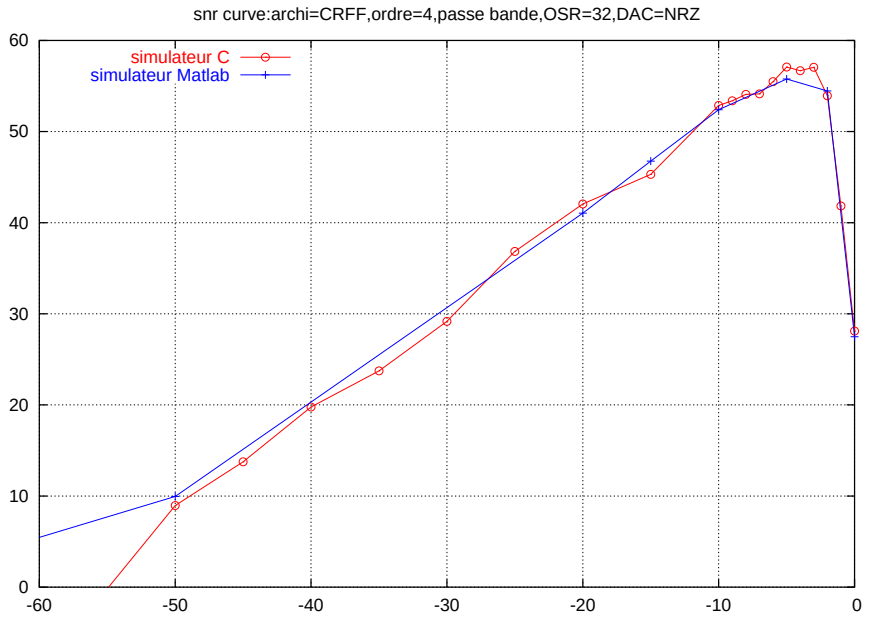


Figure 6.43: SNR curve of a fourth order pass band Σ Δ CRFF architectured modulator

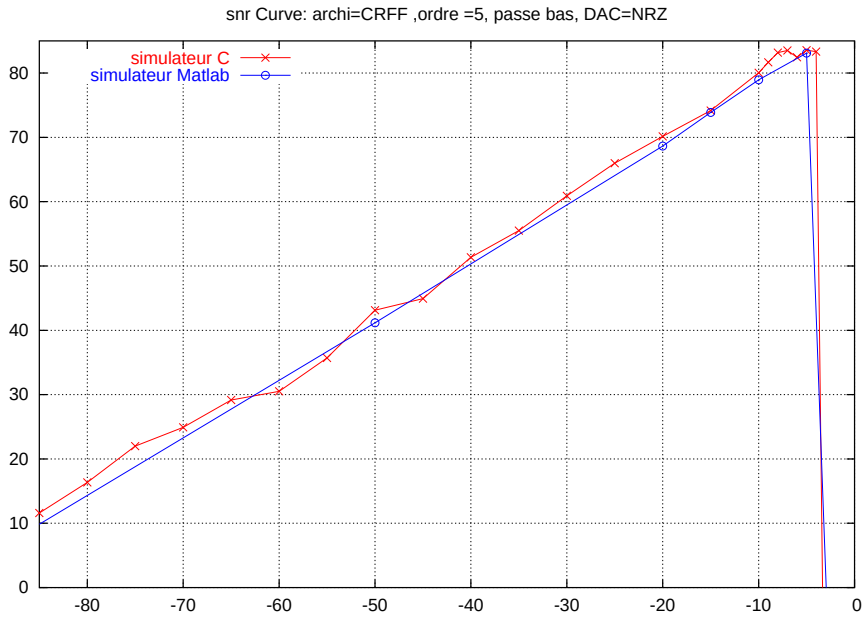


Figure 6.44: SNR curve of a fifth order pass band $\Sigma\Delta$ CRFF architectured modulator

6.3 Comparision between the time execution of the C and the Matlab simulators

Order	Time execution (s)(matlab)	Time execution (s) (C)	step size (for C)
2	15.063	3.77	0.01
3	40.093	1.84	0.1
4	54.36	1.85	0.1
5	65.625	1.95	0.1
6	4.328	5.95	0.01

Table 6.1: Time execution of the Matlab and C simulator for the CIFB architectured converter

Order	Time execution (s)(matlab)	Time execution (s) (C)	step size (for C)
2 (low pass converter)	12.3750	3.83	0.01
2 (pass band converter)	455.5470	6.03	0.005
3	15.031	1.84	0.1
4 (low pass converter)	35.75	1.89	0.1
4 (pass band converter)	896.968	32.83	0.001
5	23.65	1.93	0.1

Table 6.2: Time execution of the Matlab and C simulator for the CRFB architectured converter

Order	Time execution (s)(matlab)	Time execution (s) (C)	step size (for C)
2	14.6090	3.83	0.01
3	109.89	27.9	0.001
4	184.921	4.66	0.01
5	212.87	40.08	0.001
6	209.14	5.89	0.01

Table 6.3: Time execution of the Matlab and C simulator for the CIFF architectured converter

Order	Time execution (s)(matlab)	Time execution (s) (C)	step size (for C)
2 (low pass converter)	17.25	1.81	0.1
2 (pass band converter)	276.296	24.03	0.001
3	112.36	27.31	0.001
4 (low pass converter)	200.563	4.61	0.01
4 (pass band converter)	589.9070	32.46	0.001
5	207.516	1.9	0.1

Table 6.4: Time execution of the Matlab and C simulator for the CRFF architectured converter

Bibliography

[1] Richard Schreier. ”An Empirical study of high-order single-bit Delta-Sigma modulators”. *IEEE Transactions on circuits and systems*, vol.40(No. 8):461–466, August 1993.

[2] Truong-Thao Nguyen. *”Deterministic Analysis of Oversampled A/D Conversion and $\Sigma\,\Delta$ modulation,and decoding improvements using consistent estimates”*. PhD thesis, Columbia University, 1993.

[3] Robert M. Gray. ”Quantization Noise Spectra”. *IEEE Transactions on information theory*, vol.36(No. 6):1220–1244, November 1990.

[4] H.Aboushady. ”Simulation et conception automatique de modulateurs $\Sigma\,\Delta$ courants commuts”. Master’s thesis, ENST, 1996.

[5] H. Aboushady. *”Conception en vue de la réutilisation de convertisseurs analogiques-numriques $\Sigma\Delta$ temps continu mode courant ”*. PhD thesis, L’université Paris VI, 2002.

[6] M.M. Louërat N.Beilleau, H.Aboushady. ”Systematic Approach for Scaling Coefficients of Discrete-Time and Continuous-Time Sigma-Delta Modulators”. *MWSCAS*, 2003.

[7] R. Schreier T. Shui and F. Hudson. ”Mismatch-shaping DAC for low pass and bandpass mutibit Δ-Σ modulators ”. *IEEE Int. Symp. Circuits and Systems*, vol. 1:352–355, June 1998.

[8] R. Schreier T. Shui and F. Hudson. ”Mismatch-shaping for a current-mode multibit Δ-Σ DAC”. *IEEE Journal of Solid-State Circuits*, vol. 34(No. 3):331–338, March 1999.