Università degli Studi di Milano-Bicocca

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Triennale in Fisica

Matlab Modelling of a Chopper Preamplifier for Proton Sound Detectors

Relatore:
Prof. Marcello De Matteis

Candidato:
Sergio Verga

Correlatore:
Dott. Elia Arturo Vallicelli

Matricola:
859200

Anno Accademico 2021-2022

Abstract

Proton Sound Detectors (ProSDs) are specific devices with interesting perspectives in hadron therapy treatments. This therapeutic approach has an important advantage from the classical photon-based radiation therapy, since it releases the maximum energy at the end of its penetration path (i.e., range or Bragg peak) instead of depositing the maximum dose at the body interface. In this way, a deeper treatment for cancer is allowed reducing severe damages to the superficial tissues.
Until now, most of popular experimental verification methods for proton beam ranges, such as Positron Emission Tomography (PET) and Prompt Gamma Rays (PGR), are nuclear imaging techniques.
ProSD offers a competitive alternative based on the ionoacoustic effect, which exploits the acoustic pulse generated by the fast energy deposition at the Bragg Peak region, improving the measurement precision down to sub-millimetric scale. One of the main problems of ProSds is the extremely weak pressure signal at the acoustic sensor, that leads to poor Signal-to-Noise Ratio (SNR) performances (for instance, 0 dB at 200 MeV proton energy for approximately 50 kHz acoustic pulse fundamental frequency, where flicker noise power dominates).
This Bachelor's thesis work aims at overcoming this limitation by strongly reducing 1/f or flicker noise power adopting an advanced Chopper Preamplifier analog circuit (CP), based on the Chopper Technique (CT).
More specifically, this thesis focuses on a complete Matlab model of the CP with the main aim to study the impact of noise sources as a function of its characteristic parameters (carrier signal frequency, filter cutoff frequency, and filter order). The noise signals included in the model are thermal and 1/f noise, which is the dominant noise contribution in the frequency band of the signal ($\sim 50\,kHz$) and in general at low frequencies.
The CT is implemented by a complex cascade of stages with the purpose of attenuating low-frequency noise power with a filter. In detail, the input signal (driving the relevant information coming from the sensor) is modulated to higher frequency (by multiplication for the m(t) signal, as illustrated in Figure 1). Subsequently, the noise signal is added to the input of a classical continuous time aplifier. Then, the output signal of the amplifier is demodulated to the base band and at the same time the flicker noise signal is translated to higher frequency. Finally, the high frequency flicker noise is filtered out by a proper lowpass filter.
In Matlab, the model is configured as a modular programming system that generates each signal in a corresponding function. It is possible to visualize the chain of steps with time and frequency domain plots of input/output signals at each stage.
In the final part of the main program, the RMS values at the end of the circuit are calculated by integration until the filter cutoff frequency, to verify the correct CP modelling.

In conclusion, a noise signal processed by the CP with an input signal of 40 kHz and a cutoff frequency of 60 kHz exhibits an amplitude of 0.29 mV_{RMS}, instead of 1.25 mV_{RMS} without applying CT.

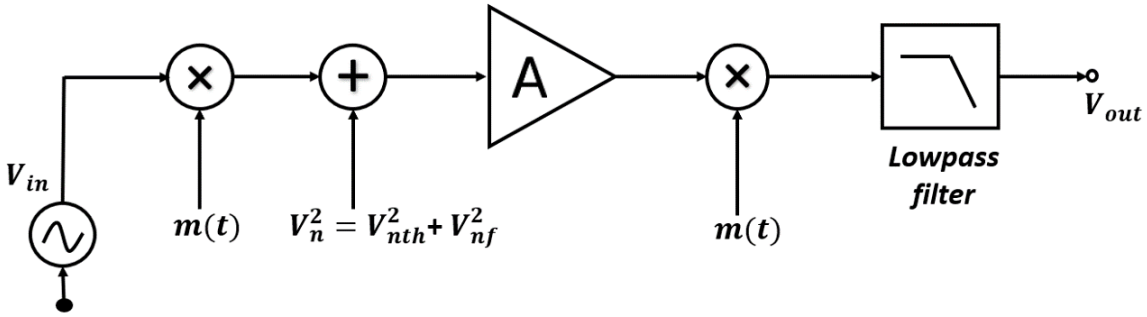

Figure 1: Concept of the Chopper Technique.

Contents

Introduction

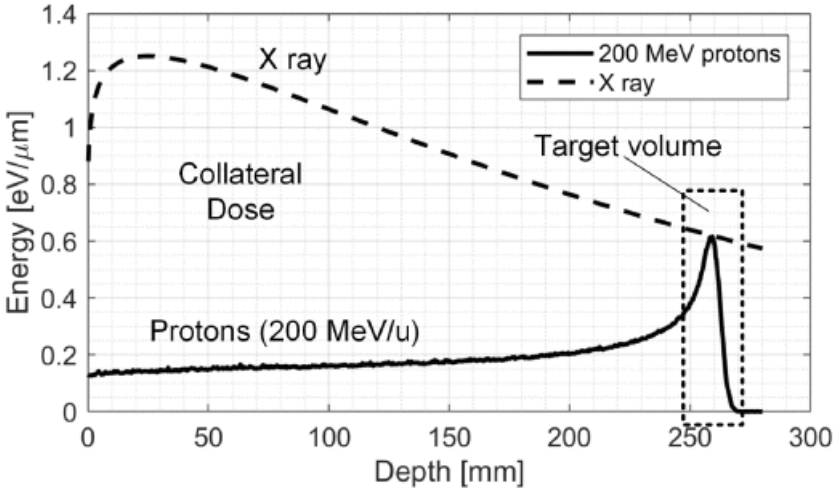

Figure 1.1: Dose distributions of protons and X-ray irradiation simulated in water [1].

Energy deposition in an absorber material volume leads to the generation of propagating pressure waves in the medium. This emission is known as the *thermoacoustic effect*, and it has several interesting perspectives in medical physics. An extremely important application is hadron therapy, thanks to the advantages it offers compared to other solutions.

In contrast to classical photon-based radiation therapy, the use of clinical ion beams has a remarkable difference in the depth of maximum energy deposition. Photon-based radiation releases maximum energy at the body interface, while clinical ion beams deposit maximum energy at the *Bragg peak* (BP) and a negligible dose behind. A qualitative difference in the behavior of the dose vs. tissue depth in case of a 200 MeV proton beam and X-rays is shown in Figure 1.1.
With this approach, it is possible to specifically target the DNA of cancerous cells within the body. Instead, in case of photon-based radiation, the tissue surface is hit with even more energy than the one actually used to destroy cancerous cells DNA, meaning severe damage that is not provoked with ions beams. The use of clinical radiation from ion beams for therapy requires low-noise electronic systems to detect the pressure waves produced in the thermoacoustic effect, in particular *Proton Sound Detectors* (ProSDs) [1] [2].

1.1 Proton Sound Detectors

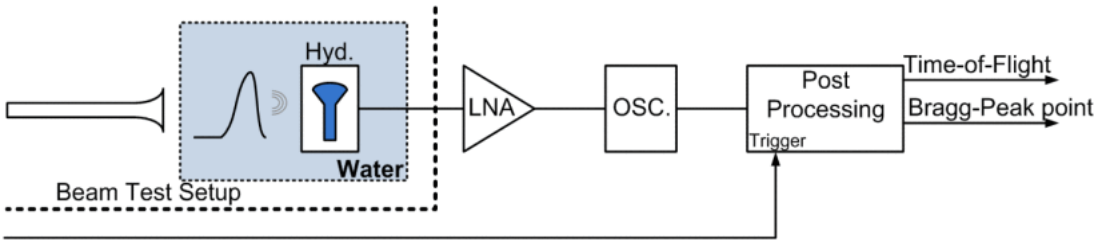

Figure 1.2: Schematic of a ProSD [2].

ProSDs are chains of acquisition consisting of the stages shown in Figure 1.2: The first acquisition of the signal occurs when the piezoelectric transducer converts a pressure signal into a voltage oscillation of proportional amplitude. Then, the signal passes across the Low Noise Amplifier (LNA), i.e. the CP, in order to be amplified up to a proper level.

For this application, in the example with a 200 MeV proton beam, the *Signal-to-Noise-Ratio* (SNR) is 0 dB, meaning that the signal power is approximatingly of the same power as the noise signal. In addition, it is not possible to increase the power of the signal, as it is a medical application: it would imply the irradiation of the patient with a higher dose or to move the sensor closer. Therefore, it is essential to limitate the impact of noise sources on the BP detection, which means to tone down the noise power implied in the amplification process, i.e. the resulting noise from the thermal contribution and the 1/f one (particularly intense at low frequencies). To overcome this problem, it is possible to adopt a CP analog circuit, based on the Chopper Technique (CT) [1] [2].

The Chopper Technique

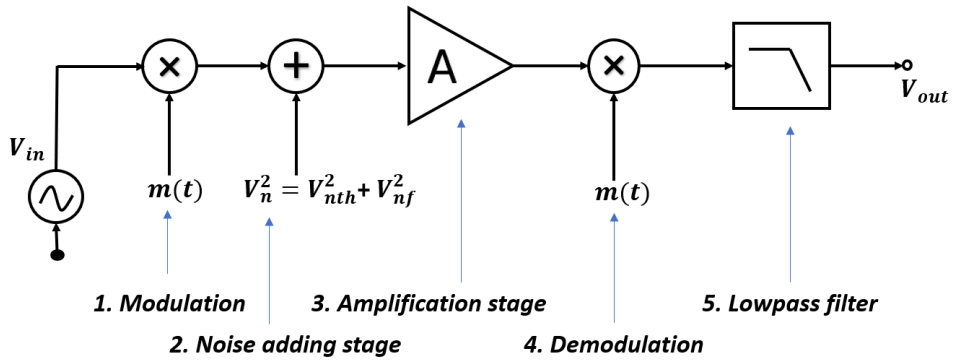

Figure 2.1: Schematic of the CT.

The cascade of stages involved in the CT is shown in Figure 2.1. The purpose of the CP is to tone down the 1/f noise power contribution, that is the main limitation to the signal processing.
In detail, the entire cascade of stages is the consequential combination of the following parts:

1. ***Modulation stage***: in the frequency domain, this operation implies a translation from the base band to the so-called translated band, centered on the frequency of the carrier signal. This operation affects the input signal V_{in}.

2. ***Noise-adding stage***: in this phase, the noise signal resulting from the flicker and the thermal contribution is added to the signal prior to the amplification stage (as that is the source of the flicker noise).

3. ***Amplification stage***: the signal crosses the time-continuous amplifier characterized by a gain G.

4. ***Demodulation stage***: in the frequency domain, this stage demodulates the tone of the original signal to the base band. At the same time, it implies a translation of the 1/f noise contribution to a band centered on the frequency of the carrier signal.

5. ***Lowpass filter***: this stage filters the flicker noise in the translated band.

The combination of these consequential stages leads to an amplified output signal V_{out} with a toned-down flicker noise effect [3].

2.1 Noise

One of the problems in the realization of a LNA for ProSDs is noise, since detection of BP is based on the acquisition of signals with an approximately unitary SNR, as shown with the 200 MeV proton beam application [2].

The noise affecting the circuit is the result of more contributions: thermal noise (also called white noise) and 1/f noise.
White noise is a random signal that in the frequency domain is approximately constant throughout the bandwidth.
Flicker noise, instead, has a *Power Spectral Density* (PSD) characterized by a 1/f dependence.

To obtain the resulting noise signal, a squared sum in the frequency domain is necessary, as follows:

$$V^2_{n\_freq} = V^2_{n1\_freq} + V^2_{n2\_freq}$$

It is then possible to obtain the RMS value considering the square root of the power signal integrated in the frequency bandwidth and to obtain the time domain representation by applying the inverse Fourier transform to the frequency domain representation of the signal.
The operation to derive the RMS values is the following:

$$RMS = \sqrt{\int_0^\infty S(f)df} \quad [V_{RMS}]$$

[4]

2.2 1/f Noise

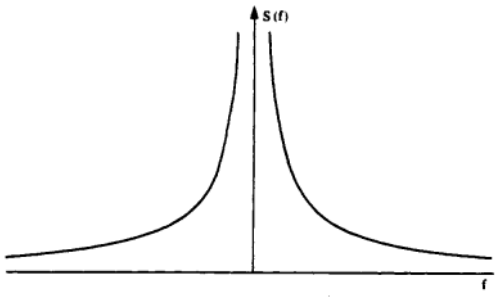

Figure 2.2: PSD of flicker noise [5].

1/f noise, also known as flicker noise, is a noise signal characterized by a PSD $\propto 1/f$, as illustrated in Fig 2.2. In particular, a flicker noise signal has the following PSD:

$$S(f) = \frac{K_f}{C_{ox}WL} \cdot \frac{1}{f} \quad \frac{[V^2]}{[Hz]}$$

With C_{ox} being the oxide capacitance, W and L the width and length of the channel. K_f is the flicker constant and depends on the specific application.

2.3 Circuit Implementation

At circuit level, the CT is implemented with the cascade of stages priorly described: modulation, noise-adding, amplification, demodulation and lowpass filter.
In detail, modulation implies a translation in the frequency domain from the base band to the translated band.
Noise-adding corresponds to the stage in which the noise signal is added to the circuit, taking into account both thermal noise and the contribution of flicker noise.
Amplification implies a scalar multiplication of the signal by a factor G, that is the gain of the amplifier. In the Matlab model, this stage is assumed to have an ideal flat frequency-response at the gain value, but it is possible to integrate its behavioral features in the program for a more accurate simulation.
Demodulation restores the tone corresponding to the original signal from the translated band to the base band and translates the flicker noise into a band centered in f_m. A constraint is that modulation and demodulation have to function in synchrony. In the Matlab model, this is granted by the definition of time/frequency domain vectors and the vectors representing the signals, which are in correspondence.
Finally, lowpass filter has the purpose of toning down the flicker noise, that in the prior stage was translated in higher frequencies than the one of the filter. One of the interesting reasons to work on the Matlab model is the possibility to rapidly explore different configurations (variating determined parameters) before further proceeding in the CP development [6].

Matlab Model

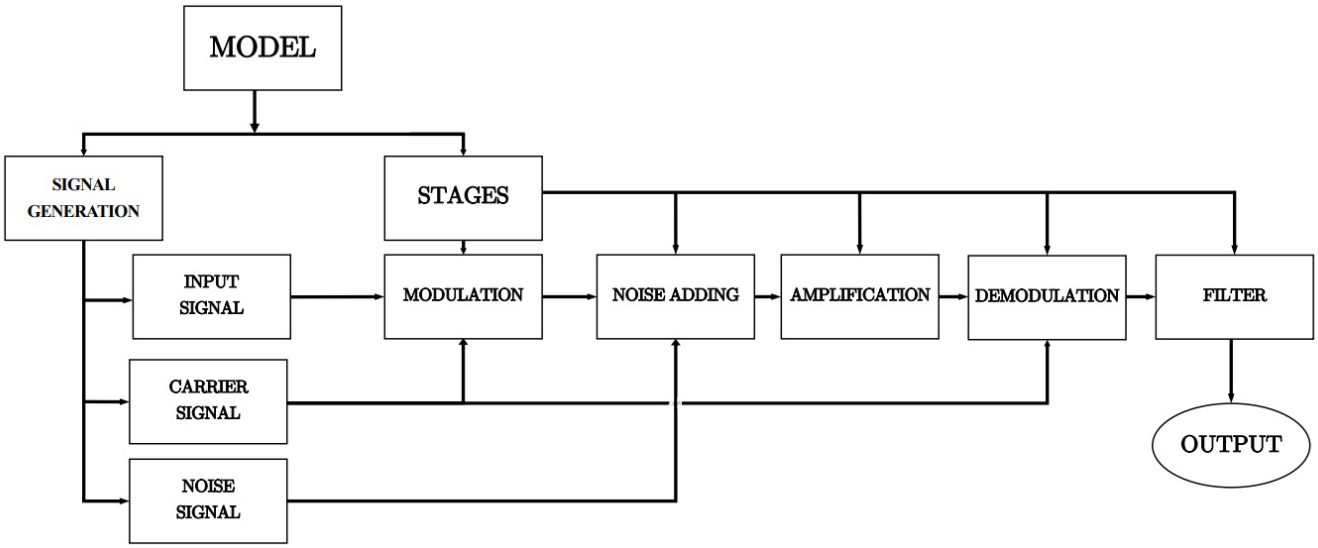

Figure 3.1: Schematic of the Matlab model.

The Matlab model is configured in a modular way, as shown in Figure 3.1. For each generated signal there is a corresponding function. The organization of the different parts is done in the main program.

- **"SineWaveGenerator.m"** generates a sine wave function that represents the input signal.
- **"ModulationSignal.m"** creates the signal corresponding to the carrier signal used for the modulation and demodulation stages.
- **"NoiseGenerator.m"** contains the generation of thermal noise and 1/f noise signals, adds them to generate the total noise vector and returns the total noise in its time domain representation.

3.1 Time and Frequency Domain Visualization

Signals in input/output (I/O) at each stage can be visualized in the time domain or frequency domain with 2-D line plots.

Passing from the time domain to the frequency domain representation of a signal is

possible with the *Fast Fourier Transform* (FFT) algorithm. In particular, a signal in the frequency domain can be visualized with the following instructions applied to the vector in the time domain. Subsequently, the code lines used to visualize a signal in the frequency domain (for a signal V_{n1_time}) are shown.

```
%% frequency domain representation
Vn1_freq = fft(Vn1_time);          % fft vector
P2a = abs(Vn1_freq/Ntot);          % double-sided spectrum
P1a = P2a(1:Ntot/2+1);             % single-sided spectrum
P1a(2:end-1) = 2*P1a(2:end-1);
```

In the frequency domain visualization, the plot is executed on the frequency vector and on the single-sided spectrum of the signal [7].

3.1.1 FFT algorithm

The signals elaborated in Matlab are discrete and made up of a finite number of data points. Therefore, instead of relying on the Fourier transform that can be performed on continuous time signals, the *discrete Fourier transform* (DFT) of the signal is needed. The spectrum of a DFT is also discrete and finite.

To compute the DFT of a vector, it is possible to use the *FFT algorithm*.
From a generic input sequence, this algorithm produces an output vector through matrix multiplication.

This solution has significant proportionalities that need to be taken into account when programming the model:

- $\Delta f \propto \frac{1}{N_{tot}}$: the frequency resolution is inversely proportional to the length of the signal. The longer the signal sampled, the more precise the frequency resolution.
- $\Delta t \propto \frac{1}{f_{max}}$: the distance between the time domain representation of the data points is inversely proportional to the max frequency distinguishible.

Specifically, the maximum frequency that can be computed in the spectrum is the *Nyquist frequency*, determined by the Nyquist-Shannon theorem.
By using the Matlab code to compute the FFT on the following line:

```
% Fast Fourier Transform
Y = fft(X);
```

it is implemented the operation:

$$Y(k) = \sum_{j=1}^{n} X(j) W_n^{(j-1)(k-1)}$$

Similarly, the inverse DFT is implemented with:

```
% Inverse Fast Fourier Transform
X = ifft(Y);
```

That represents:

$$X(j) = \frac{1}{n}\sum_{k=1}^{n} Y(k) W_n^{-(j-1)(k-1)}$$

[7]

3.1.2 Resolution Bandwidth

To maintain a correspondence between the time domain and the frequency domain, two vectors are used to describe these domains: t and f, respectively.
The vector representing the frequency domain ranges from the DC component to the Nyquist frequency, which is half the maximum acquisition frequency.
Considering that the *frequency resolution* is proportional to the length of the time domain vector, it is possible to select the number of sinusoidal periods of the input signal (which determines the length of all other signals) to configure the frequency resolution.

3.2 Signal Generation

To generate the different signals used in the simulation, there is a corresponding function, as described at the start of the chapter. Subsequently, the generation of each signal is examined in detail.

3.2.1 Input Signal

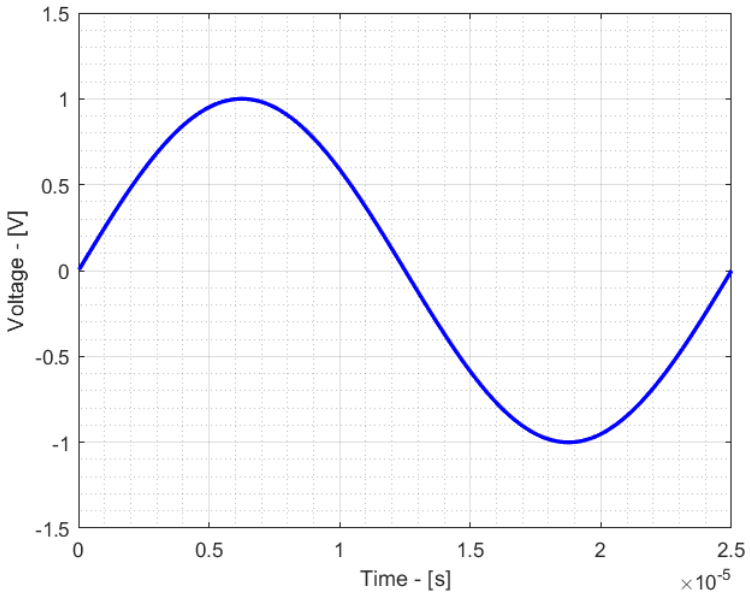

Figure 3.2: Time domain representation of the input signal vin. A single period of input sinusoid is considered.

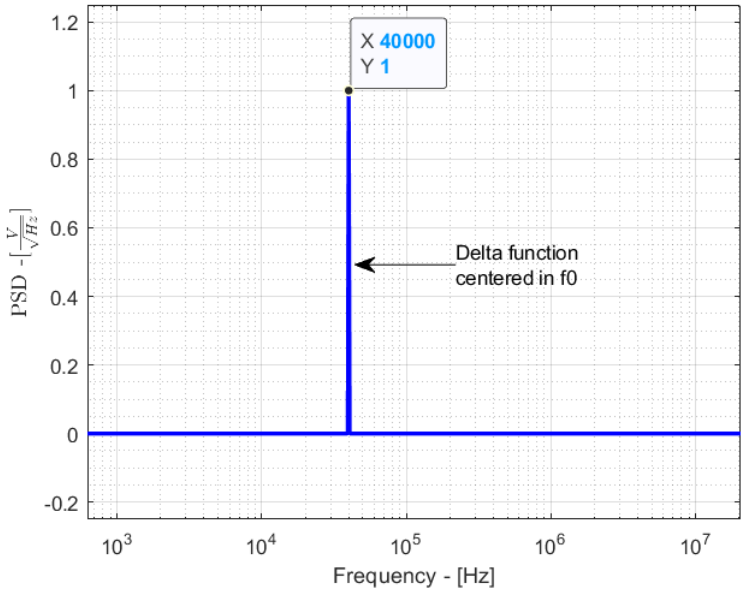

Figure 3.3: Frequency domain representation of the input signal vin. In this case, 64 periods of input signal are considered.

The input signal is the first input of the circuit, i.e. the signal produced from the piezoelectric transducer and passed to the LNA.
In particular, it is characterized by the following equation:

$$V_{in} = V_{0p} sin(2\pi f_0 t)$$

And it is implemented with the instruction:

```
% Input Signal
vin = V0p*sin(2*pi*f0*t);
```

With V_{0p} being the amplitude of the signal, f_0 the frequency of the signal, and t the time vector. In order to represent the input signal in the current application, f_0 is set to 40 kHz.

To test the performance of the circuit completely depending on noise, it is possible to null V_{0p} and therefore the entire input signal V_{in}. In this case the only stages the noise signal goes through are amplification, demodulation and lowpass filtering.

The input signal V_{in} can be observed in the time domain and in the frequency domain in Figure 3.2 and in Figure 3.3.
In the frequency domain, the single-sided spectrum of the input signal is a delta function centered on the frequency of the input sine wave $f_0 = 40\ kHz$.
In addition, for a 1 V amplitude V_{0p} of the input signal V_{in}, the amplitude of the signal tone in f_0 is 1 $\frac{V}{\sqrt{Hz}}$. To clearly observe the effects of the CT on both the signal and the noise, the amplitude of the input signal V_{0p} is set to 100 nV.

3.2.2 Carrier Signal

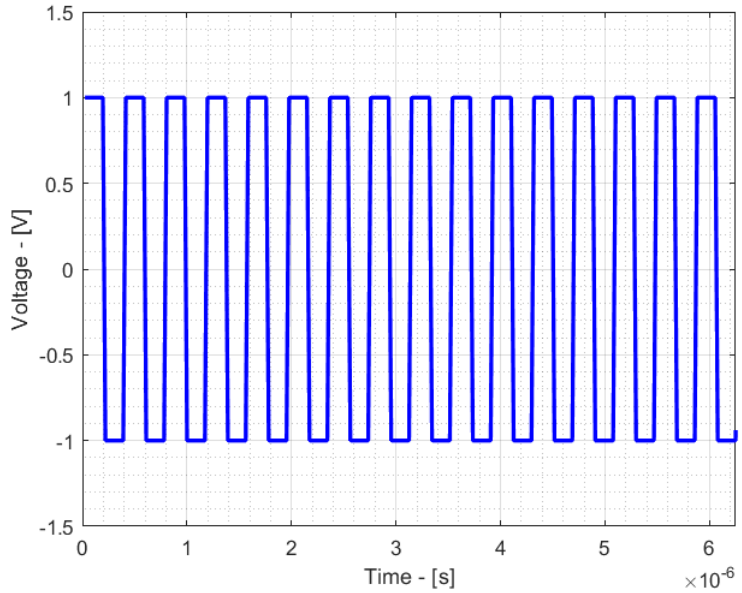

Figure 3.4: Time domain representation of the carrier signal. 16 periods of the carrier signal in the time domain are considered in the plot.

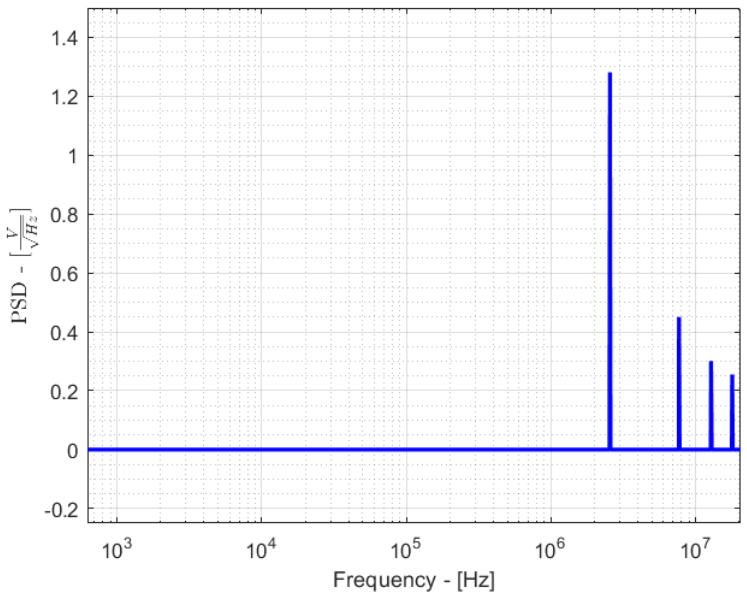

Figure 3.5: Frequency domain representation of the carrier signal. In this case, 64 periods of input signal are considered. It is possible to observe that the main tone in the spectrum is at f_m.

The choice of the carrier signal is a key part of the simulation. In fact, the frequency of the carrier signal, or the *modulation frequency* f_m, determines the translation of the band in the frequency domain. As a consequence, there is a constraint on the *cutoff frequency* of the filter f_{cut}, because it cannot be $f_m < f_{cut}$, or flicker noise would only be translated, without being toned down.

In addition, the choice of the carrier signal (and its f_m) implies an optimal order of the filter choice: fixed the cut-off frequency of the filter, it is possible to obtain the same flicker noise power attenuation even with a lower filter order by incrementing the modulation frequency f_m.

Hence, there are many interesting options for the carrier signal.
A first possibility is the impulse train, that is a periodic signal of alternating unitary values [+1, -1, +1, -1, ...]. At programming level, this signal has the highest possible frequency for a carrier wave (the Nyquist frequency).

In the model a square wave signal with $f_m = 2.56$ MHz is used. Specifically, a carrier signal made of a periodic vector of eight points of value +1 alternated with eight points of value -1 is selected, and it is implemented with the following Matlab instructions:

```
%%% Carrier Signal
vm = ones(1, length(t));
vm(9:16:end) = vm(9:16:end)-2;
vm(10:16:end) = vm(10:16:end)-2;
vm(11:16:end) = vm(11:16:end)-2;
vm(12:16:end) = vm(12:16:end)-2;
vm(13:16:end) = vm(13:16:end)-2;
vm(14:16:end) = vm(14:16:end)-2;
vm(15:16:end) = vm(15:16:end)-2;
vm(16:16:end) = vm(16:16:end)-2;
```

In an interval of the time domain and in the frequency domain, this signal can be observed in Figure 3.4 and in Figure 3.5.

The carrier signal has relevant consequences in the modulation stage. In fact, there are possibilities that imply a translation of the main tone of the input signal, which is the one from f_0 to $[f_m - f_0, f_m + f_0]$, but at the same time they imply a tone in f_0 (for example the choice of square waves with f_m significantly lower than the presented one).

3.2.3 Noise Signal

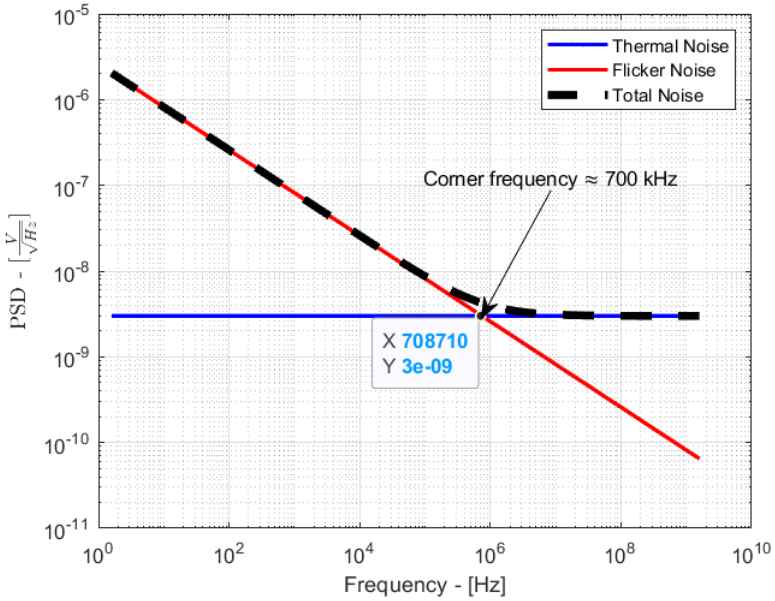

Figure 3.6: Ideal behaviour of noise signals simulated in the model. In particular thermal noise is represented in blue, flicker noise in red and total noise in black.

Table 3.1: Noise specifications.

Symbol	Parameter	Value
f_c	Corner frequency	$\approx$ 700 kHz
PSD_{Th}	Thermal noise PSD	3 $\frac{nV}{\sqrt{Hz}}$
K_{ftot}	Total flicker coefficient	$6.7 \cdot 10^{-11}$ V^2
W	Channel width	100 μm
L	Channel length	0.1 μm

The ideal signal of noise is the combination of 1/f and thermal contributions, which can be observed in Figure 3.6. Table 3.1 reports the specifications of the noise signal.

To simulate the noise, in the script "NoiseGenerator.m", the total noise vector is obtained by squared sum in the frequency domain of the thermal contribution V_{n1} and the flicker V_{n2}, as described in Section 2.1.

Thermal Noise Generation

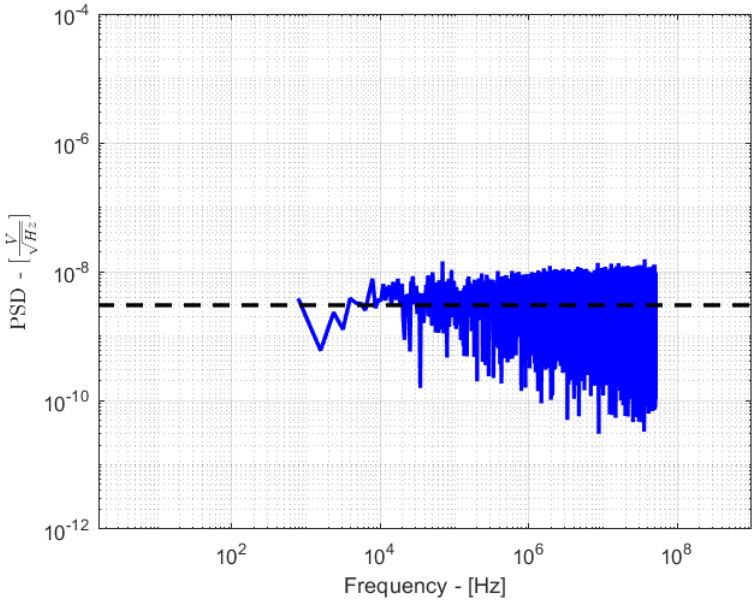

Figure 3.7: Superposition of the ideal thermal noise behavior and the generated thermal noise.

Thermal noise is generated through the following code:

```
%%% Thermal Noise generation
% Gaussian distributed vector
Vn1_time = randn(1,Ntot);
% Actual standard deviation
rmsgauss = std(Vn1_time);
% Normalization
Vn1_time = Vn1_time/rmsgauss;

Vn1_time = rms_th*Vn1_time/sqrt(rsbw);
```

White noise is constant over all the bandwidth in the frequency domain, as illustrated in Figure 3.7.
The noise PSD has a constant value of 3 $\frac{nV}{\sqrt{Hz}}$ and the shaping of the noise is linearly distributed over the frequency domain.

1/f Noise Generation

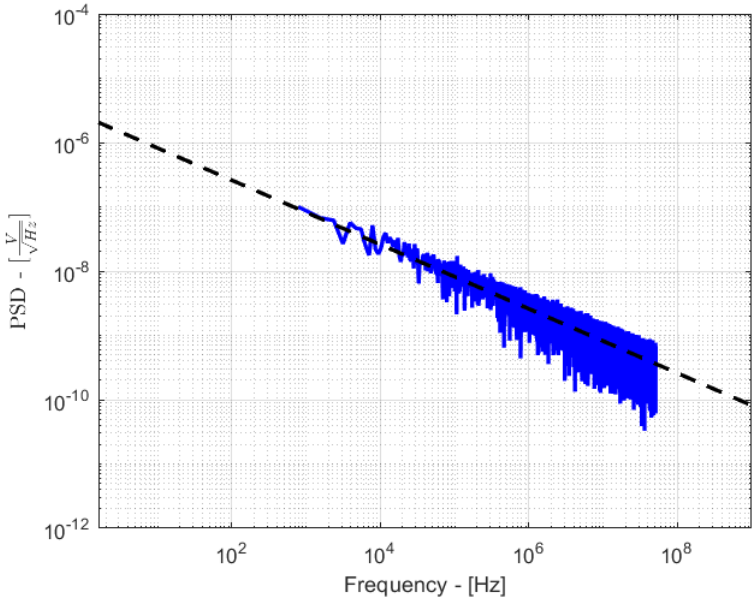

Figure 3.8: Superposition of the ideal 1/f noise behavior and the generated 1/f noise.

To generate a representation of the 1/f noise signal in the time domain, the following steps are applied:

1. Creation of a white noise vector in the time domain with the *randn()* function.
2. Application of the Fast Fourier Transform *fft()* to the vector to obtain the frequency domain representation.
3. Filter the vector to force the proportionality of PSD to $\sim 1/f$.
4. Application of the Inverse Fast Fourier Transform *ifft()* to the vector to obtain the representation in the time domain.

The instructions for implementing these operations are the following:

```
%%% 1/f Noise Generation
Cpox=15e-15/1e-12;
W=100e-6;
L=0.1e-6;

if rem(Ntot,2)
    M = Ntot+1;
else
    M = Ntot;
end

% White Noise Generation
x = randn(1, M);
```

```
% Frequency domain representation
X = fft(x);

% Filtering the signal to impress 1/f PSD proportionality
NumPts = M/2+1;
n = 1:NumPts;

X(1:NumPts) = X(1:NumPts)./n;
X(1:NumPts) = X(1:NumPts)*(1e-23)/(Cpox*W*L);
X = sqrt(X);
X(NumPts+1:end) = real(X(M/2:-1:2))-1i*imag(X(M/2:-1:2));

% Corner frequency setting
X = X*(scaling);

% IFFT
Vn2_time = ifft(X);
```

[8]

In the model, this noise signal is combined with thermal noise through squared sum in the frequency domain to obtain the resulting noise vector.
The scaling coefficient allows the configuration of the *corner frequency*, i.e. the frequency in which thermal noise and 1/f noise coincide. In the simulation, a corner frequency of $\approx 700\ kHz$ is set.
With these instructions, the result is shown in Figure 3.8.

Resulting Noise

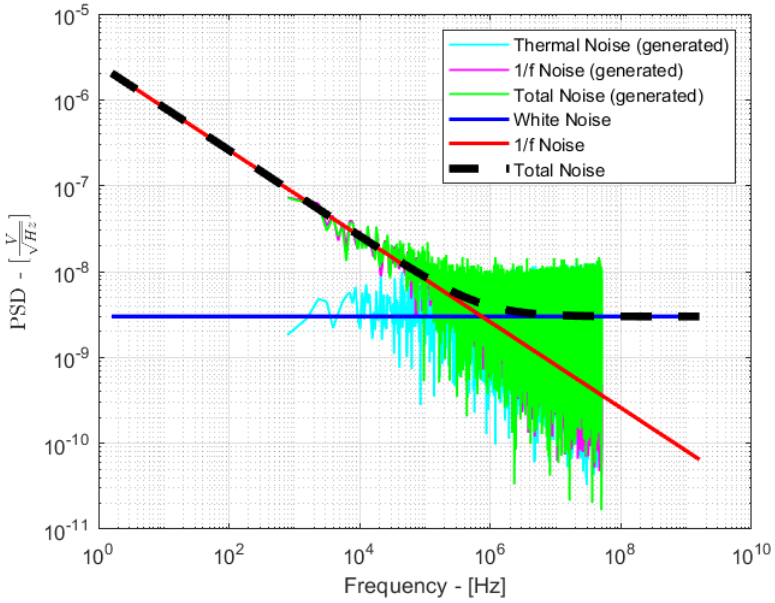

Figure 3.9: Superposition of the noise signals.

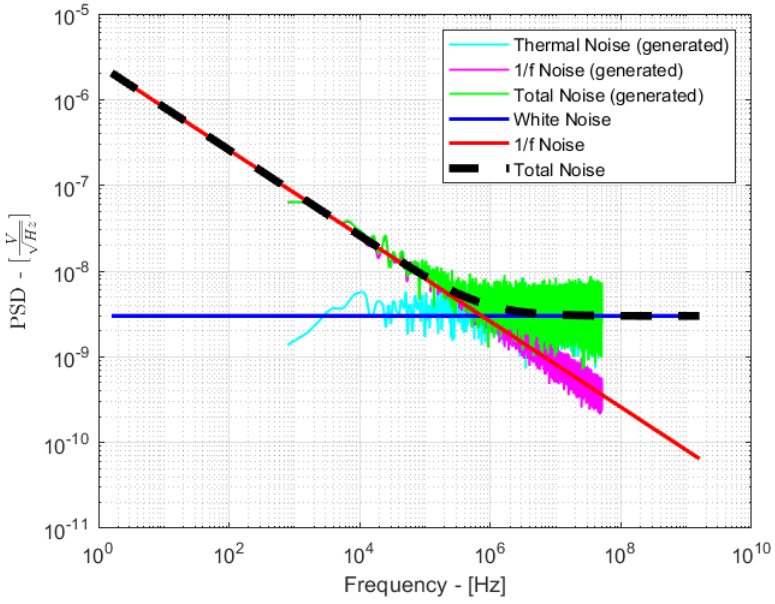

Figure 3.10: Superposition of the noise signals with a filter to check the shapings.

As a result, the noise signal obtained from the thermal contribution and 1/f noise contribution can be observed in Figure 3.9, in a superposition with the previously described signals. In addition, to have a better verification on the shapings of the signals, it is applied a check with a filter. This leads to the result shown in Figure 3.10.

3.3 Stages

In the main program, the cascade of stages in the CT is implemented through vector operations on the signals generated in the corresponding functions.

3.3.1 Modulation Stage

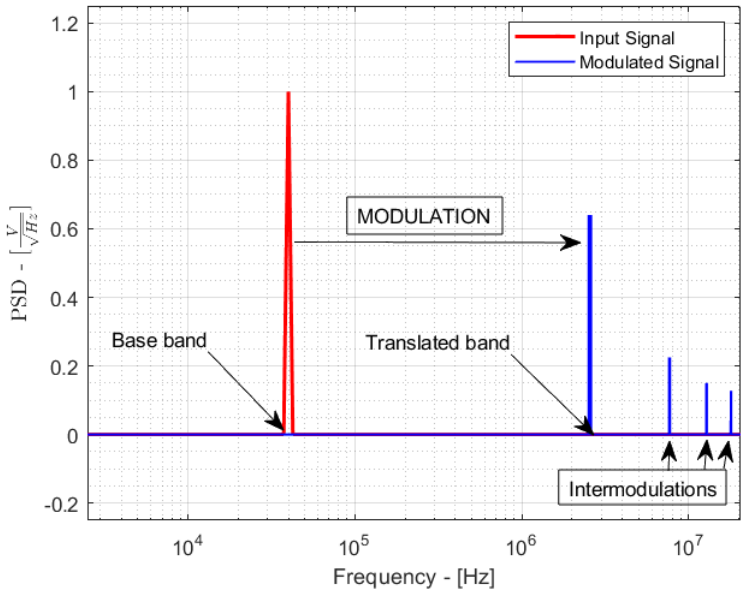

Figure 3.11: Superposition of the I/O signal at the modulation stage.

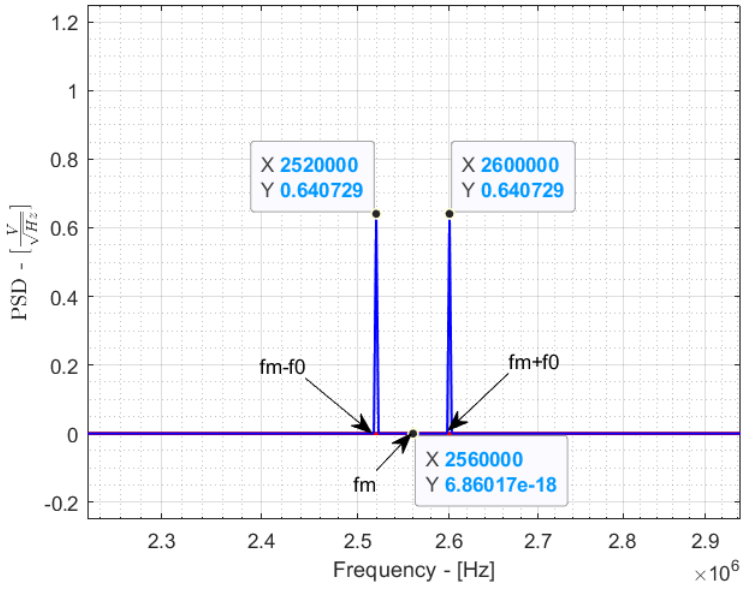

Figure 3.12: Zoom of the translated band.

The modulation stage applies only to the original signal V_{in} and it consists of the multiplication of the input signal V_{in} and the carrier signal V_m. This operation is

implemented through the following line of code:

```
%%% Modulation Stage
yin = vin.*vm;       % Modulated Signal
```

After this operation, the resulting vector is the modulated signal y_{in}. In the frequency domain, it is possible to observe the translation of the input signal V_{in} from the base band to the translated band in Figure 3.11.

The tone in f_0 (in this simulation: 40 kHz) corresponds to the FFT of the input signal and it is centered in the base band. In the same figure, it is possible to observe the modulated signal in the translated band, centered on the frequency of the carrier signal (in this simulation: 2.56 MHz).
Specifically, the main tones of the modulated signal are two: one in $f_m - f_0$ and one in $f_m + f_0$, as illustrated in Figure 3.12. At higher frequencies, it is possible to observe the phenomenon of *intermodulation*, which involves loss of information. Depending on the carrier signal choice, intermodulations can occur even at frequencies lower than $f_m - f_0$.

To check the coherence of this operation, it is possible to calculate the integral of the power signal on the entire bandwidth before and after modulation. The two results are expected to coincide. This verification can be implemented with the following instructions:

```
% Check
vinPow = trapz(P1vin.^2);
yinPow = trapz(P1yin.^2);

fprintf("Power of vin: "+vinPow+" and of yin: "+yinPow);
```

These instructions compute the power integral from the single-sided spectrums on all frequencies in the band, from DC to Nyquist frequency.
In this particular case, an amplitude of 1 V for V_{0p} has been used. Subsequently, to visualize the effects of the stages on the signal, it is used $V_{0p} = 100\ nV$.

3.3.2 Noise-Adding Stage

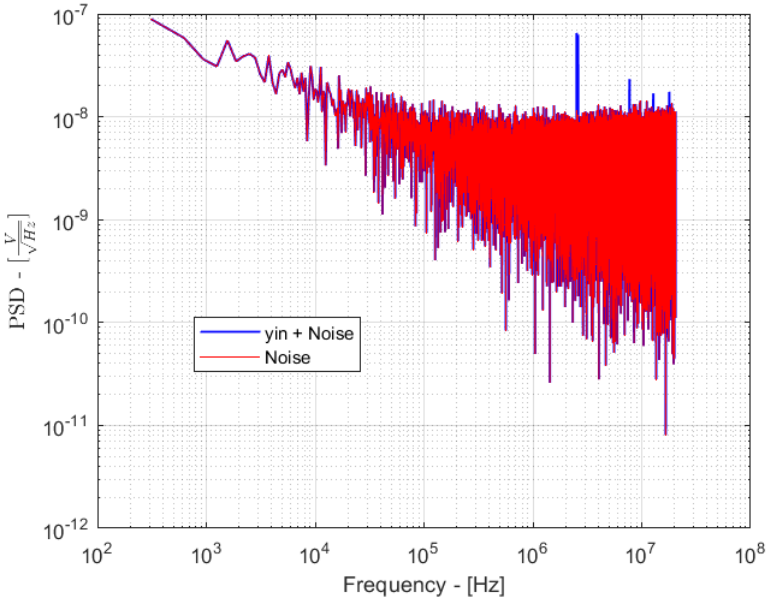

Figure 3.13: Modulated corrupted signal and noise signal superposition.

The noise signal, generated as described in Section 3.2.2 is added to the modulated signal y_{in} in the time domain with the instruction:

```
%%% Noise Addition
Sn = yin+Vn;        % modulated and corrupted signal
```

The superposition of the noise signal with the modulated signal corrupted from noise is illustrated in Figure 3.13.

3.3.3 Amplification Stage

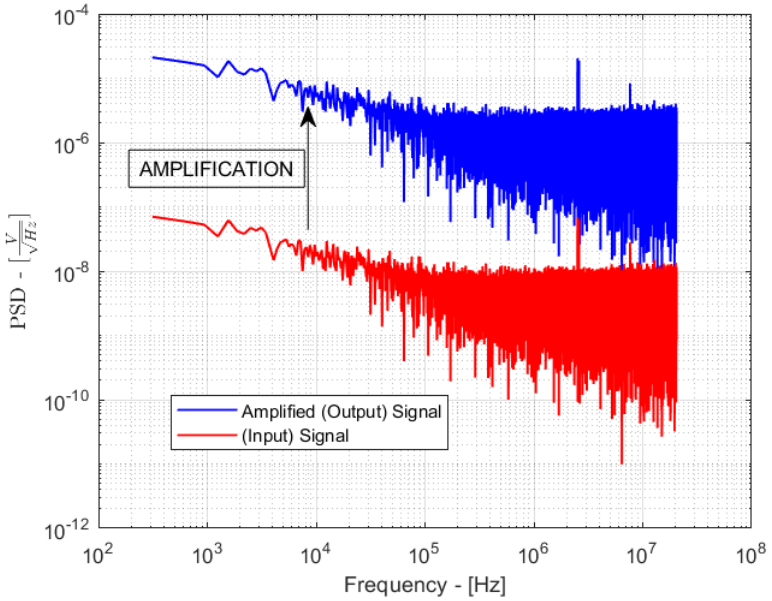

Figure 3.14: I/O signals at the amplification stage.

In the Matlab modelling of the CP it is assumed that the amplifier has a flat frequency-response at the value of the gain $G = 300$. Therefore, this stage is implemented in the following way:

```
%%% Amplification
Va = G*Sn;        % amplified signal
```

By the property of linearity of the Fourier transform, in the frequency domain, the signal shape is the same as before amplification, with the only difference of a factor G. This can be observed in Figure 3.14.

3.3.4 Demodulation Stage

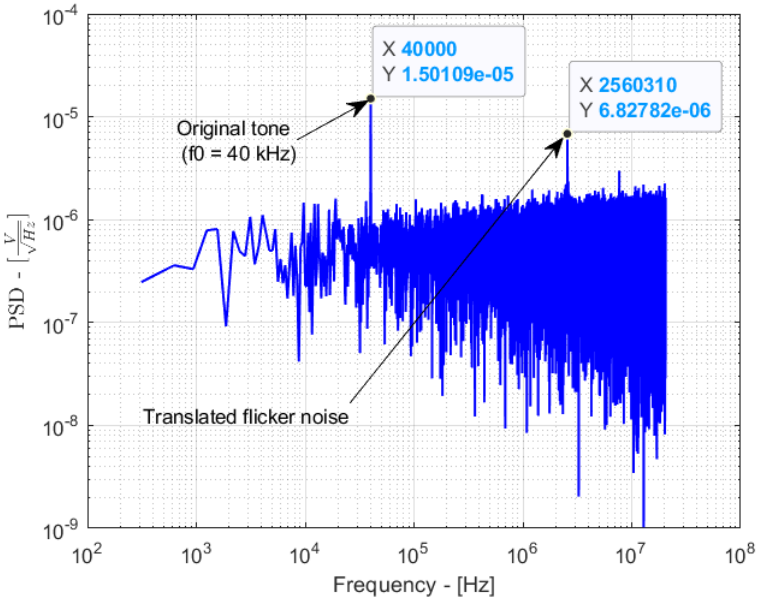

Figure 3.15: Frequency domain representation of the demodulated signal.

In the demodulation stage, the signal is multiplied with the same carrier signal that was used in the modulation stage. For this purpose it is used the following instruction:

```
%%% DEMODULATION
Vd = Va.*vm;          % demodulated signal
```

In the frequency domain, demodulation implies that the tone corresponding to the original signal V_{in} is translated to the base band, while the 1/f noise contribution is modulated in the translated band. This behavior is illustrated in Figure 3.15. The two frequency bands are switched and therefore it is possible to lowpass filter the signal in the next stage.

3.3.5 Lowpass Filter

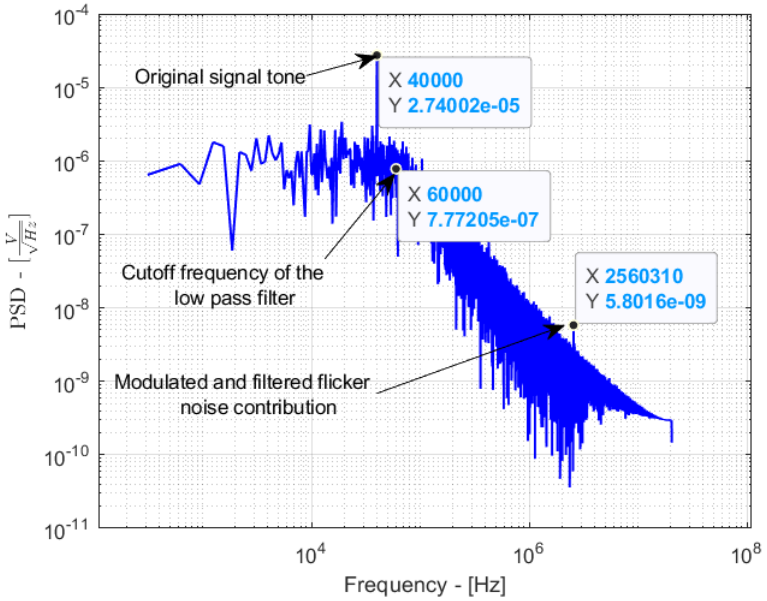

Figure 3.16: Frequency domain representation of the filtered signal.

To obtain the final output, it is necessary to tone down the 1/f noise contribution in the translated band. Hence, it is used a lowpass filter with a cutoff frequency f_{cut} lower than the modulation frequency f_m. In Matlab, the filter is an nth-order Butterworth filter implemented with:

```
%%% Lowpass filter stage (second order)
% Configure the cutoff frequency of the filter
f_cut = 60 kHz;

% Transfer function coefficients of the filter
[Num, Den] = butter(2, 2*(f_cut)/Fs);

% filtered signal : Sf
Sf_time = filter(Num, Den, Sd_time);
```

The effect of this filter on the demodulated signal can be observed in Figure 3.16. These instructions return the transfer function coefficients of the Butterworth filter, in this case of the second order.

3.4 Results

Table 3.2: Recap table of Input Referred Noise Power (IRNoP) and Output Referred Noise Power (ORNoP) RMS values integrated until $f_c = 60$ kHz, in case of only thermal noise simulation and only 1/f noise simulation. There is a comparison of CT implementation and simple amplification of the noise signal.

	Symbol	Parameter	Value
CT	$V_{RMS\_th\_in\_CT}$	Thermal IRNoP	0.93 μV_{RMS}
	$V_{RMS\_fl\_in\_CT}$	Flicker IRNoP	0.29 μV_{RMS}
	$V_{RMS\_th\_out\_CT}$	Thermal ORNoP	278.48 μV_{RMS}
	$V_{RMS\_fl\_out\_CT}$	Flicker ORNoP	87.87 μV_{RMS}
Amplification	$V_{RMS\_th\_in\_NoCT}$	Thermal IRNoP	1.02 μV_{RMS}
	$V_{RMS\_fl\_in\_NoCT}$	Flicker IRNoP	3.96 μV_{RMS}
	$V_{RMS\_th\_out\_NoCT}$	Thermal ORNoP	312.36 μV_{RMS}
	$V_{RMS\_fl\_out\_NoCT}$	Flicker ORNoP	1134.8 μV_{RMS}

Table 3.3: External parameters used in the simulation with only noise.

Symbol	Parameter	Value
f_c	Corner frequency	$\approx$ 700 kHz
V_{Thfreq}	Thermal noise PSD	3 $\frac{nV}{\sqrt{Hz}}$
K_{ftot}	Total flicker coefficient	$6.7 \cdot 10^{-11}$ V^2
W	Channel width	100 μm
L	Channel length	0.1 μm

Table 3.4: Internal parameters used in the simulation with only noise.

Parameter	Value
Points/period of vin	1024
Number of periods	128
Modulation frequency	2.56 MHz
Cutoff frequency	60 kHz
Filter order	2

Table 3.5: Total noise results with CT and without.

	Symbol	Parameter	Value
CT	$V_{RMS\_tot\_in\_CT}$	Total IRNoP	0.97 μV_{RMS}
	$V_{RMS\_tot\_out\_CT}$	Total ORNoP	290.94 μV_{RMS}
Amplification	$V_{RMS\_tot\_in\_NoCT}$	Total IRNoP	4.18 μV_{RMS}
	$V_{RMS\_tot\_out\_NoCT}$	Total ORNoP	1254.08 μV_{RMS}

The simulation, with the set of parameters resumed in Tables 3.3 and 3.4, leads to the RMS values shown in Table 3.5, where the CT output with total noise is compared to the case of amplification only. In particular, the signal processed by the CP exhibits an output referred noise power of 0.29 mV_{RMS}, instead of 1.25 mV_{RMS} with exclusively the amplification stage, which is a factor 4.3 in noise power attenuation. The same comparison is shown in Table 3.2 for only thermal noise and only flicker noise simulations.

Furthermore, in this section two checks on the output are analyzed to verify the coherence of the Matlab model. In particular, the tests involve the RMS values computation and the graphical superposition of the signals.

The first check consists in the calculation of the RMS values in all the possible cases for a simulation. In particular, there are two expected results:

1. The RMS value with the application of the CT is expected to approximate the RMS value of the white noise multiplied by a factor G in the band from the DC frequency to the cutoff frequency of the filter.

$$\sqrt{\int_0^{f_{cut}} S_{CT}(f)df} \approx G \cdot \sqrt{\int_0^{f_{cut}} S_{Thermal\ Noise}(f)df}$$

   This expectation is legitimate in case of simulation without null input signal (only noise).

2. The RMS value without the application of the CT, that is, the signal that only crosses the amplification and lowpass filter stages, is expected to approximate the RMS value of the total noise multiplied by a factor G in the filter band.

$$\sqrt{\int_0^{f_{cut}} S_{No\ CT}(f)df} \approx G \cdot \sqrt{\int_0^{f_{cut}} S_{Total\ Noise}(f)df}$$

   Similarly to the first expected result, this expectation considers a simulation without input signal.

The second check is the graphical superposition of the output signals. In detail, it is expected that by multiplying the thermal noise PSD in $V/\sqrt{Hz}$ by G, it approximates the CT output value for $f < f_c$, and that by multiplying the total noise PSD in $V/\sqrt{Hz}$ by G, it approximates the output without CT for $f < f_c$.

3.4.1 RMS values

The first verification consists in the computation of the RMS values, calculated until the cutoff frequency of the filter.
For the CT output and the equivalent white noise (RMS value of the white noise multiplied by a factor G), the instructions used are the following:

```
%%% RMS VALUES COMPUTATION
% CT output
P1fF = P1f(1:IndecFcut);
PowerCT = trapz(rsbw, P1fF.^2);

% Equivalent white noise output
PowerWn = (Wn*G)^2 *bwth;

fprintf("\nThermal noise: "+sqrt(PowerWn)*1e6+" micro-
   Vrms");
fprintf("\nCT : "+sqrt(PowerCT)*1e6+" micro-Vrms\n");
```

This lead to 220.45 μV_{RMS} in the case of amplified white noise and 290.94 μV_{RMS} in case of CT output. The difference in the results is implied by the choice of the corner frequency (700 kHz) and the filter cutoff frequency (60 kHz), allowing the model to reduce the effect of the flicker noise power, but not completely remove it.

Similarly, the second expected result is the RMS value of the equivalent total noise, multiplied by G and calculated until the cutoff frequency, and the RMS value without Chopper, which is simply the combination of the amplification stage and the lowpass filter.
To compute those integrals, the following instructions are used:

```
%%% RMS VALUES COMPUTATION
% total noise
P1cF = P1c(1:IndexFcut);
PowerTn = trapz(rsbw, (G*P1cF).^2);

% No CT output
P1jF = P1j(1:IndexFcut);
PowerNoCT = trapz(rsbw, P1jF.^2);

fprintf("\nTotal noise: "+sqrt(PowerTn)*1e6+" micro-Vrms
   ");
fprintf("\nNo CT : "+sqrt(PowerNoCT)*1e6+" micro-Vrms\n")
   ;
```

This code leads to the result 1254.08 μV_{RMS} ORNoP for equivalent total noise and 1109.23 μV_{RMS} ORNoP without CT application.

3.4.2 Graphical superposition of the signals

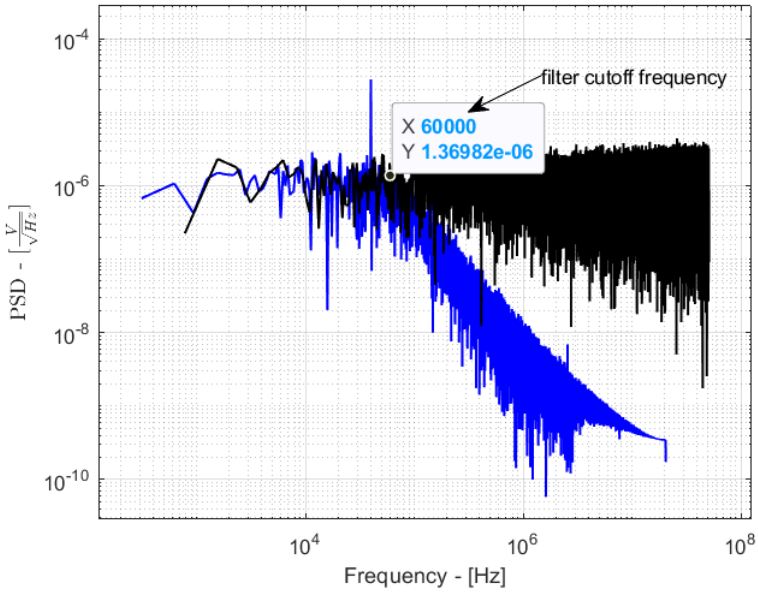

Figure 3.17: Superposition of CT output and amplified thermal noise signal.

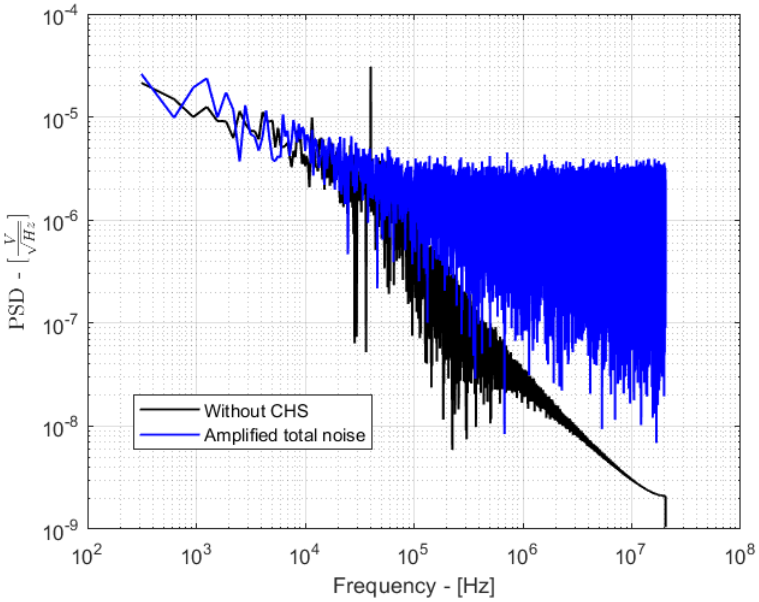

Figure 3.18: Superposition of resulting amplified noise and the output signal in case of amplification and filtering.

The last check is to verify the behavior of the output vectors. In detail, the expectation is that:

1. the CT output approximates the shape of the white noise vector amplified by G for $f < f_{cut}$, as illustrated in Figure 3.17.

2. the output vector without CT approximates the total noise vector amplified by G for $f < f_{cut}$, as shown in Figure 3.18.

It is needed to consider that in these simulation, the tone in $f = 40\ kHz$ corresponds to the input signal.
Additionally, if the corner frequency f_c of the noise signal is lower than the cutoff frequency of the filter f_{cut} it is expected another graphical superposition in case of CT implementation and in case of mere amplification: the expectation is that the two signals differ for $f < f_c$ and are approximately coincident for $f > f_c$. This is not the case of the last simulation, because the corner frequency f_c is 700 kHz, while the cutoff frequency of the filter f_{cut} is 60 kHz.

Conclusions

4.1 Results

The Matlab model of the CP is entirely customizable, which means that all the parameters relative to input signal, modulation, noise-adding, amplification, demodulation and lowpass filter can be adapted to different applications. This parameterization allows to study an optimal configuration for the specific case.
With the set of parameters specified for the simulation with only noise, the signal processed by the CP exhibits an output referred noise power of 0.29 mV_{RMS}, instead of 1.25 mV_{RMS} in case of mere amplification, that is approximately a factor 4 in the noise power attenuation. To obtain the same result via average of the signals, 16 acquisitions are necessary, instead of 1. Clinically, the problem is that 16 acquisitions imply 160 mGy dose, instead of 10 mGy to irradiate the patient and to obtain the same result.
In conclusion, the CT is an analog signal processing method to reduce the impact of noise sources (in particular at low frequencies). Hence, it can be integrated with digital techniques, or other methods in general, to improve the signal conditioning performance.

4.2 Future Perspectives

Using the Matlab model, another interesting elaboration on the CT is to study how to lower power consumption once a certain value of noise power reduction value is fixed, for instance $\alpha_N = \frac{V_{RMS\_CT}}{V_{RMS\_NoCT}}$ choosen to be a certain value.
Having this contraint, there are many possibilities for the CT parameters (the frequency of modulation f_m, the cutoff frequency of the filter f_{cut} and the filter order) to obtain α_N as a result, and each of these possibilities implies a certain power consumption. Therefore, it is possible to study the different cases to determine the configuration to obtain α_N with the minimum power consumption.
The importance in this direction of work are the consequences in terms of low-power consumption systems, such as, for example, in the medical perspective.

Bibliography

[1] E. A. Vallicelli, A. Baschirotto, S. Lehrack, W. Assmann, K. Parodi, S. Viola, G. Riccobene, and M. De Matteis, "22 db signal-to-noise ratio real-time proton sound detector for experimental beam range verification," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 1, pp. 3–13, 2021.

[2] M. Riva, E. A. Vallicelli, A. Baschirotto, and M. De Matteis, "Acoustic analog front end for proton range detection in hadron therapy," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 4, pp. 954–962, 2018.

[3] C. Enz and G. Temes, "Circuit techniques for reducing the effects of op-amp imperfections: autozeroing, correlated double sampling, and chopper stabilization," *Proceedings of the IEEE*, vol. 84, no. 11, pp. 1584–1614, 1996.

[4] M. Sampietro. Amplif. lineari a singolo transistore. [Online]. Available: https://sampietro.faculty.polimi.it/didattica/CAP8%20-%20Rumore.pdf

[5] M. Keshner, "1/f noise," *Proceedings of the IEEE*, vol. 70, no. 3, pp. 212–218, 1982.

[6] A. H. Said. Design of a chopper amplifier for use in biomedical signal acquisition. [Online]. Available: https://www.siue.edu/~gengel/BCIproject/KaderThesisFinal.pdf

[7] Mathworks. Fast fourier transform - matlab. [Online]. Available: https://it.mathworks.com/help/matlab/ref/fft.html

[8] M. E. H. Zhivomirov. Pink noise generation with matlab implementation. [Online]. Available: https://github.com/cortex-lab/MATLAB-tools/blob/master/pinknoise.m