MODULADORES SIGMA-DELTA

Electrónica III

Filipe Perestrelo nº39656

Maio de 2016

Índice

INTRODUÇÃO

Este trabalho teve como objectivo fazer o *design* e simular um conversor analógico-digital (ADC) para sinais, cuja banda vai até 16kHz. Começou-se por construir um modulador sigma-delta de segunda ordem e de seguida procedeu-se aos cálculos.

Foi pedido para considerar que as amostras de saída eram emitidas a uma frequência de 48kHz e que, para um sinal de amplitude máxima, a sua THD (*Total Harmonic Distortion*) seria de -76dB, sendo que a mesma (amplitude máxima) seria de 0.9Vrms. Foi também pedido para considerar que a SNR (*Signal to Noise Ratio*) mínima seria de 8dB para um sinal cuja amplitude mínima seria de 0.08mVrms.

Para realizar este trabalho, foi necessário recorrer a ferramentas, como o programa *MatLab*, maioritariamente para efectuar cálculos e apresentar gráficos, e o programa *LTSpice*, maioritariamente para desenho de circuitos e simulação dos mesmos.

Foi também pedido, como tarefa opcional, para realizar o mesmo, mas para um modulador sigma-delta com a arquitectura MASH (2+1).

Este trabalho pôde ser dividido em cinco partes. Na primeira parte, consolidaram-se as especificações do ADC, realizando cálculos e fazendo esboços do que se iria esperar. Na segunda parte, tratou-se do modulador, arranjando as expressões matemáticas que permitiram a observação do sinal dentro do mesmo (modulador). Na terceira parte, introduziu-se um filtro decimador e observou-se o efeito que este tinha no SNR do sinal. Na quarta parte, fez-se simulações de modo a variar as amplitudes do sinal, bem como a saturação dos integradores dentro do modulador, utilizando a ferramenta *MatLab*. Na quinta parte, simulou-se o circuito no programa *LTSpice* e observou-se os gráficos obtidos do sinal em determinados pontos do circuito. Por fim, foi-nos pedido para realizar todas as outras partes para o modulador sigma-delta com a arquitectura MASH (2+1).

CONSOLIDAÇÃO DAS ESPECIFICAÇÕES

Esta parte começou pela análise do esquema em blocos do ADC (figura 1) e fez-se o esboço do SINAD (*Signal-to-Noise And Distortion ratio*) que neste caso correspondia ao SNDR (figura 2) visto não haver distorção para amplitudes muito pequenas.

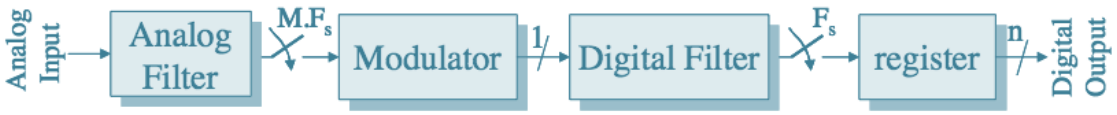

Figura 1 – Esquema em blocos do ADC

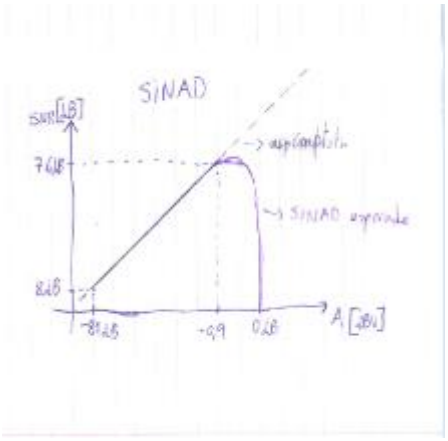

Figura 2 - Esboço do SINAD esperado

Como se pode ver pela SINAD, o ponto que corresponde ao pior caso, é o ponto onde pode haver distorção harmónica, ou seja será quando a amplitude do sinal de entrada for tão pequena que este se confundirá com o ruído térmico. Neste caso o ponto onde isso ocorre será com uma amplitude de 0.08×10^{-3}Vrms que corresponde a aproximadamente -81dB. Nesse ponto temos um SNDR/SINAD de 8dB como seria de esperar. É de notar que a SINAD é aproximadamente uma recta, porque estamos a desprezar a distorção, pois estamos dentro da banda de funcionamento do ADC.

Assumiu-se que o modulador começava a saturar para sinais 5 dB abaixo da tensão de referência. Com esta informação foi possível calcular essa tensão, através da equação 1:

Equação 1

$$SNR = 20 \times \log_{10}\left(\frac{Vamp}{Vref}\right)$$

Sendo Vamp=1V para garantir uma margem mínima e SNR=-5dB chegou-se à equação 2:

Equação 2

$$Vref = 10^{\frac{5}{20}}$$

Concluiu-se que Vref=1.7783 Vrms. Esse valor foi transformado em V (Volt) através da equação 3:

Equação 3

$$V = \sqrt{2} \times Vrms$$

Obteve-se o valor de Vref de 2.5149V.

De seguida calculou-se o máximo de ruído permitido na saída deste ADC para se obter aquele esboço de SNDR/SINAD. Neste ponto considerou-se o pior caso, ou seja, SNDR=SNRmin=8dB, que correspondia a uma amplitude de 0.08×10^{-3}V para Vref. Calculou-se então a potência de sinal para esse ponto através da equação 4:

Equação 4

$$Ps = Vref^2$$

Como este valor estava em $Vrms^2$, foi necessário convertê-lo para dB, logo utilizou-se a equação 5 para tal:

Equação 5

$$Ps[dB] = 10 \times \log_{10}(Ps)$$

Obteve-se então o valor da potência de sinal (em dB) de -81.9382dB.

A partir deste valor, foi possível calcular a potência de ruído total pela equação 6:

Equação 6

$$PNT[dB] = SNDR[dB] - Ps[dB]$$

Tendo este valor, obteve-se o valore W (Watt) da potência de ruído total através da equação 7:

Equação 7

$$PNT = 10^{\frac{PNT[dB]}{10}}$$

O valor foi de $1x10^{-9}W$.

Assumiu-se ainda que a potência do erro térmico (Ptn) era três vezes maior do que a potência do erro de quantização (Pqn). Assim, tendo já calculado a potência total e sabendo através da equação 8 que a potência do ruído total é igual à soma das potências do erro térmico e do erro de quantização, pôde-se calcular cada um dos ruídos:

Equação 8

$$PNT = Ptn + Pqn$$

O valor obtido para Ptn foi de $7.6x10^{-10}Vrms$ e para Pqn obteve-se o valor de $2.5\ x10^{-10}Vrms$. Para converter estes valores para decibéis, foi necessário a utilização da equação 9:

Equação 9

$$dBV = 20 \times \log_{10}\left(\sqrt{P}\right)$$

Substituindo na expressão P primeiro por Ptn e depois por Pqn, obteve-se os valores -91dB e -96dB respectivamente.

SIMULAÇÃO NO *MATLAB*

Nesta parte, estudou-se o comportamento do modulador de segunda ordem. Desta feita, foi necessário "desenhar" o modulador em diagrama de blocos (figura 3) e fazer os cálculos necessários para demonstrar qual a sua função de transferência de sinal (*STF(z)*) e a sua função de transferência de ruído(*NTF(z)*).

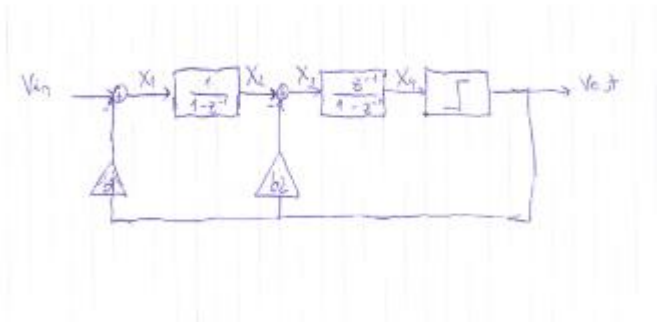

Figura 3 - Modulador de segunda ordem

Como se pode ver pela imagem foram definidos quatro pontos para o cálculo de expressões que mais tarde simplificariam o cálculo de de NTF(z) e de STF(z).

Em X1 (entrada do primeiro integrador) obteve-se a expressão 10:

Equação 10

$$X1 = Vin - Vout \times b1$$

Em X2 (saída do primeiro integrador) obteve-se a expressão 11:

Equação 11

$$X2 = \frac{X1}{1 - z^{-1}} = \frac{Vin - Vout \times b1}{1 - z^{-1}}$$

Em X3 (entrada do segundo integrador) obteve-se a expressão 12:

Equação 12

$$X3 = X2 - Vout \times b2 = \frac{Vin - Vout \times b1}{1 - z^{-1}} - Vout \times b2$$

Em X4 (saída do segundo integrador) obteve-se a expressão 13:

Equação 13

$$X4 = X3 \times \frac{z^{-1}}{1 - z^{-1}} = \left[\frac{Vin - Vout \times b1}{1 - z^{-1}} - Vout \times b2\right] \times \frac{z^{-1}}{1 - z^{-1}}$$

Finalmente, em Vout obteve-se a expressão 14:

$$Vout = X4 + V_Q = \left[\frac{Vin - Vout \times b1}{1 - z^{-1}} - Vout \times b2\right] \times \frac{z^{-1}}{1 - z^{-1}} + V_Q \Leftrightarrow$$

$$\text{Com: } b1 = b2 = 1, \text{vem}$$

Equação 14

$$\Leftrightarrow Vout = Vin \times z^{-1} + V_Q(1 - z^{-1})^2$$

Depois de se ter obtido a expressão de Vout, procedeu-se ao cálculo das expressões da NTF(z), com Vin=0, que se encontra na equação 15 e da STF(z), com V_Q=0, que se encontra na equação 16:

Equação 15

$$NTF(z) = (1 - z^{-1})^2$$

Equação 16

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

De seguida procedeu-se ao cálculo da expressão da potência do ruído de quantização na saída do modulador, assumindo que a saída era filtrada por um filtro passa-baixo ideal de frequência de corte igual a B. Para isso recorreu-se à expressão 17:

Equação 17

$$V_{QN}{}^2_{rms} = \int_{-B}^{B} S_{VQ}(f) \times \left|NTF(e^{\frac{j2\pi f}{Fs}})\right|^2 df$$

Sabe-se que S_{VQ} (f) é a potência de sinal dada pela expressão 18:

Equação 18

$$S_{VQ} = \frac{Vlsb^2}{12 \times Fs}$$

Sabe-se ainda que OSR (*Over Sampling Ratio*) é dado pela expressão 19:

Equação 19

$$OSR = \frac{Fs}{2 \times B}$$

Assim, a expressão 17 foi resolvida, tornando-se numa expressão muito mais simples:

$$V_{QN}{}^{2}_{rms} = \int_{-B}^{B} \frac{Vlsb^2}{12 \times Fs} \times \left|(1 - e^{\frac{j2\pi f}{Fs}})^2\right|^2 df \approx \frac{Vlsb^2}{12 \times Fs} \times \int_{-B}^{B} \times \left|\frac{-j2\pi f^2}{Fs}\right|^2 df$$
$$\approx \frac{Vlsb^2 \times 2^5 \times \pi^4 \times B^5}{12 \times Fs^5}, \text{pela expressão19, vem}$$

Equação 20

$$V_{QN}{}^{2}_{rms} = \frac{Vlsb^2 \times \pi^4}{12} \times \left(\frac{1}{OSR}\right)^5$$

Assim, pela expressão 20, foi possível saber qual o OSR mínimo para que o modulador tivesse um erro de quantização (Pnq) menor do que o tinha sido calculado anteriormente, isto é, $Vnq^2<2.5 \times 10^{-10}$Vrms. O valore calculado foi OSR=96.1074. Este valor foi arredondado para a potência de 2 mais próxima, ou seja, $2^n>96.1074$ e portanto, $2^n=128$.

Para calcular o número de bits da saída digital de modo a ter uma potência de ruído de quantização 10dB menor do que a potência de ruído total, utilizou-se a equação 21:

Equação 21

$$V_{QN}{}^{2}_{rms} = \frac{Vref^2}{2^n}$$

Assim, desenvolvendo a equação 21, chegou-se à expressão 22:

$$n = \frac{1}{2} \times \log_2\left(\frac{Vref^2}{V_{QN}{}^{2}_{rms}}\right)$$

Obteve-se então, o valor para n de 17 bit.

Simulando no *MatLab*, obteve-se os gráficos do comportamento, do modulador. Nesta simulação obteve-se o gráfico do comportamento do filtro decimador de terceira ordem (figura 4), o gráfico das transformadas de *Fourier* do filtro *sync* (figura 5), o gráfico das transformadas de *Fourier* do sinal de entrada e do sinal de saída (figura 6) e o gráfico da saída decimada (figura 7).Com as especificações calculadas anteriormente, calculou-se o SNDR, tendo-se obtido o valor de 86dB, para um sinal de amplitude 1.2728V.

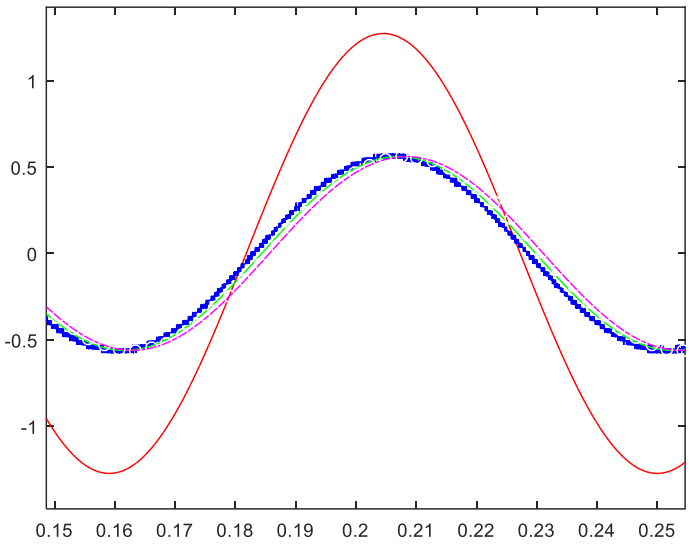

Figura 4 – Sinal de entrada (vermelho), saída do primeiro integrador (azul), saída do segundo integrador (verde) e sinal de saída (magenta)

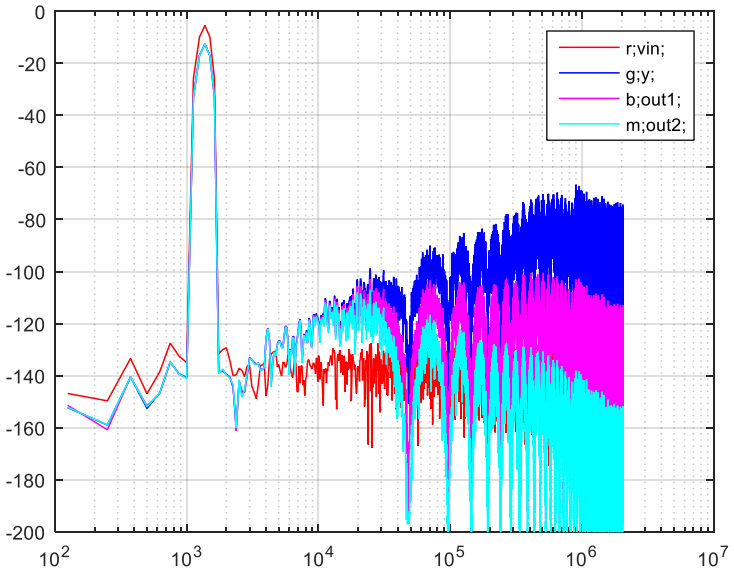

Figura 5 – Transformadas de Fourier do sinal de entrada (vermelho), da saída do primeiro integrador (azul escuro), da saída do segundo integrador (magenta) e do do sinal de saída (azul cyan)

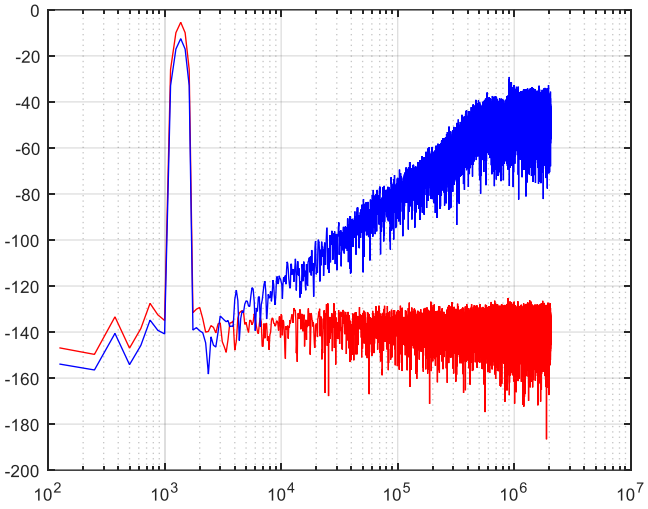

Figura 6 – Transformadas de Fourier do sinal de entrada (vermelho) e da saída decimada (azul)

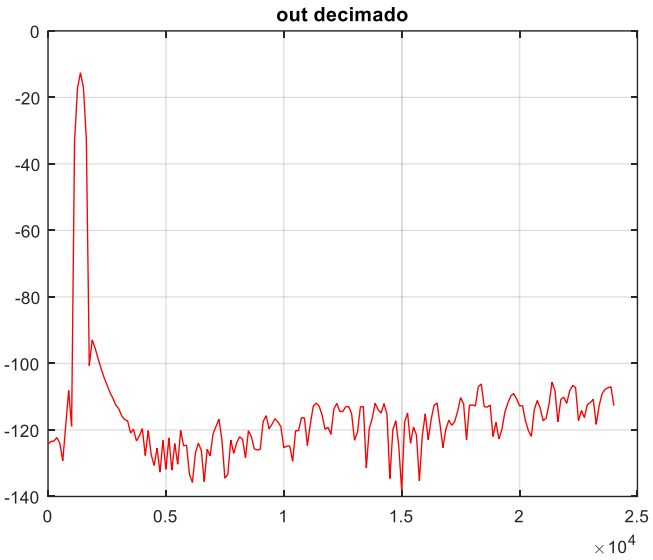

Figura 7 – Saída decimada

Pelos gráficos pode-se concluir que as especificações foram verificadas. O SNDR deu algo próximo do que se esperava, visto que a SINAD só se começa a degradar depois dos 1dB, logo o sinal ainda consegue aumentar o seu SNDR.

Através da simulação pelo MatLab, colocou-se a amplitude do sinal a variar entre 0.08×10^{-3}V e 2xVref, de modo a que se pudesse ver bem a acentuação da SINAD. Conseguiu-se obter a SNDR à saída do filtro decimador (figura 8).

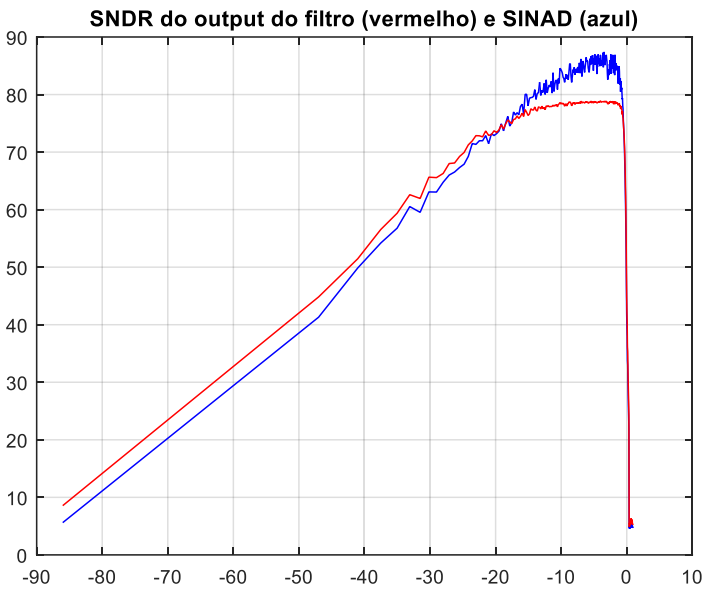

Figura 8 – SNDR do sinal de entrada (azul) e SNDR do sinal de saída do filtro

Nota-se claramente no gráfico, que a saída do filtro, ao contrário do SNDR do sinal, satura antes de chegar a 80dB, tendo como SNDR máximo um valor próximo de 76dB (neste caso de 78.90dB) como seria de esperar.

Este filtro produz uma amostra para cada N amostras que estiverem presentes na entrada. Consegue também atenuar o ruído de quantização do modulador presente nas altas frequências. Consegue também "empurrar" o ruído presente no sinal para as altas frequências.

Fez-se a simulação do ADC. Começou por se verificar a SNDR do sinal de entrada apenas (figura 9).

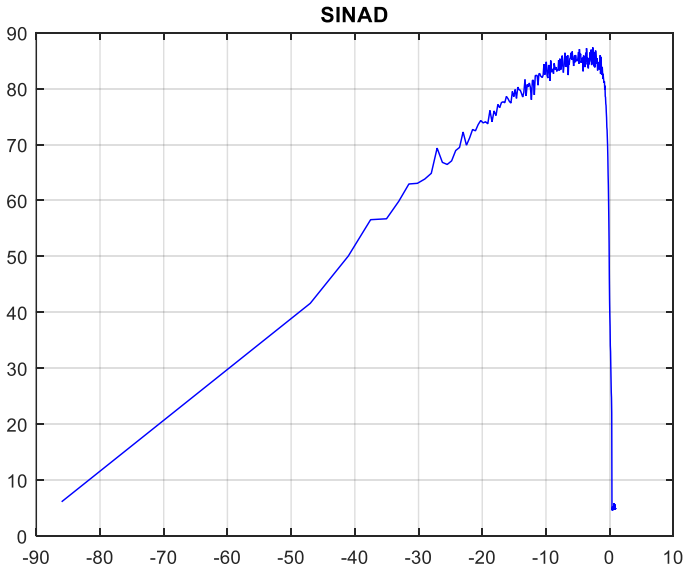

Figura 9 – SNDR do sinal de entrada

Como já foi referido anteriormente, a SNDR do sinal de entrada, realmente satisfaz os requisitos do sistema. A frequência do sinal de entrada situa-se perto dos 1kHz (1.375kHz).

De seguida introduziu-se distorção na entrada de cada integrador e na saída do sinal, de modo a que esta "eliminasse" os sinais com amplitude superior 0.9Vrms. O resultado encontra-se na figura 10.

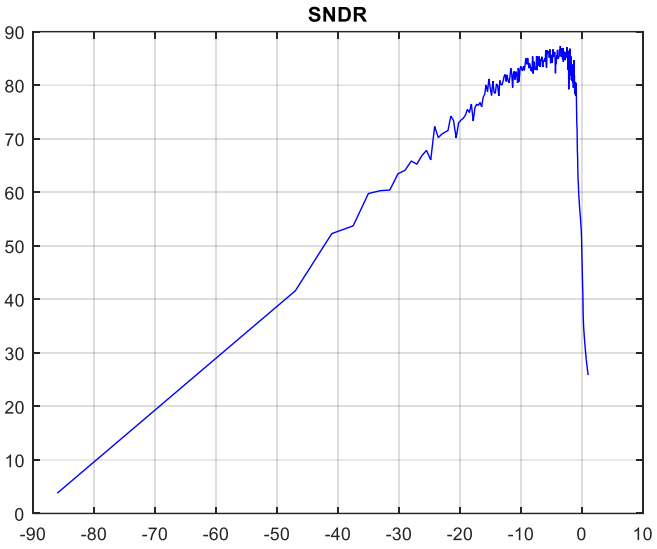

Figura 10 – SNDR do sinal de saída saturado

Embora sejam gráficos muito semelhantes, é de notar que neste último, a SNDR começa a "cair mais cedo" do que no anterior. Isto acontece, porque a

saturação à entrada de cada integrador, faz limitar o sinal, denegrindo a sua SNDR.

Os códigos em *MatLab*, encontram-se nos anexos.

SIMULAÇÃO NO *LTSPICE*

Para teste deste circuito, foi necessário a utilização da ferramenta *LTSpice*. Através deste programa, começou-se por desenhar o esquemático do circuito a implementar (figura 11).

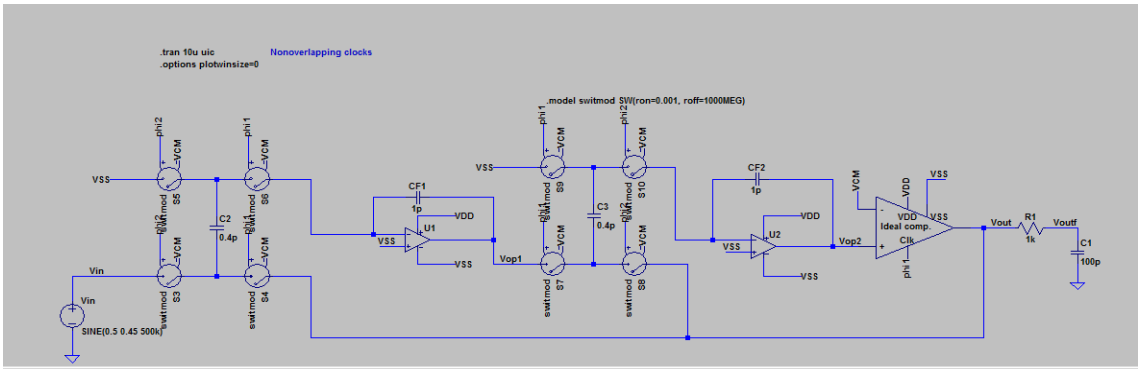

Figura 11 – Esquemático do modelador de segunda ordem

Para o cálculo do valor dos condensadores, recorreu-se à fórmula 22:

Equação 22

$$H(z) = \frac{Vout(z)}{Vin(z)} = \frac{CF1}{C2} \times \frac{z^{-1}}{1 - z^{-1}} = \frac{CF2}{C3} \times \frac{z^{-1}}{1 - z^{-1}}$$

Chegou-se à conclusão que, para condensadores de realimentação de capacidade 1pF, os outros condensadores teriam de ter 0.4pF de capacidade.

Observou-se então os gráficos para se verificar se de facto correspondiam à simulação realizada no *MatLab*. Assim, observou-se o sinal de saída, já filtrado (figura 12) bem como a sua transformada de Fourier (figura 13), o sinal de saída do circuito sem estar filtrado (figura 14) e a sua transformada de Fourier (figura 15), o sinal de saída do segundo integrador (figura 16) e a sua transformada de Fourier (figura 17), o sinal de saída do primeiro integrador (figura 18) e a sua transformada de Fourier (figura 19), quando tínhamos um sinal de entrada com as características descritas anteriormente (figura 20).

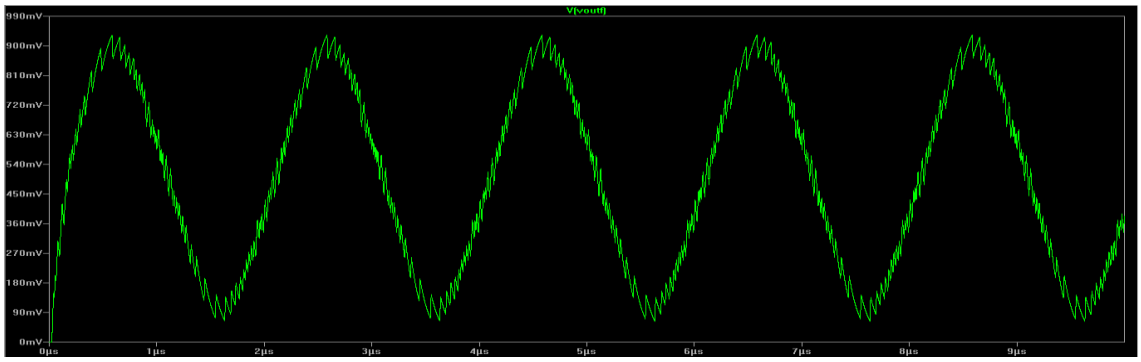

Figura 12 – Sinal de saída do modulador (filtrado)

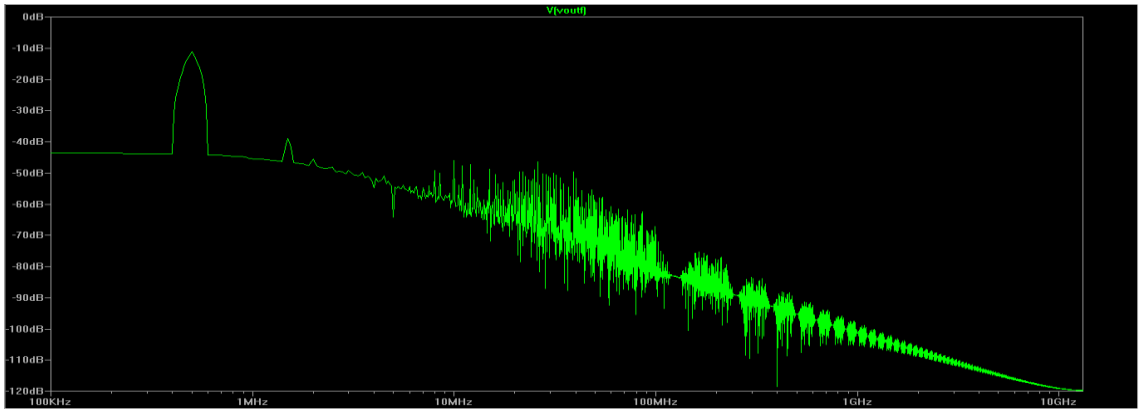

Figura 13 – Espectro na frequência (transformada de Fourier) do sinal de saída do modulador (filtrado)

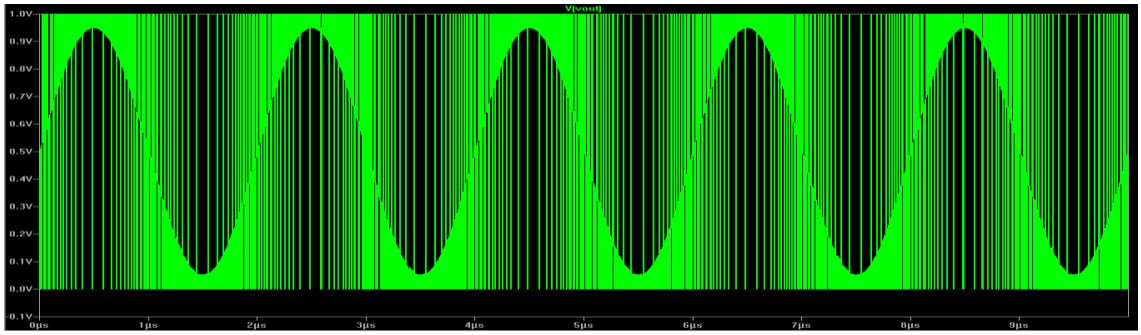

Figura 14 – Sinal de saída do modulador (não filtrado)

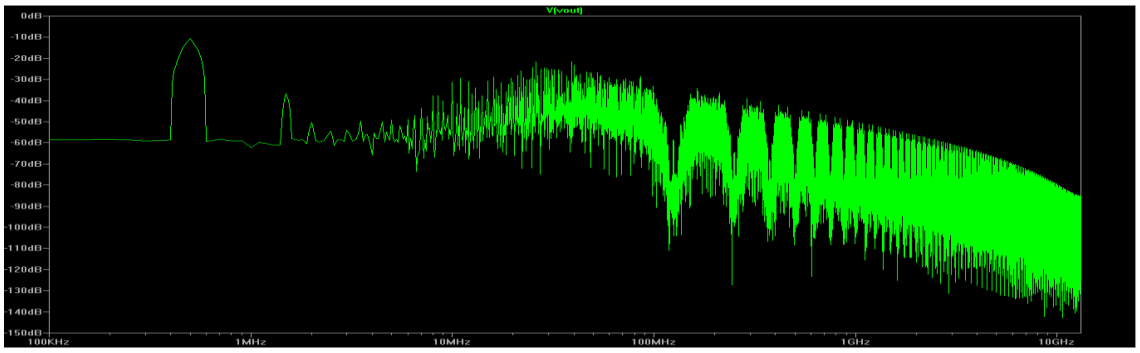

Figura 15 – Espectro na frequência (transformada de Fourier) do sinal de saída do modulador (não filtrado)

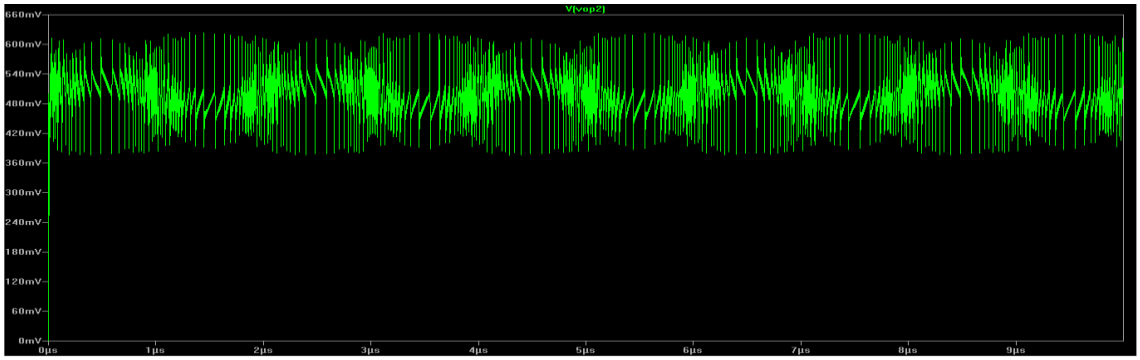

Figura 16 - Sinal de saída do segundo integrador

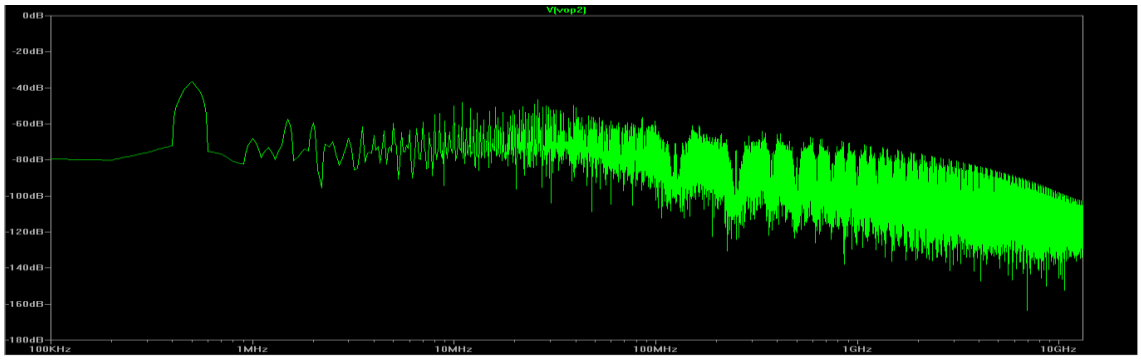

Figura 17 - Espectro na frequência (transformada de Fourier) do sinal de saída do segundo integrador

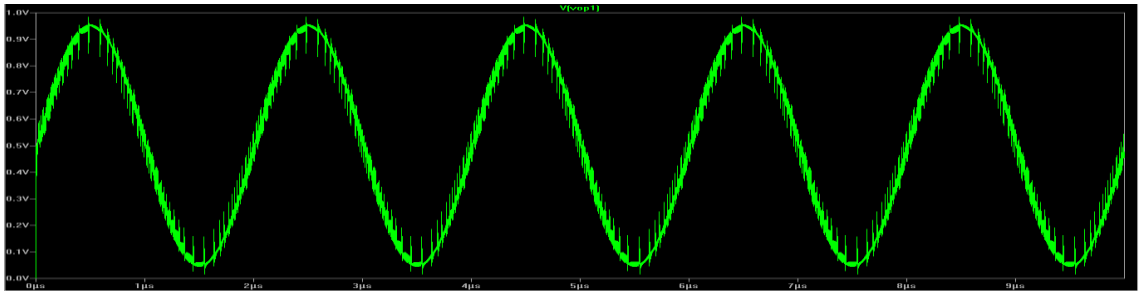

Figura 18 - Sinal de saída do primeiro integrador

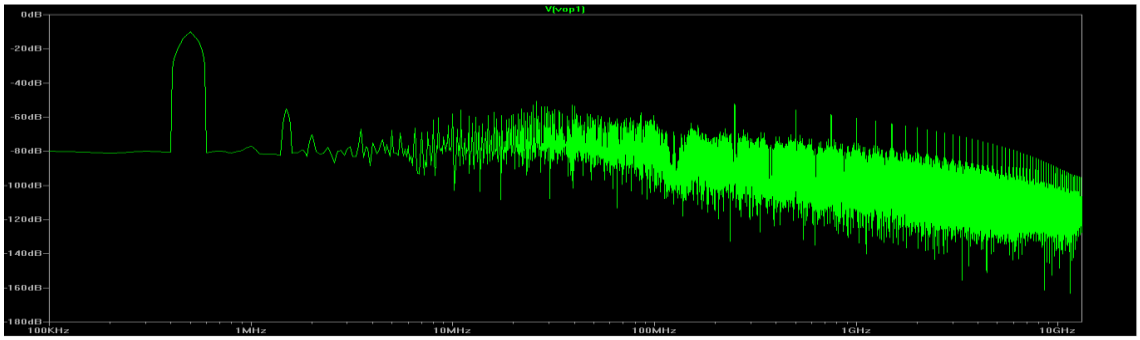

Figura 19 - Espectro na frequência (transformada de Fourier) do sinal de saída do primeiro integrador

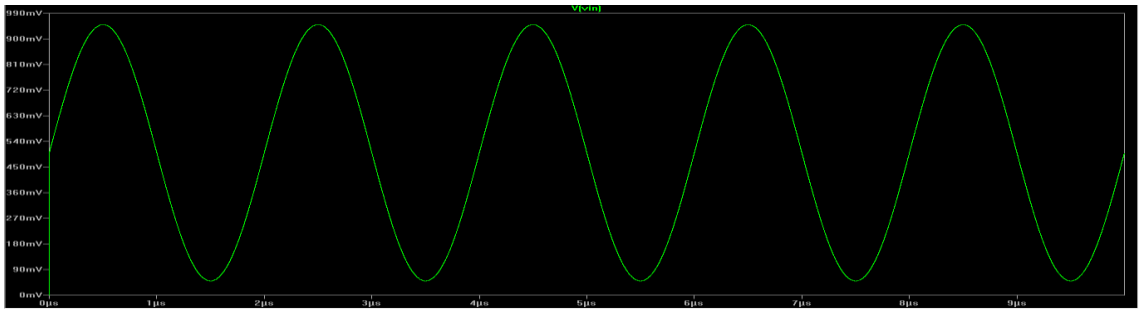

Figura 20 – Sinal de entrada do modulador

Para se ter uma melhor noção do que está a acontecer na frequência, sobrepôs-se todos os espectros no mesmo gráfico (figura 21).

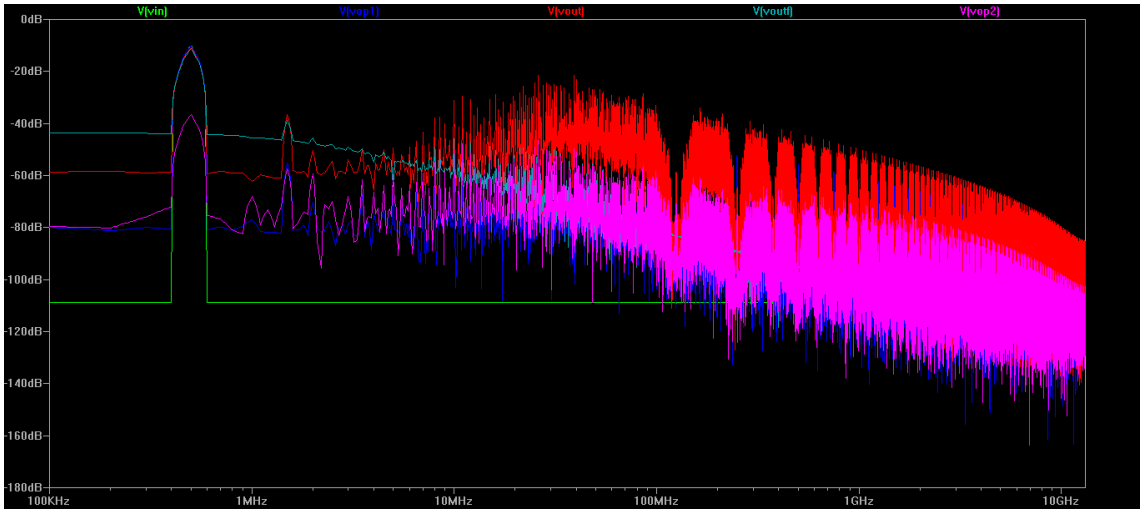

Figura 21 – Espectros na frequência do sinal de entrada (verde), da saída do primeiro integrador (azul escuro), da saída do segundo integrador (magenta), do sinal de saída do modulador não filtrado (vermelho) e do sinal de saída do modulador filtrado (cyan)

A partir da figura 21 pode-se concluir que os gráficos correspondem ao esperado, pois o sinal de entrada está definido apenas com uma harmónica (fundamental) e os restantes apresentam um comportamento *sinc* como já tinha sido visto na simulação pelo *MatLab*, tendo como principal harmónica, a mesma do sinal de entrada.

Depois desta simulação, foi introduzida saturação nos integradores, o que fez com que se esperasse uma degradação do sinal na saída. Partiu-se então para a observação dos gráficos nos mesmos pontos, ou seja observou-se novamente o sinal de saída, já filtrado (figura 22) bem como a sua transformada de Fourier (figura 23), o sinal de saída do circuito sem estar filtrado (figura 24) e a sua transformada de Fourier (figura 25), o sinal de saída do segundo integrador (figura 26) e a sua transformada de Fourier (figura 27), o sinal de saída do primeiro integrador (figura 28) e a sua transformada de Fourier (figura 29), para o mesmo sinal de entrada.

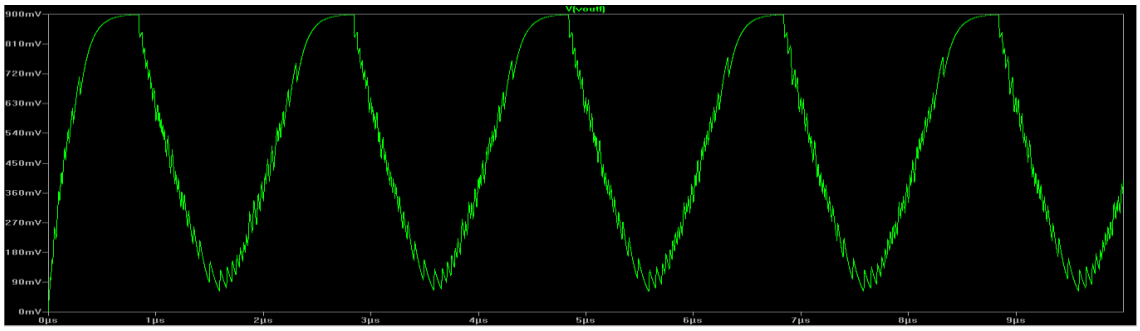

Figura 22 - Sinal de saída do modulador (filtrado)

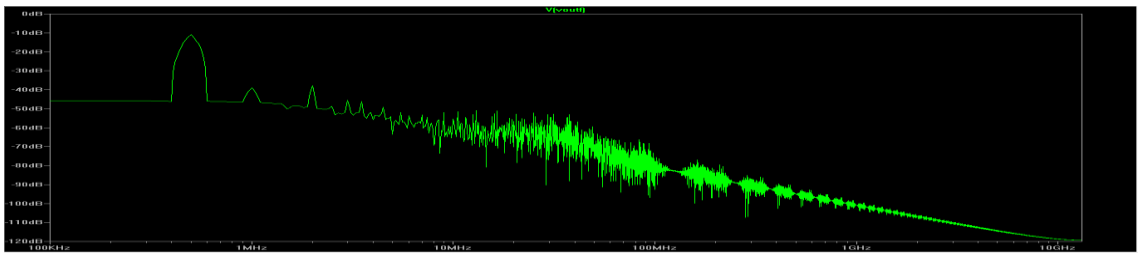

Figura 23 - Espectro na frequência (transformada de Fourier) do sinal de saída do modulador (filtrado)

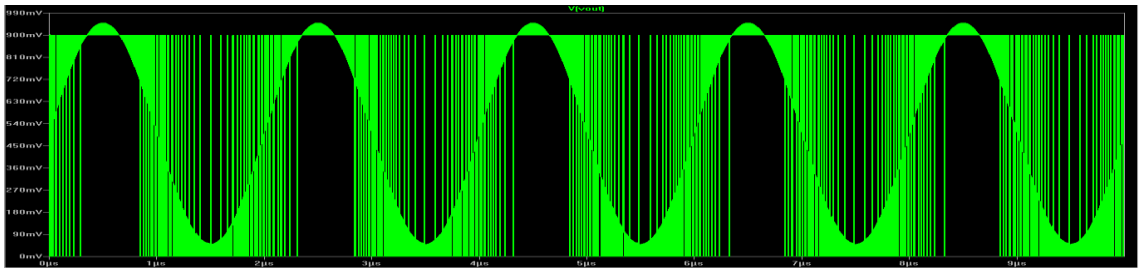

Figura 24 - Sinal de saída do modulador (não filtrado)

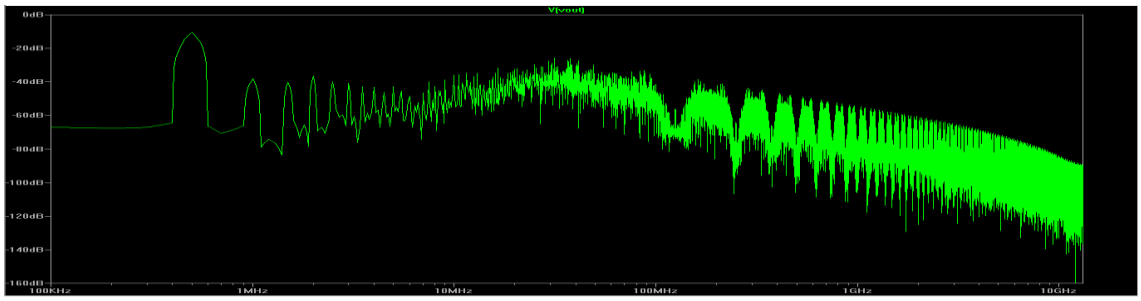

Figura 25 – Espectro na frequência (transformada de Fourier) do sinal de saída do modulador (não filtrado)

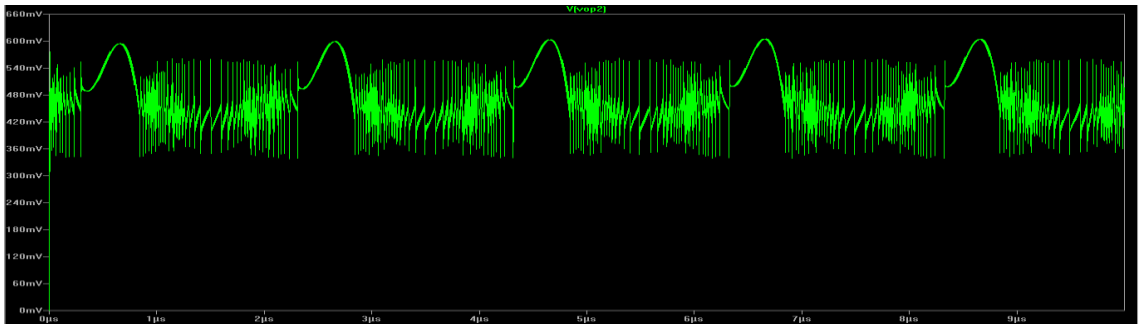

Figura 26 - Sinal de saída do segundo integrador

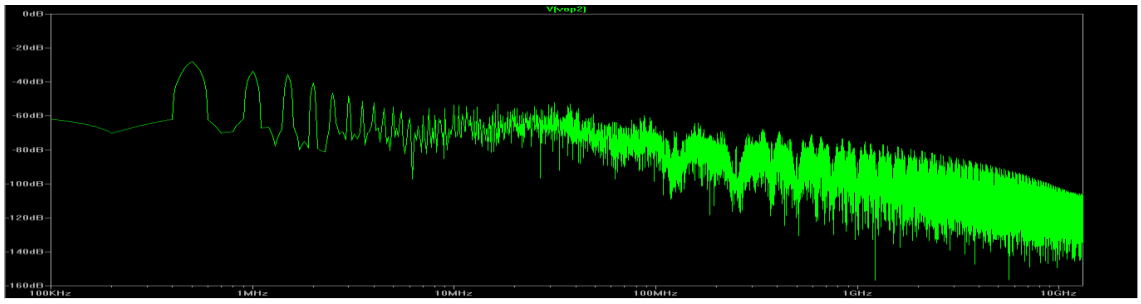

Figura 27 - Espectro na frequência (transformada de Fourier) do sinal de saída do segundo integrador

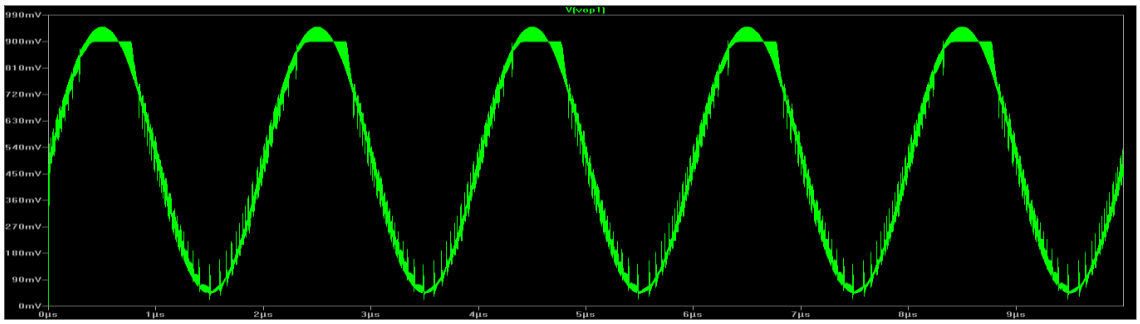

Figura 28 - Sinal de saída do primeiro integrador

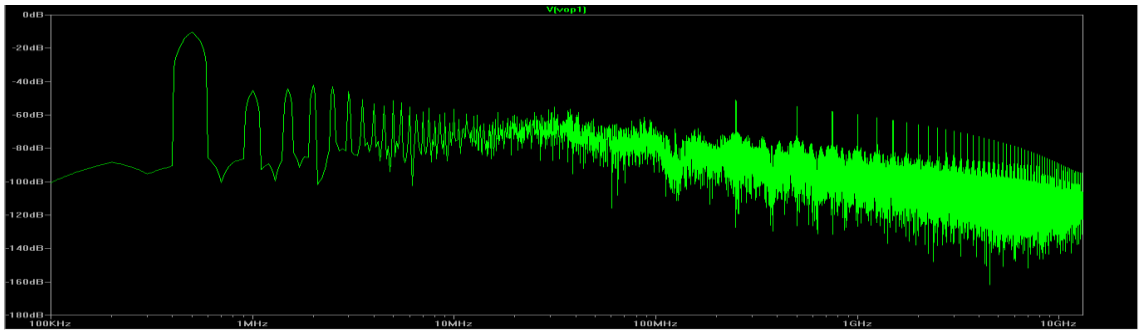

Figura 29 - Espectro na frequência (transformada de Fourier) do sinal de saída do primeiro integrador

Como se pôde verificar pelas figuras dos sinais, os comportamentos, a saturação "molda" a forma do sinal, havendo mais conteúdo harmónico. Assim o comportamento na frequência de cada saída será diferente.

À semelhança do modulador sem saturação, sobrepôs-se os espectros para se ter uma melhor noção do que estava a acontecer (figura 30).

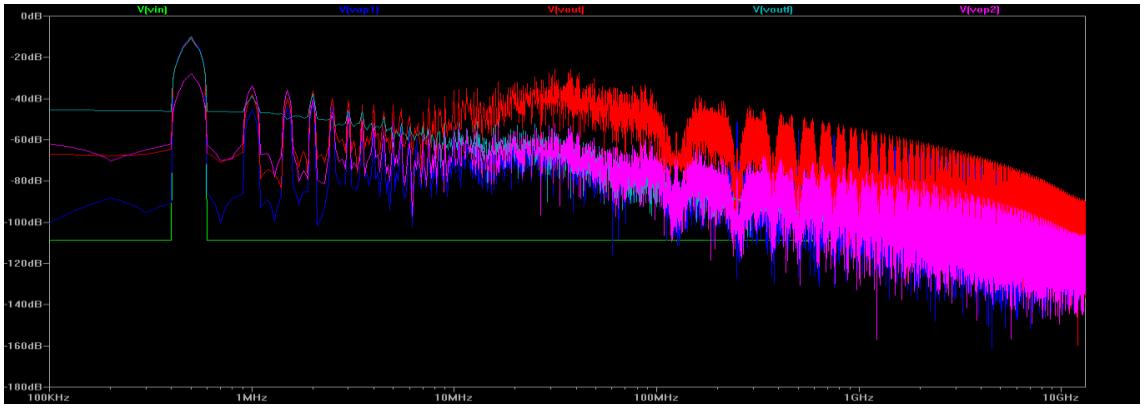

Figura 30 - Espectros na frequência do sinal de entrada (verde), da saída do primeiro integrador (azul escuro), da saída do segundo integrador (magenta), do sinal de saída do modulador não filtrado (vermelho) e do sinal de saída do modulador filtrado (cyan)

Como seria de esperar, por haver saturação nos integradores, existe mais conteúdo harmónico, como se pode verificar na figura 30.

Conclusão

No geral, o trabalho correspondeu às espectativas, na medida em que se pôde observar as SNDR dos diferentes sinais. Foi possível ver também os efeitos que o filtro causou nos diferentes sinais de saída dos integradores, bem como no sinal de saída geral do modulador.

Foi possível observar também, o efeito que a saturação introduz nos sinais, bem como nos seus comportamentos na frequência e também nos SNDR.

As especificações bateram certo com as simulações efectuadas, quer no programa *MatLab*, quer no programa *LTSpice*. Também bateu certo simulações efectuadas no *MatLab* com as mesmas efectuadas no *LTSpice*.

Como opção, foi pedido para realizar a mesma simulação, mas com um modulador MASH (2+1). Não foi possível realizar devido a falta de tempo, mas encontra-se no anexo 4 uma tentativa de simulação do mesmo no programa *MatLab*.

ANEXOS

```
%PARTE 2

%2.3

Vref=10^(5/20)

VQF = sqrt(2.5358*10^(-10)); %Vrms

Vlsb = Vref/2;

osr = ((Vlsb^2/12)*(pi^4/5)*(1/VQF^2))^(1/5)    %osr=128%=2^7

%2.4
PNtotaldB = -89.9382;  %dB
Pqf_dB = PNtotaldB-10; %dB

VQF = 10^(Pqf_dB/20)     %Vrms



n = (1/2)*log2(Vref^2/VQF^2) %Porque não há efeito de distorção!!!

%2.5
%Eletronica III - Modulador Sigma-Delta
        %%
clear all

%dimensionamento

%Especificações
fmax=16000;
vin_max=0.9;     % rms
SINAD1=76;
vin_max_dBV=20*log10(vin_max/1);
vin_min=0.08e-3;     %100 uVrms
SINAD2=8;
vin_min_dBV=20*log10(vin_min/1);

%nivel maximo em dBr (estimativa)
vin_max_dBr=-5;
Vref=vin_max*sqrt(2)/10^(vin_max_dBr/20)

%valor total do ruido do modulador
VN1_dBV=vin_max_dBV-SINAD1;
VN2_dBV=vin_min_dBV-SINAD2;

%escolher o ruido mais baixo como especificao
if VN1_dBV>VN2_dBV
    VN_dBV=VN2_dBV;
else
    VN_dBV=VN1_dBV;
end
```

```
% Ruido em dBr
factor=20*log10(sqrt(2)/Vref);
VN_dBr=VN_dBV+factor;

%Ruido em Vrms
VN=10^(VN_dBV/20)
VNQ=VN/2
VNT=VN*sqrt(3)/2

%calculo da sobreamostragem
osr=(((Vref/2)^2/12)*(pi^4/5)*1/VNQ^2)^(1/5)
osr=2^ceil(log(osr)/log(2))

Fs=fmax*2*osr

Vamp=vin_max*sqrt(2);

nbits=15;


b1=1;
b2=1;
K=2;


n=2^15; %numero de pontos na simulacao transiente
dec=round(Fs/48e3) %factor de decimacao
nmedias=10;
fin=round(fmax/(14*0.8)/Fs*n)*Fs/n;

time=0:1/n:1-1/n;
%time_dec=0:dec/(n):1-dec/(n);
time_dec=0:dec/(n):1;


%
%declarar as variaveis
clear vin e1 x11 x21 y1 out1 out2 out3
vin=zeros(1,n);
e1=zeros(1,n);
x11=zeros(1,n)+1e-6;
x21=zeros(1,n)+1e-6;
y1=zeros(1,n);
x12=zeros(1,n)+1e-6;
z1=zeros(1,n);
z2=zeros(1,n);
z3=zeros(1,n);
out1=zeros(1,n);
out2=zeros(1,n);
out3=zeros(1,n);


%



Vamp = vin_min*sqrt(2):0.01:2*vin_max*sqrt(2);
```

```
for k=1:length(Vamp)
for i=2:n
      %sinal de entrada
      vin(i)=Vamp(k)*sin(2*pi*i*fin/Fs)+randn*VNT;
      %Modulador de segunda ordem 1
      % primeiro integrador
      b1=1;
        e1(i)=vin(i)-b1*y1(i-1)*Vref;
        x11(i)=e1(i)+x11(i-1); %saída do primeiro integrador
      % segundo integrador
      b2=1;
        x21(i)=x11(i)-b2*y1(i-1)*Vref;
        x12(i)=x21(i)+x12(i-1);   %saída do segundo integrador
      %saída do modulador
      y1(i) = sign(x12(i));


      %Filtro decimador (sink1 sink2 sink3)
      z1(i)= z1(i-1)+y1(i)/dec;
      if i>dec
        out1(i)=z1(i)-z1((i-dec));
        z2(i)=z2(i-1)+out1(i)/dec;
        out2(i)=z2(i)-z2(floor(i-dec));
        z3(i)=z3(i-1)+out2(i)/dec;
        out3(i)=z3(i)-z3(floor(i-dec));
        else
        out1(i)=z1(i);
        z2(i)=z2(i-1)+out1(i)/dec;
        out2(i)=z2(i);
        z3(i)=z3(i-1)+out2(i)/dec;
        out3(i)=z3(i);
      end

end %for i



janela=     blackman(max(size(vin((0)+1:end))))';
janela_dec= blackman(max(size(vin((0)+1:dec:end))))';

vin_f=fft(vin((0)+1:end).*janela);
%para obter dBr multiplicar por sqrt(2)*sqrt(2)
vin_fp=vin_f.*conj(vin_f)*(2/n)^2;

y1_f=fft(y1((0)+1:end).*janela);
y1_fp=y1_f.*conj(y1_f)*(2/n)^2;

% sync filter
out1_f=fft(out1((0)+1:end).*janela);
out1_fp=out1_f.*conj(out1_f)*(2/n)^2;

out2_f=fft(out2((0)+1:end).*janela);
out2_fp=out2_f.*conj(out2_f)*(2/n)^2;

out3_f=fft(out3((0)+1:end).*janela);
out3_fp=out3_f.*conj(out3_f)*(2/n)^2;

out3dec_f=fft(out3((0)+1:dec:end).*janela_dec);
```

```
out3dec_fp=out3dec_f.*conj(out3dec_f)*(2*dec/n)^2;

out=round(out3((0)+1:dec:end)*2^(nbits-1));     %quantificao na saida
out_f=fft(out);
out_fp=out_f.*conj(out_f)*(2*dec/n)^2;



vin_fdBr=20*log10(abs(2*vin_f)/n+1e-10);
y1_fdBr=20*log10(abs(2*y1_f)/n+1e-10);
out1_fdBr=20*log10(abs(2*out1_f)/n+1e-10);
out2_fdBr=20*log10(abs(2*out2_f)/n+1e-10);
out3_fdBr=20*log10(abs(2*out3_f)/n+1e-10);
out3dec_fdBr=20*log10(abs(2*out3dec_f)/n+1e-10);
out_fdBr=20*log10(abs(2*out_f)/n+1e-10);

a=fmax*n/Fs;
[valor signal_index] = max(y1_fp(1:a));
Psignal = sum(y1_fp(signal_index-3:signal_index+3));
Pnoise = sum(y1_fp(1:a)) - Psignal;
sndr(k) = 10*log10(Psignal/Pnoise);     %SNDR/SINAD do sinal



vin_fdBr_=10*log10(vin_fp+1e-20);
y1_fdBr_=10*log10(y1_fp+1e-20);
out1_fdBr_=10*log10(out1_fp+1e-20);

out2_fdBr_=10*log10(out2_fp+1e-20);
out3_fdBr_=10*log10(out3_fp+1e-20);
out3dec_fdBr_=10*log10(out3dec_fp+1e-20);
out_fdBr_=10*log10(out_fp+1e-20);


out=10.^(out3dec_fdBr_/10);
[valor signal_index] = max(out(1:a));

Psignal2 = sum(out(signal_index-3:signal_index+3));
Pnoise2 = sum(out(1:a)) - Psignal2;
sndr2(k) = 10*log10(Psignal2/Pnoise2);  %SNDR do sinal filtrado

end




% plotting results


  %%
%4.1 %SNDR do filtro decimador
figure(5)
plot(20*log10(Vamp/Vref),sndr,'b')
grid on
```

```
hold on


plot(20*log10(Vamp/Vref),sndr2,'r')
grid on
title('SNDR do output do filtro (vermelho) e SINAD (azul)')

hold off

figure(6)
plot(20*log10(Vamp/Vref),sndr,'b')
grid on
title('SINAD')
```

Anexo1: Cálculo dos SNDR do filtro e da saída do modulador.

```
fmax=16000;
vin_max=0.9;     % rms
SINAD1=76;
vin_max_dBV=20*log10(vin_max/1);
vin_min=0.08e-3;     %100 uVrms
SINAD2=8;
vin_min_dBV=20*log10(vin_min/1);

%nivel maximo em dBr (estimativa)
vin_max_dBr=-5;
Vref=vin_max*sqrt(2)/10^(vin_max_dBr/20)

%valor total do ruido do modulador
VN1_dBV=vin_max_dBV-SINAD1;
VN2_dBV=vin_min_dBV-SINAD2;

%escolher o ruido mais baixo como especificao
if VN1_dBV>VN2_dBV
    VN_dBV=VN2_dBV;
else
    VN_dBV=VN1_dBV;
end

% Ruido em dBr
factor=20*log10(sqrt(2)/Vref);
VN_dBr=VN_dBV+factor;

%Ruido em Vrms
VN=10^(VN_dBV/20)
VNQ=VN/2
VNT=VN*sqrt(3)/2

%calculo da sobreamostragem
osr=(((Vref/2)^2/12)*(pi^4/5)*1/VNQ^2)^(1/5)
osr=2^ceil(log(osr)/log(2))

Fs=fmax*2*osr

Vamp=vin_max*sqrt(2);

nbits=15;
```

```
b1=1;
b2=1;
K=2;


n=2^15; %numero de pontos na simulacao transiente
dec=round(Fs/48e3) %factor de decimacao
nmedias=10;
fin=round(fmax/(14*0.8)/Fs*n)*Fs/n;

time=0:1/n:1-1/n;
%time_dec=0:dec/(n):1-dec/(n);
time_dec=0:dec/(n):1;


%
%declarar as variaveis
clear vin e1 x11 x21 y1 out1 out2 out3
vin=zeros(1,n);
e1=zeros(1,n);
x11=zeros(1,n)+1e-6;
x21=zeros(1,n)+1e-6;
y1=zeros(1,n);
x12=zeros(1,n)+1e-6;
z1=zeros(1,n);
z2=zeros(1,n);
z3=zeros(1,n);
out1=zeros(1,n);
out2=zeros(1,n);
out3=zeros(1,n);


%







for i=2:n
      %sinal de entrada
      vin(i)=Vamp*sin(2*pi*i*fin/Fs)+randn*VNT;
      %Modulador de segunda ordem 1
      % primeiro integrador
      b1=1;
        e1(i)=vin(i)-b1*y1(i-1)*Vref;
        x11(i)=e1(i)+x11(i-1); %saída do primeiro integrador
      % segundo integrador
      b2=1;
        x21(i)=x11(i)-b2*y1(i-1)*Vref;
        x12(i)=x21(i)+x12(i-1);   %saída do segundo integrador
      %saída do modulador
      y1(i) = sign(x12(i));
```

```
      %Filtro decimador (sink1 sink2 sink3)
      z1(i)= z1(i-1)+y1(i)/dec;
      if i>dec
        out1(i)=z1(i)-z1((i-dec));
        z2(i)=z2(i-1)+out1(i)/dec;
        out2(i)=z2(i)-z2(floor(i-dec));
        z3(i)=z3(i-1)+out2(i)/dec;
        out3(i)=z3(i)-z3(floor(i-dec));
        else
        out1(i)=z1(i);
        z2(i)=z2(i-1)+out1(i)/dec;
        out2(i)=z2(i);
        z3(i)=z3(i-1)+out2(i)/dec;
        out3(i)=z3(i);
      end

end %for i



janela=     blackman(max(size(vin((0)+1:end))))';
janela_dec= blackman(max(size(vin((0)+1:dec:end))))';

vin_f=fft(vin((0)+1:end).*janela);
%para obter dBr multiplicar por sqrt(2)*sqrt(2)
vin_fp=vin_f.*conj(vin_f)*(2/n)^2;

y1_f=fft(y1((0)+1:end).*janela);
y1_fp=y1_f.*conj(y1_f)*(2/n)^2;

% sync filter
out1_f=fft(out1((0)+1:end).*janela);
out1_fp=out1_f.*conj(out1_f)*(2/n)^2;

out2_f=fft(out2((0)+1:end).*janela);
out2_fp=out2_f.*conj(out2_f)*(2/n)^2;

out3_f=fft(out3((0)+1:end).*janela);
out3_fp=out3_f.*conj(out3_f)*(2/n)^2;

out3dec_f=fft(out3((0)+1:dec:end).*janela_dec);
out3dec_fp=out3dec_f.*conj(out3dec_f)*(2*dec/n)^2;

out=round(out3((0)+1:dec:end)*2^(nbits-1));     %quantificao na saida
out_f=fft(out);
out_fp=out_f.*conj(out_f)*(2*dec/n)^2;



vin_fdBr=20*log10(abs(2*vin_f)/n+1e-10);
y1_fdBr=20*log10(abs(2*y1_f)/n+1e-10);
out1_fdBr=20*log10(abs(2*out1_f)/n+1e-10);
out2_fdBr=20*log10(abs(2*out2_f)/n+1e-10);
out3_fdBr=20*log10(abs(2*out3_f)/n+1e-10);
out3dec_fdBr=20*log10(abs(2*out3dec_f)/n+1e-10);
out_fdBr=20*log10(abs(2*out_f)/n+1e-10);

a=fmax*n/Fs;
```

```
[valor signal_index] = max(y1_fp(1:a));
Psignal = sum(y1_fp(signal_index-3:signal_index+3));
Pnoise = sum(y1_fp(1:a)) - Psignal;
sndr = 10*log10(Psignal/Pnoise)    %SNDR/SINAD do sinal



vin_fdBr_=10*log10(vin_fp+1e-20);
y1_fdBr_=10*log10(y1_fp+1e-20);
out1_fdBr_=10*log10(out1_fp+1e-20);

out2_fdBr_=10*log10(out2_fp+1e-20);
out3_fdBr_=10*log10(out3_fp+1e-20);
out3dec_fdBr_=10*log10(out3dec_fp+1e-20);
out_fdBr_=10*log10(out_fp+1e-20);


out=10.^(out3dec_fdBr_/10);
[valor signal_index] = max(out(1:a));

Psignal2 = sum(out(signal_index-3:signal_index+3));
Pnoise2 = sum(out(1:a)) - Psignal2;
sndr2 = 10*log10(Psignal2/Pnoise2)  %SNDR do sinal filtrado







% plotting results


figure(1)
plot(time,vin,'r',time,out1,'b',time,out2,'g',time,out3,'m')
hold on
plot(time_dec,out3(1:dec:n),'wo')
hold off


f=1:max(size(vin_fdBr_));
f=(f-1)*Fs/f(end);

figure(2)
semilogx(f(1:end/2),vin_fdBr_(1:end/2),'r',f(1:end/2),out1_fdBr_(1:end
/2),'b',f(1:end/2),out2_fdBr_(1:end/2),'m',f(1:end/2),out3_fdBr_(1:end
/2),'c')
legend('r;vin;','g;y;','b;out1;','m;out2;','c;out3;')
grid on

figure(3)
semilogx(f(1:end/2),vin_fdBr_(1:end/2),'r',f(1:end/2),y1_fdBr_(1:end/2
),'b')
grid on

figure(4)
f_dec=f(1:dec:end)/dec;
```

```
plot(f_dec(1:end/2),out3dec_fdBr_(1:end/2),'r')
grid on
title('out decimado')
```

Anexo2: Verificação das especificações

```
%4.2

%Eletronica III - Modulador Sigma-Delta
        %%
clear all

%dimensionamento

%Especificações
fmax=16000;
vin_max=0.9;     % rms
SINAD1=76;
vin_max_dBV=20*log10(vin_max/1);
vin_min=0.08e-3;     %100 uVrms
SINAD2=8;
vin_min_dBV=20*log10(vin_min/1);

%nivel maximo em dBr (estimativa)
vin_max_dBr=-5;
Vref=vin_max*sqrt(2)/10^(vin_max_dBr/20)

%valor total do ruido do modulador
VN1_dBV=vin_max_dBV-SINAD1;
VN2_dBV=vin_min_dBV-SINAD2;

%escolher o ruido mais baixo como especificao
if VN1_dBV>VN2_dBV
    VN_dBV=VN2_dBV;
else
    VN_dBV=VN1_dBV;
end

% Ruido em dBr
factor=20*log10(sqrt(2)/Vref);
VN_dBr=VN_dBV+factor;

%Ruido em Vrms
VN=10^(VN_dBV/20)
VNQ=VN/2
VNT=VN*sqrt(3)/2

%calculo da sobreamostragem
osr=(((Vref/2)^2/12)*(pi^4/5)*1/VNQ^2)^(1/5)
osr=2^ceil(log(osr)/log(2))

Fs=fmax*2*osr

Vamp=vin_max*sqrt(2);

nbits=15;
```

```
b1=1;
b2=1;
K=2;


n=2^15; %numero de pontos na simulacao transiente
dec=round(Fs/48e3) %factor de decimacao
nmedias=10;
fin=round(fmax/(14*0.8)/Fs*n)*Fs/n;

time=0:1/n:1-1/n;
%time_dec=0:dec/(n):1-dec/(n);
time_dec=0:dec/(n):1;


%
%declarar as variaveis
clear vin e1 x11 x21 y1 out1 out2 out3
vin=zeros(1,n);
e1=zeros(1,n);
x11=zeros(1,n)+1e-6;
x21=zeros(1,n)+1e-6;
y1=zeros(1,n);
x12=zeros(1,n)+1e-6;
z1=zeros(1,n);
z2=zeros(1,n);
z3=zeros(1,n);
out1=zeros(1,n);
out2=zeros(1,n);
out3=zeros(1,n);


%

sat=12;


Vamp = vin_min*sqrt(2):0.01:2*vin_max*sqrt(2);

for k=1:length(Vamp)
for i=2:n
      %sinal de entrada
      vin(i)=Vamp(k)*sin(2*pi*i*fin/Fs)+randn*VNT;
      %Modulador de segunda ordem 1
      % primeiro integrador
      b1=1;
        e1(i)=vin(i)-b1*y1(i-1)*Vref;
        x11(i)=e1(i)+x11(i-1); %saída do primeiro integrador
        if x11(i)>sat
            x11(i)=sat;
        else
           if x11(i)<-sat
                x11(i)=-sat;
           end
      end
      % segundo integrador
      b2=1;
```

```
        x21(i)=x11(i)-b2*y1(i-1)*Vref;
        x12(i)=x21(i)+x12(i-1);   %saída do segundo integrador
        if x12(i)>sat
            x12(i)=sat;
        else
            if x12(i)<-sat
                x12(i)=-sat;
            end
      end
      %saída do modulador
      y1(i) = sign(x12(i));
      if y1(i)>sat
          y1(i)=sat;
      else
           if y1(i)<-sat
                y1(i)=-sat;
           end
      end


      %Filtro decimador (sink1 sink2 sink3)
      z1(i)= z1(i-1)+y1(i)/dec;
      if i>dec
        out1(i)=z1(i)-z1((i-dec));
        z2(i)=z2(i-1)+out1(i)/dec;
        out2(i)=z2(i)-z2(floor(i-dec));
        z3(i)=z3(i-1)+out2(i)/dec;
        out3(i)=z3(i)-z3(floor(i-dec));
        else
        out1(i)=z1(i);
        z2(i)=z2(i-1)+out1(i)/dec;
        out2(i)=z2(i);
        z3(i)=z3(i-1)+out2(i)/dec;
        out3(i)=z3(i);
      end

end %for i



janela=     blackman(max(size(vin((0)+1:end))))';
janela_dec= blackman(max(size(vin((0)+1:dec:end))))';

vin_f=fft(vin((0)+1:end).*janela);
%para obter dBr multiplicar por sqrt(2)*sqrt(2)
vin_fp=vin_f.*conj(vin_f)*(2/n)^2;

y1_f=fft(y1((0)+1:end).*janela);
y1_fp=y1_f.*conj(y1_f)*(2/n)^2;

% sync filter
out1_f=fft(out1((0)+1:end).*janela);
out1_fp=out1_f.*conj(out1_f)*(2/n)^2;

out2_f=fft(out2((0)+1:end).*janela);
out2_fp=out2_f.*conj(out2_f)*(2/n)^2;

out3_f=fft(out3((0)+1:end).*janela);
out3_fp=out3_f.*conj(out3_f)*(2/n)^2;
```

```
out3dec_f=fft(out3((0)+1:dec:end).*janela_dec);
out3dec_fp=out3dec_f.*conj(out3dec_f)*(2*dec/n)^2;

out=round(out3((0)+1:dec:end)*2^(nbits-1));     %quantificao na saida
out_f=fft(out);
out_fp=out_f.*conj(out_f)*(2*dec/n)^2;



vin_fdBr=20*log10(abs(2*vin_f)/n+1e-10);
y1_fdBr=20*log10(abs(2*y1_f)/n+1e-10);
out1_fdBr=20*log10(abs(2*out1_f)/n+1e-10);
out2_fdBr=20*log10(abs(2*out2_f)/n+1e-10);
out3_fdBr=20*log10(abs(2*out3_f)/n+1e-10);
out3dec_fdBr=20*log10(abs(2*out3dec_f)/n+1e-10);
out_fdBr=20*log10(abs(2*out_f)/n+1e-10);

a=fmax*n/Fs;
[valor signal_index] = max(y1_fp(1:a));
Psignal = sum(y1_fp(signal_index-3:signal_index+3));
Pnoise = sum(y1_fp(1:a)) - Psignal;
sndr(k) = 10*log10(Psignal/Pnoise);     %SNDR/SINAD do sinal



vin_fdBr_=10*log10(vin_fp+1e-20);
y1_fdBr_=10*log10(y1_fp+1e-20);
out1_fdBr_=10*log10(out1_fp+1e-20);

out2_fdBr_=10*log10(out2_fp+1e-20);
out3_fdBr_=10*log10(out3_fp+1e-20);
out3dec_fdBr_=10*log10(out3dec_fp+1e-20);
out_fdBr_=10*log10(out_fp+1e-20);


out=10.^(out3dec_fdBr_/10);
[valor signal_index] = max(out(1:a));
if signal_index<=2
    Psignal2 = sum(out(signal_index:signal_index+3));
else
    Psignal2 = sum(out(signal_index-3:signal_index+3));
end
Pnoise2 = sum(out(1:a)) - Psignal2;
sndr2(k) = 10*log10(Psignal2/Pnoise2);  %SNDR do sinal filtrado

end

figure(5)
plot(20*log10(Vamp/Vref),sndr,'b')
grid on

hold on

plot(20*log10(Vamp/Vref),sndr2,'r')
grid on
title('SNDR')
```

```
figure(6)
plot(20*log10(Vamp/Vref),sndr,'b')
grid on
title('SNDR')
```

Anexo 3: Verificação do SNDR com saturação.

```
%---------MASH(2+1)------------%

fmax=16000;
vin_max=0.9;     % rms
SINAD1=76;
vin_max_dBV=20*log10(vin_max/1);
vin_min=0.08e-3;     %100 uVrms
SINAD2=8;
vin_min_dBV=20*log10(vin_min/1);

%nivel maximo em dBr (estimativa)
vin_max_dBr=-5;
Vref=vin_max*sqrt(2)/10^(vin_max_dBr/20)

%valor total do ruido do modulador
VN1_dBV=vin_max_dBV-SINAD1;
VN2_dBV=vin_min_dBV-SINAD2;

%escolher o ruido mais baixo como especificao
if VN1_dBV>VN2_dBV
    VN_dBV=VN2_dBV;
else
    VN_dBV=VN1_dBV;
end

% Ruido em dBr
factor=20*log10(sqrt(2)/Vref);
VN_dBr=VN_dBV+factor;

%Ruido em Vrms
VN=10^(VN_dBV/20)
VNQ=VN/2
VNT=VN*sqrt(3)/2

%calculo da sobreamostragem
osr=(((Vref/2)^2/12)*(pi^4/5)*1/VNQ^2)^(1/5)
osr=2^ceil(log(osr)/log(2))

Fs=fmax*2*osr

Vamp=vin_max*sqrt(2);

nbits=15;
```

```
b1=1;
b2=1;
K=2;


n=2^15; %numero de pontos na simulacao transiente
dec=round(Fs/48e3) %factor de decimacao
nmedias=10;
fin=round(fmax/(14*0.8)/Fs*n)*Fs/n;

time=0:1/n:1-1/n;
%time_dec=0:dec/(n):1-dec/(n);
time_dec=0:dec/(n):1;


%
%declarar as variaveis
clear vin e1 x11 x21 y1 out1 out2 out3
vin=zeros(1,n);
e1=zeros(1,n);
x11=zeros(1,n)+1e-6;
x21=zeros(1,n)+1e-6;
y1=zeros(1,n);
x12=zeros(1,n)+1e-6;
z1=zeros(1,n);
z2=zeros(1,n);
z3=zeros(1,n);
out1=zeros(1,n);
out2=zeros(1,n);
out3=zeros(1,n);
dout1=zeros(1,n);

%Modulador de primeira ordem (especificações)
vin2=zeros(1,n);
e2=zeros(1,n);
x2=zeros(1,n);
dout2=zeros(1,n);
%
dout=zeros(1,n);
z4=zeros(1,n);
out4=zeros(1,n);


for i=3:n
      %sinal de entrada
      vin(i)=Vamp*sin(2*pi*i*fin/Fs)+randn*VNT;
      %Modulador de segunda ordem 1
      % primeiro integrador
      b1=1;
        e1(i)=vin(i)-b1*dout1(i-1)*Vref;
        x11(i)=e1(i)+x11(i-1); %saída do primeiro integrador
      % segundo integrador
      b2=1;
        x21(i)=x11(i)-b2*dout1(i-1)*Vref;
        x12(i)=x21(i)+x12(i-1);   %saída do segundo integrador
      %saída do modulador de primeira ordem
      dout1(i) = sign(x12(i));

      %entrada do modulador de primeira ordem
      vin2(i)=dout1(i)-x12(i);
```

```
        e2(i)=vin2(i)-dout2(i);
        x2(i)=e2(i-1)+x2(i);
        dout2(i)=sign(x2(i));

        %Lógica de cancelamento de ruído
        dout(i)=dout1(i-1)-dout2(i)+2*dout2(i-1)-dout(i-2);


        %Filtro decimador (sink1 sink2 sink3)
        z1(i)= z1(i-1)+dout(i)/dec;
        if i>dec
          out1(i)=z1(i)-z1((i-dec));
          z2(i)=z2(i-1)+out1(i)/dec;
          out2(i)=z2(i)-z2(floor(i-dec));
          z3(i)=z3(i-1)+out2(i)/dec;
          out3(i)=z3(i)-z3(floor(i-dec));
          z4(i)=z4(i-1)+out3(i)/dec;
          out4(i)=z4(i)-z4(floor(i-dec));
          else
          out1(i)=z1(i);
          z2(i)=z2(i-1)+out1(i)/dec;
          out2(i)=z2(i);
          z3(i)=z3(i-1)+out2(i)/dec;
          out3(i)=z3(i);
          z4(i)=z4(i-1)+out3(i)/dec;
          out4(i)=z4(i);
        end

end %for i



janela=     blackman(max(size(vin((0)+1:end))))';
janela_dec= blackman(max(size(vin((0)+1:dec:end))))';

vin_f=fft(vin((0)+1:end).*janela);
%para obter dBr multiplicar por sqrt(2)*sqrt(2)
vin_fp=vin_f.*conj(vin_f)*(2/n)^2;

y1_f=fft(dout((0)+1:end).*janela);
y1_fp=y1_f.*conj(y1_f)*(2/n)^2;

% sync filter
out1_f=fft(out1((0)+1:end).*janela);
out1_fp=out1_f.*conj(out1_f)*(2/n)^2;

out2_f=fft(out2((0)+1:end).*janela);
out2_fp=out2_f.*conj(out2_f)*(2/n)^2;

out3_f=fft(out3((0)+1:end).*janela);
out3_fp=out3_f.*conj(out3_f)*(2/n)^2;

out4_f=fft(out4((0)+1:end).*janela);
out4_fp=out4_f.*conj(out4_f)*(2/n)^2;

% out3dec_f=fft(out3((0)+1:dec:end).*janela_dec);
% out3dec_fp=out3dec_f.*conj(out3dec_f)*(2*dec/n)^2;
```

```
out4dec_f=fft(out4((0)+1:dec:end).*janela_dec);
out4dec_fp=out4dec_f.*conj(out4dec_f)*(2*dec/n)^2;

out=round(out4((0)+1:dec:end)*2^(nbits-1));     %quantificao na saida
out_f=fft(out);
out_fp=out_f.*conj(out_f)*(2*dec/n)^2;



vin_fdBr=20*log10(abs(2*vin_f)/n+1e-10);
y1_fdBr=20*log10(abs(2*y1_f)/n+1e-10);
out1_fdBr=20*log10(abs(2*out1_f)/n+1e-10);
out2_fdBr=20*log10(abs(2*out2_f)/n+1e-10);
out3_fdBr=20*log10(abs(2*out3_f)/n+1e-10);
out4_fdBr=20*log10(abs(2*out4_f)/n+1e-10);
out4dec_fdBr=20*log10(abs(2*out4dec_f)/n+1e-10);
out_fdBr=20*log10(abs(2*out_f)/n+1e-10);

a=fmax*n/Fs;
[valor signal_index] = max(y1_fp(1:a));
Psignal = sum(y1_fp(signal_index-3:signal_index+3));
Pnoise = sum(y1_fp(1:a)) - Psignal;
sndr = 10*log10(Psignal/Pnoise)   %SNDR/SINAD do sinal



vin_fdBr_=10*log10(vin_fp+1e-20);
y1_fdBr_=10*log10(y1_fp+1e-20);
out1_fdBr_=10*log10(out1_fp+1e-20);
out2_fdBr_=10*log10(out2_fp+1e-20);
out3_fdBr_=10*log10(out3_fp+1e-20);
out4_fdBr_=10*log10(out4_fp+1e-20);

out4dec_fdBr_=10*log10(out4dec_fp+1e-20);
out_fdBr_=10*log10(out_fp+1e-20);



out=10.^(out4dec_fdBr_/10);
[valor signal_index] = max(out(1:a));

Psignal2 = sum(out(signal_index-3:signal_index+3));
Pnoise2 = sum(out(1:a)) - Psignal2;
sndr2 = 10*log10(Psignal2/Pnoise2)  %SNDR do sinal filtrado









% plotting results



figure(1)
plot(time,vin,'r',time,out1,'b',time,out2,'g',time,out3,'m',time,out4,
'y')
hold on
```

```
plot(time_dec,out3(1:dec:n),'wo')
hold off


f=1:max(size(vin_fdBr_));
f=(f-1)*Fs/f(end);

figure(2)
semilogx(f(1:end/2),vin_fdBr_(1:end/2),'r',f(1:end/2),out1_fdBr_(1:end
/2),'b',f(1:end/2),out2_fdBr_(1:end/2),'m',f(1:end/2),out3_fdBr_(1:end
/2),'c',f(1:end/2),out4_fdBr_(1:end/2),'y')
legend('r;vin;','g;y;','b;out1;','m;out2;','c;out3;')
grid on

figure(3)
semilogx(f(1:end/2),vin_fdBr_(1:end/2),'r',f(1:end/2),y1_fdBr_(1:end/2
),'b')
grid on

figure(4)
f_dec=f(1:dec:end)/dec;
plot(f_dec(1:end/2),out4dec_fdBr_(1:end/2),'r')
grid on
title('out decimado')
```

Anexo 5: Tentativa de concretização do MASH (2+1)

```
B=16000;%Hz

%------ADC-------%

f=48000;%Hz

THDmax=-78;%dB

A=0.9;%Vrms

SNRmin=8;%dB

Amin=0.08*10^(-3);%Vrms

%------ADC-------%


%1.1
x1=20*log10(A);       %esboço da recta
x2=20*log10(Amin);    %
y1=76;
y2=8;

X=[x1 x2];
Y=[y1 y2];

X2=[x1 x2 1 1.1 1.2 1.5 1.6];
Y2=[y1 y2 y1 65 60 10 0];

figure(1)
plot(X,Y,'y')
```

```
hold on
plot(X2,Y2,'r')
title('Esboço da SINAD esperada (a vermelho)')
hold off


%%O pior caso corresponde à amplitude mais longe dos 0dB

%1.2
Vref=10^(5/20);%Vrms -> -5=20*log(Vamp/Vref)

VampREF=sqrt(2)*Vref

%1.3
SNDR=SNRmin;

PSdB=20*log10(Amin)

PNtotaldB=PSdB-SNDR

%1.4
PNtotal=10^(PNtotaldB/10) %W

Pnq=PNtotal/4 %Vrms
Pnt=Pnq*3           %Vrms

Vnq_dB= 20*log10(sqrt(Pnq)) %dBV
Vnt_dB= 20*log10(sqrt(Pnt)) %dBV
```

Anexo 6: Consolidação das especificações