DAFTAR PUSTAKA

[1] Guidolin, F. dan Nekovee, M. 2015, "Investigating Spectrum Sharing between 5G Millimeter Wave Networks and Fixed Satellite Systems," *IEEE Globecom Workshops (GC Workshop)*, San Diego, CA, hal. 1-7.

[2] Evans, B. G., 2014, "The role of satellites in 5G," , *the 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, Livorno, hal. 197-202.

[3] Ericsson Inc . 2018 . "*More than 50 billion connected devices"*, http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf. [Online akses 27 September 2018].

[4] Halide, L. dkk (2020). *Aturan Pemodelan Komputasi Untuk Eliminasi Signal Spektrum-Rf Yang Tidak Diinginkan Computational Modeling Rule For Unwanted Signal Rf-Spectrum Elimination*. *7*(1), 797–804.

[5] Chae, K., dan Yoon, S., 2014, "Cancellation of AltBOC Correlation Side-Peaks for Frequency Sharing in Satellite Communication Spectrum", the journal of Korea Information and Communication Society, edisi 39B, Vol. 11, hal 810-816.

[6] Sirmayanti, S., dan Faulkner, M., 2014, "ΣΔ Modulator for Digital Wireless Architecture: A review*", IEEE MICEEI International Conference*, hal. 83-87

[7] Miao, Yao., Munawwar, M. S., Xiaofu, M., Vuk, M., dan Jeffrey, H., R. 2019, "Sustainable green networking: exploiting degrees of freedom towards energy-efficient 5G systems", Journal of Wireless Network, Volume 25, Issue 3, hal 951–960.

[8] Helaoui, M., Hatami, S., Negra, R., dan Ghannouchi, F. M., 2008, "A novel architecture of delta-sigma modulator enabling all-digital ultiband multistandard RF transmitters design", IEEE Transactions on ircuits and Systems II: Express Briefs, vol. 55, hal. 1129- 1133.

assoo, V., dan Faulkner, M., 2009, "Sigma-delta digital drive signals for

switchmode power amplifiers", Journal Electronics Letters, vol. 44, hal. 1299-1300.

[10] Nielsen, M., dan Larsen, T., 2007, "A transmitter architecture based on delta–sigma modulation and switch-mode power amplification", IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 54, hal. 735-739.

[11] Beckman, J. A. (1983). Communication and data transmission systems. *Manual of Remote Sensing, 2nd Edition. Vol I*, 681–698.

[12] Ii, B. A. B., & Pustaka, T. (2004). *Hwei P. Hsu. 2004. Komunikasi Analog dan Digital, Jakarta: PT Gelora Aksara Pratama. hlm.76 5*. 5–21.

[13] Irtawaty, A. S. (2019). Implementasi Metode Fast Fourier Transform (Fft) Dalam Mengklasifikasikan Suara Pria Dan Wanita Di Laboratorium Jurusan Teknik Elektro Politeknik Negeri Balikpapan. *JTT (Jurnal Teknologi Terpadu)*, *7*(2), 70–75. https://doi.org/10.32487/jtt.v7i2.661

[14] Armstrong, J. (2009). OFDM for Optical Communications(Invited Tutorial). *Journal of Lightwave Technology*, *27*(3), 189–204.

[15] Buku Telekomunikasi dan Komputer pdf. Bab 35 "*Noise_dan_Distorsi*" di akses pada Maret 2021 pukul 10.00. hlm 724-725.

[16] Haldi, M.W. (2019). Teknologi Jaringan 5G (https://binus.ac.id/bandung/2019/12/teknologi-jaringan-5g/ diakses pada 19 Mei 2021)

[17] Schreier, R., Temes, G. C., dan Wiley, J., 2005, "Understanding delta-sigma data converters", IEEE press Piscataway, NJ, vol. 74.

[18] Schreier, R., Temes, G. C., dan Wiley, J., 2005, "Understanding delta-sigma data converters", IEEE press Piscataway, NJ, vol. 74.

[19] Keyzer, J., Hinrichs, J., Metzger, A., Iwamoto, M., Galton, I., dan Asbeck, P., 2001, "Digital generation of RF signals for wireless communications with band-pass DS modulation", *Microwave Symposium Digest IEEE MTT-S International*, hal. 2127-2130.

eyzer, J., Uang, R., Sugiyama, Y., Iwamoto, M., Galton, I., dan Asbeck, , 2002, "Generation of RF pulsewidth modulated microwave signals

using delta-sigma modulation", *Microwave Symposium Digest IEEE MTT-S International,* hal. 397-400.

[21] Sirmayanti, S., Bassoo, V., King, H., dan M. Faulkner, M., 2011, "Sigma delta (ΣΔ) architecture integration with digital pre-distortion to enhance optimal switch mode power amplification (OSMPA) in FEMTO cell transceiver design", *IEEE 8th International Conference on Information, Communications and Signal Processing (ICICS),* hal. 1-4.

[22] Sirmayanti, S., Bassoo, V., dan Faulkner, M., 2012, "OFDM performance with Odd-Even Quantisation in Cartesian DS upconverters*", IEEE International Conf on Signal Processing and Communication Systems (ICPCS),* hal. 1-5.

[23] Sirmayanti, S., Bassoo, V., dan Faulkner, M., 2013, "Joint odd-even quantisation in Cartesian Delta-Sigma (DS) upconverters", *2013 IEEE AFRICON,* hal. 1-4.

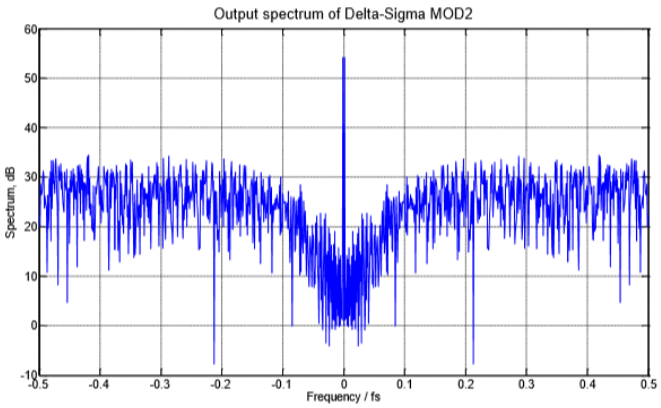
[24] Sirmayanti, S., dan Faulkner, M., 2014, "Tuning baseband on Cartesian Delta-Sigma Up-conversion". IET e-Letters Journal, Vol. 50, Iss.8, hal. 635-637.

[25] Lidemar, H., dan Sirmayanti, S., 2016' "*Fenomena noise shaping dan harmonik pada ΣΔ-based RF Transmiter untuk aplikasi sofware radio multiband 5G*", Laporan Penelitian Hibah Bersaing 2016.

LAMPIRAN

FIGURE RESULT

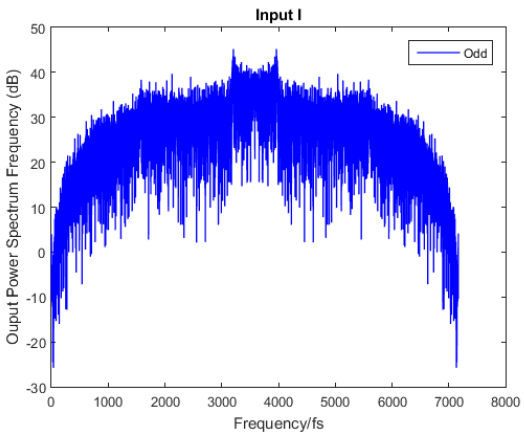
Output Spectrum Delta-Sigma MOD2

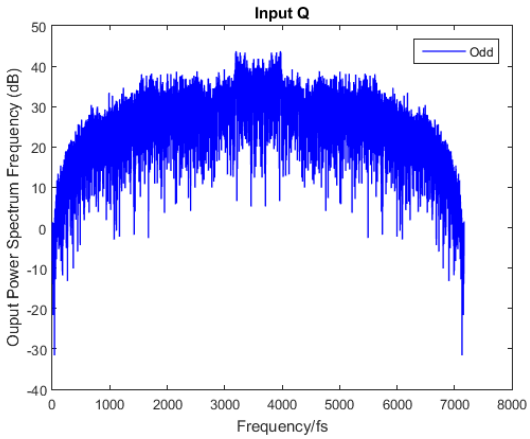


Gambar 5. Noise shaping pada filter ΔΣ MOD2 (sample per period =1024).

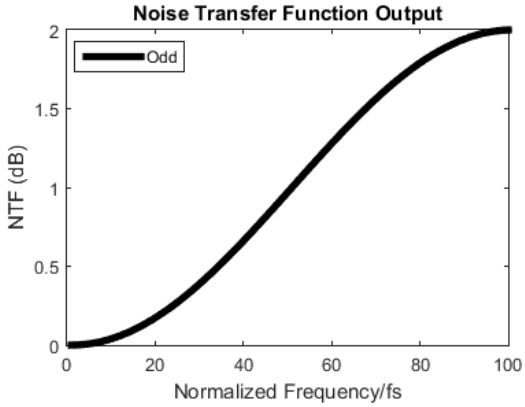
Sumber : Analisis Fenomena Harmonik Pasca Proses PWM/PPM Pada Struktur RF-Upconverter (Sirmayanti, Ichsan Mahmud, 2017)

Ploting Input I dan Q



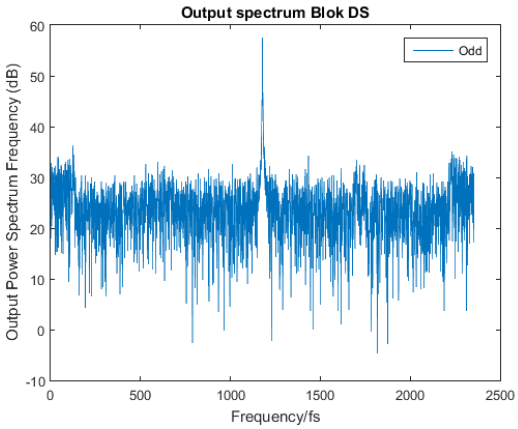


Ploting NTF

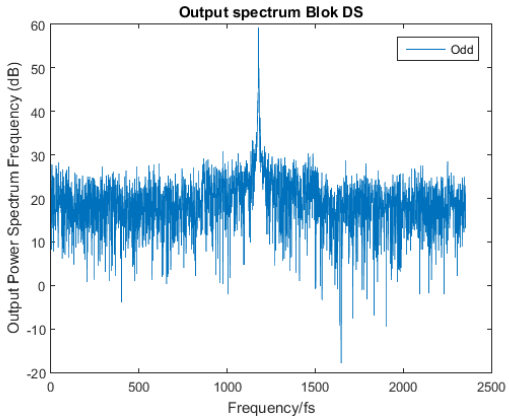


Ploting Keluaran Delta Sigma

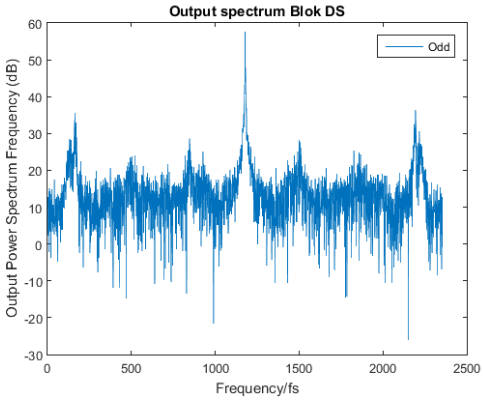
a. Frekuensi 2,3 GHz



OSR 4

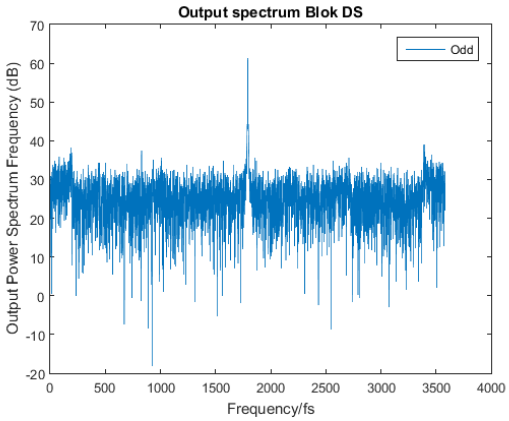


OSR 8

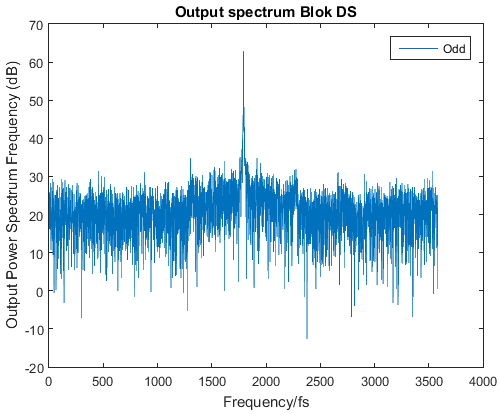


OSR 16

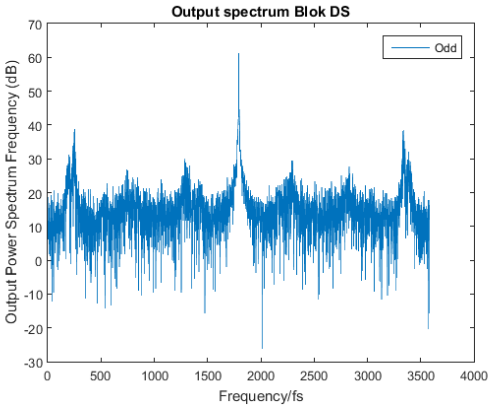
b. **Frekuensi 3,5 GHz**



OSR 4



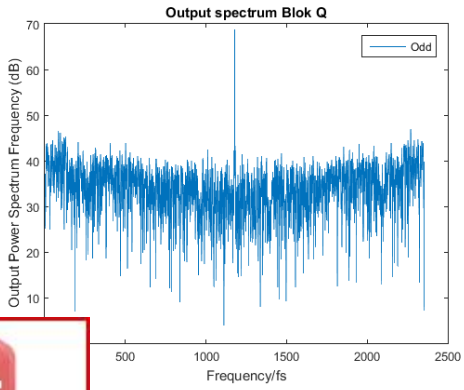
OSR 8



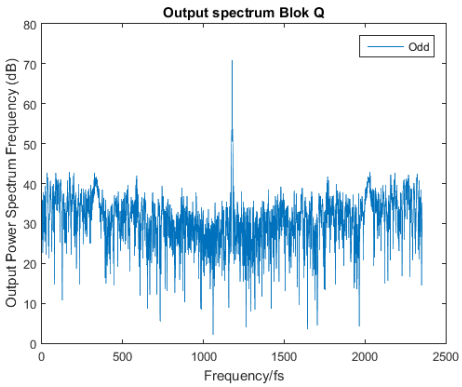
OSR 16

Ploting Keluaran Blok Quantisasi

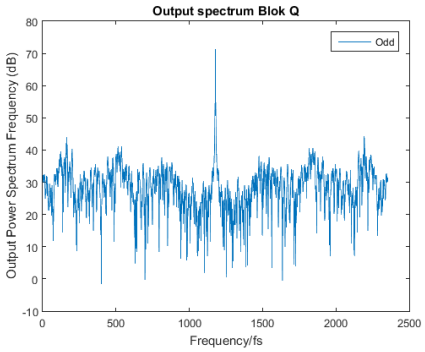
a. **Frekuensi 2,3 GHz**



OSR 4

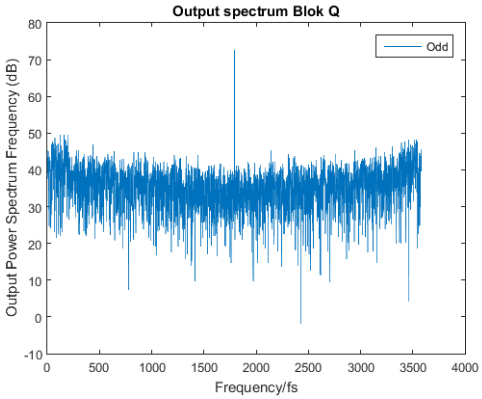


OSR 8

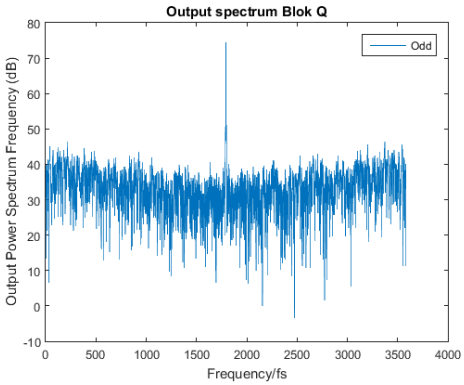


OSR 16

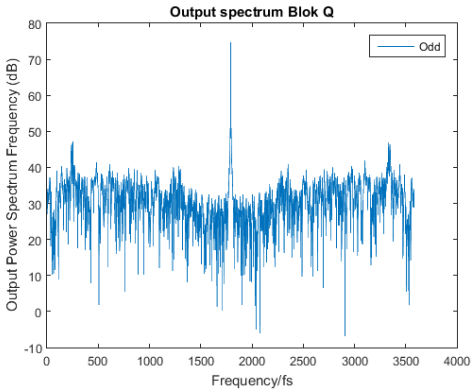
b. Frekuensi 3,5 GHz



OSR 4



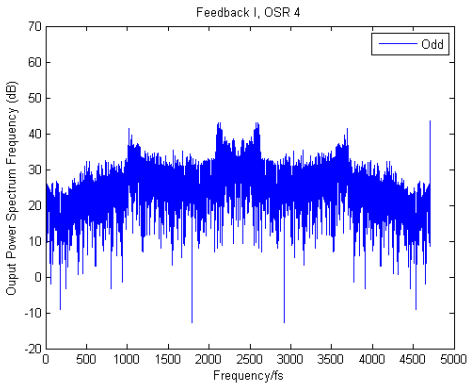
OSR 8

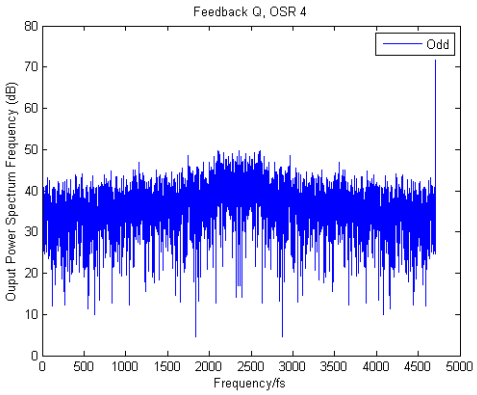


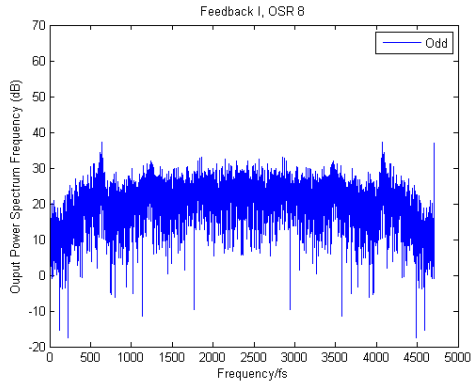
OSR 16

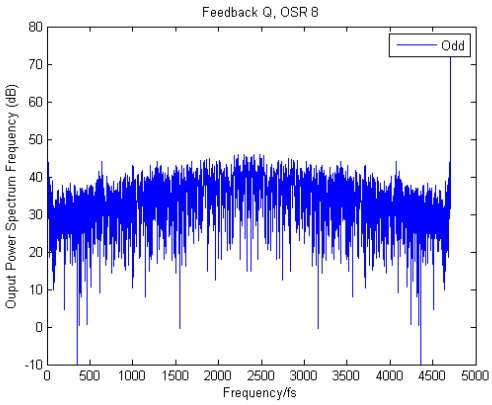
Ploting Keluaran Feedback I dan Q ($\acute{y}_i$ dan $\acute{y}q$)

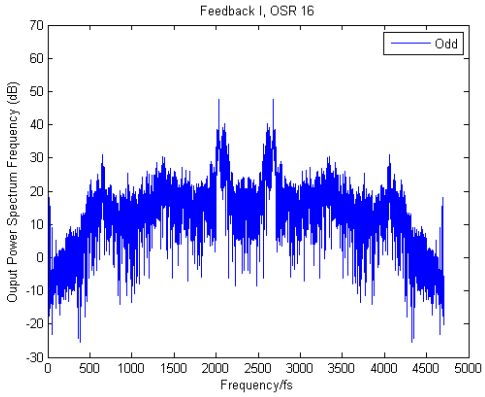
- **Frekuensi 2,3 GHz**

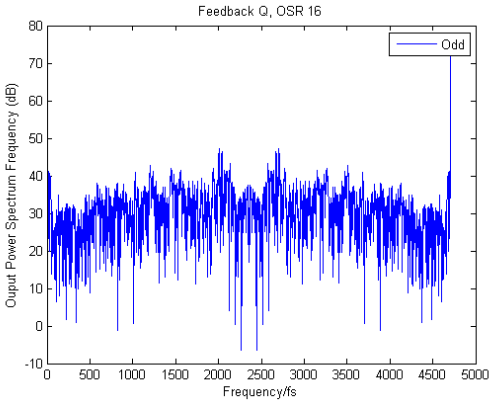




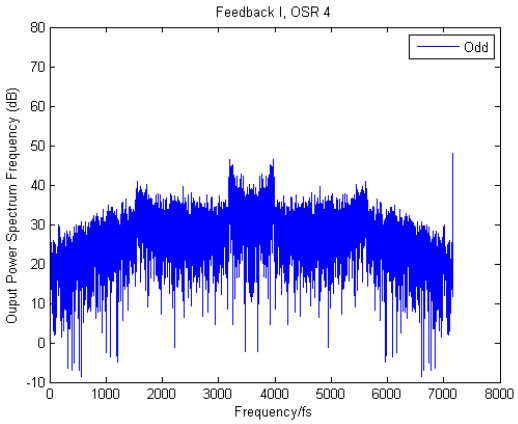


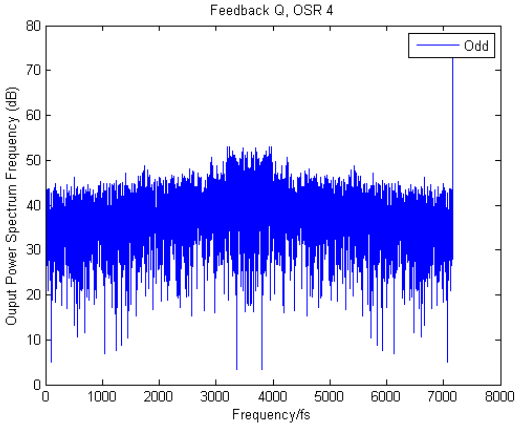


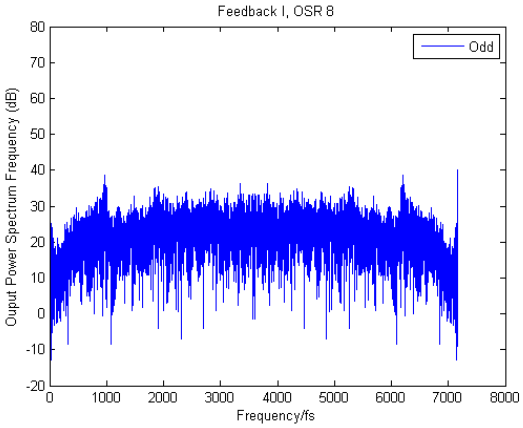


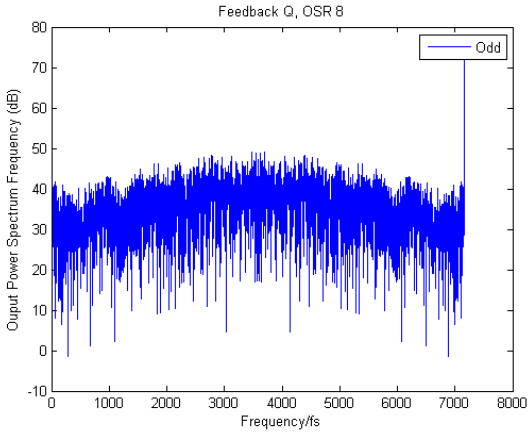


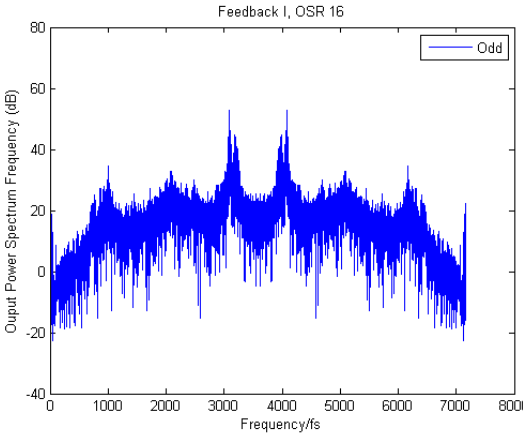
- **Frekuensi 3,5 GHz**

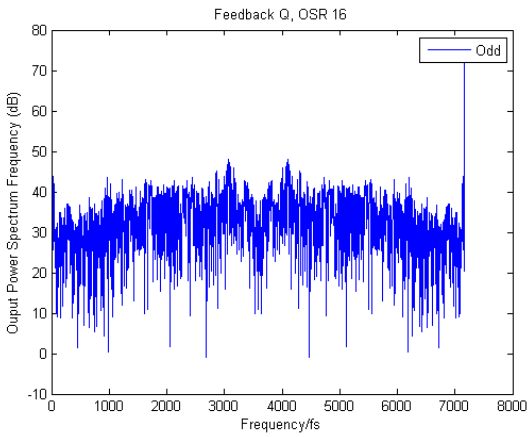








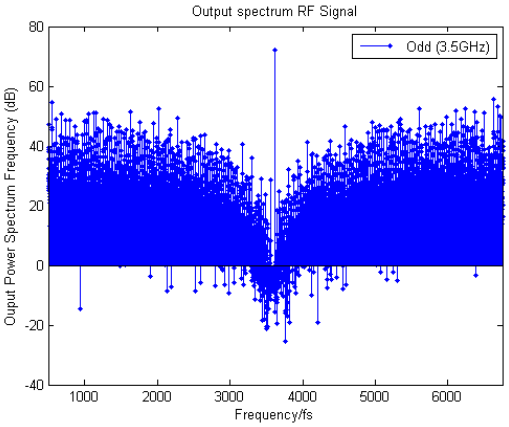


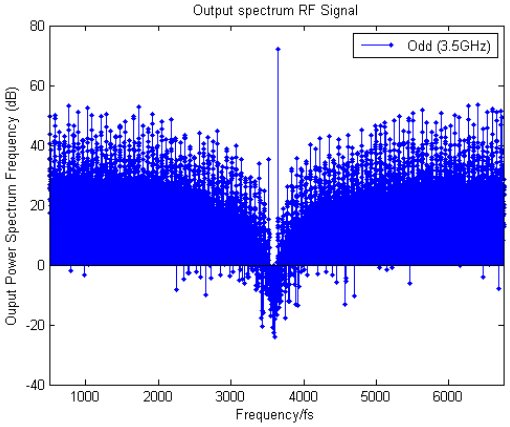


Setting frekuensi offset 32 dan 64 MHz

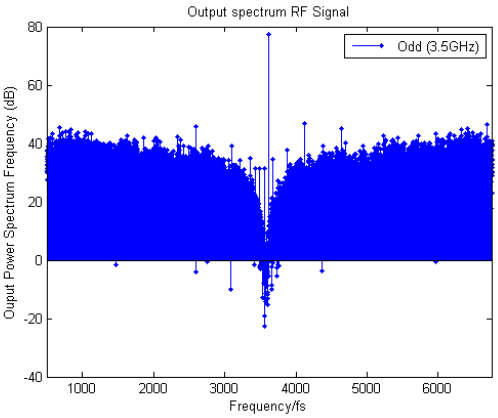
- **Frekuensi 3,5 GHz**

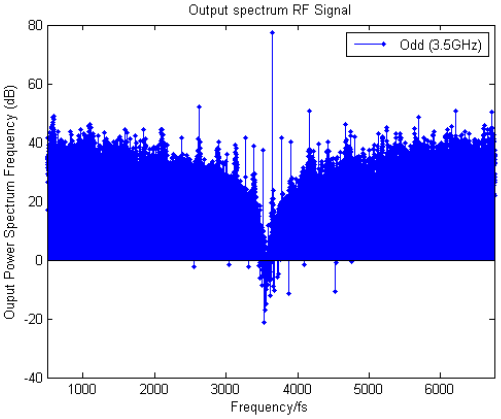
OSR 4



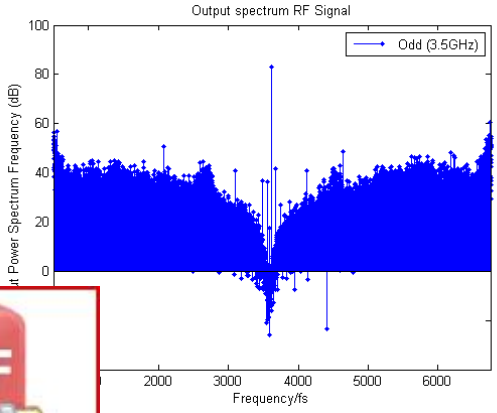


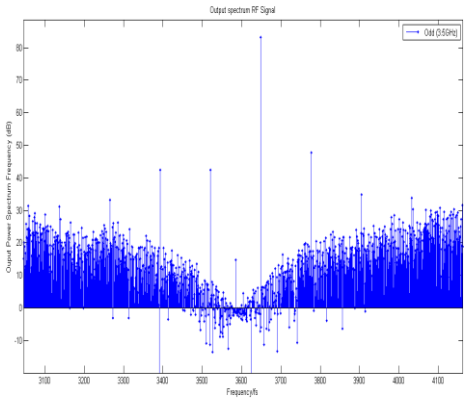
OSR 8





OSR 16

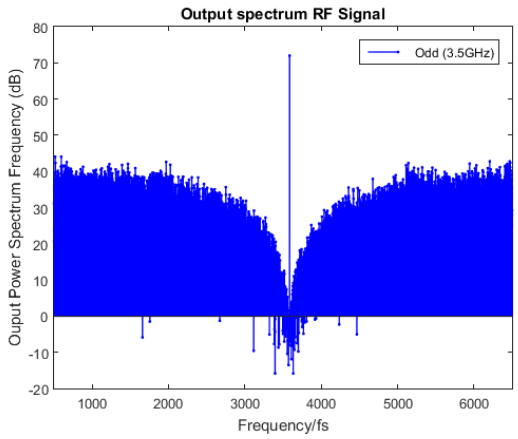




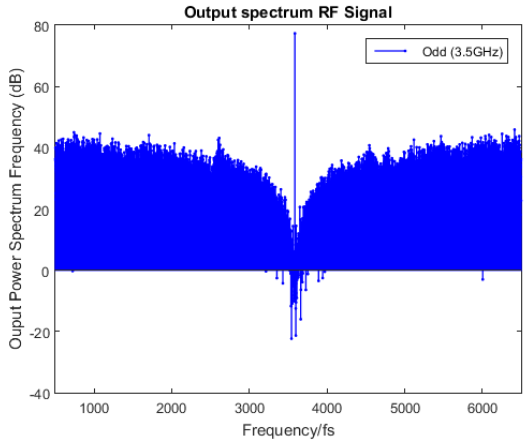
SIMULASI SKEMA PENELITIAN MENGGUNAKAN DS MOD 2 (TAMBAHAN)

- **Frekuensi 3,5 GHz**

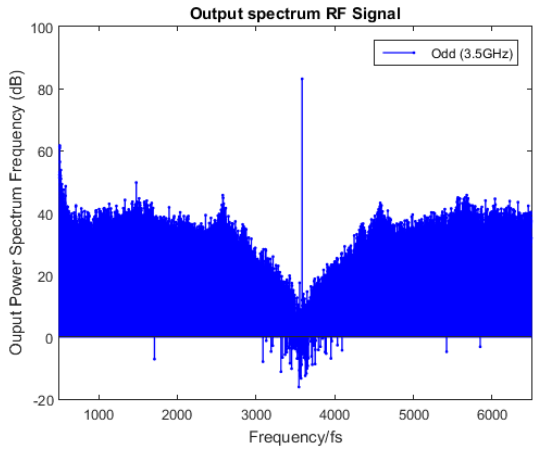
OSR 4



OSR 8



OSR 16



Syntax MATLAB

(Output Sinyal Digital Delta Modulasi)

```
close all
clear all
clc

T_sample = 100; %N_sample
Amp = 1;    % input 1 atau 0,5
un=Amp*sin(2*pi*1/T_sample*[0:(T_sample)]); %input signal
en=0;
en_delay=0;
delta=2;

      L = length(un);
      Delta_step = 0.1;   % input 0,05 atau 0,1
      xn = 0;
      for i=1:L;
      if un(i) >= xn(i)
      D(i) = 1;
      xn(i+1) = xn(i) + Delta_step;
      else
      D(i) = -1;
      xn(i+1) = xn(i) - Delta_step;
      end
      end

      for i=1:D;
      if D(i) >= xn(i)
      D(i) = 1;
      xn(i+1) = xn(i) + Delta_step;
      else
      D(i) = -1;
      xn(i+1) = xn(i) - Delta_step;
      end
      end

figure ()
plot (un, '--');
%grid on;
hold on;
stairs (xn, 'r');
hold on;
stairs (D, 'k'); %Delta Mod
legend ('signal input', 'Integrated signal','DeltaMod quantized
signal output');
('Sample Number')
('Amplitude')
'Delta Modulation')
0 100]);
-1 1]);
```

(Output Spectrum Delta Modulasi)

```
close all
clear all
clc

T_sample = 100; %N_sample
Amp = 1;
un=Amp*sin(2*pi*1/T_sample*[0:(T_sample)]); %input signal
en=0;
en_delay=0;
delta=2;

      L = length(un);
      Delta_step = 0.1;
      xn = 0;
      for i=1:L;
            if un(i) >= xn(i)
            D(i) = 1;
            xn(i+1) = xn(i) + Delta_step;
      else
            D(i) = -1;
            xn(i+1) = xn(i) - Delta_step;
      end
      end

      for i=1:D;
            if D(i) >= xn(i)
            D(i) = 1;
            xn(i+1) = xn(i) + Delta_step;
      else
            D(i) = -1;
            xn(i+1) = xn(i) - Delta_step;
      end
      end

un1_fft_DM = 20*log10(abs(fft(D)));
un1_fft_fftshift_DM =fftshift(un1_fft_DM);
F= [-T_sample/2 : T_sample/2]/T_sample;

figure()
plot(F,un1_fft_fftshift_DM, 'r')
xlabel('Frequency/fs')
ylabel('Ouput to Input Power Spectrum Frequency (dB)')
title ('Output spectrum of Delta Modulation')
grid on
```

(Output Sinyal Digital Delta-Sigma Modulasi)

```
close all
clear all
clc

T_sample = 100; %N_sample
Amp = 1;
un=Amp*sin(2*pi*1/T_sample*[0:(T_sample)]); %input signal
en=0;
en_delay=0;
delta=2;

for aa=1:length(un)
    yn=un(aa) + en_delay;

            if yn>=0  % imilar with %vn=sgn(yn); as quantizing
            %if yn>0  % imilar with %vn=sgn(yn); as quantizing
            vn=1;
            else
            vn=-1;
            %vn=0;
            end
     en = yn - vn;
     en_delay=en;

     output_vn(aa)=vn;
end

figure()
t=0:T_sample;
plot(t, un(t+1),'--')
hold on
grid on
stairs(t, output_vn(t+1),'r')
legend ('signal input', 'DSMod quantized signal output');
xlabel ('Sample Number')
ylabel ('Amplitude')
title ('Delta-Sigma Modulation')
```

(Output Spectrum Delta-Sigma Modulasi)

```
close all
clear all

e = 100; %N_sample
sin(2*pi*1/T_sample*[0:(T_sample)]); %input signal
```

```
en=0;
en_delay=0;
delta=2;

for aa=1:length(un)
    yn=un(aa) + en_delay;

            if yn>=0  % imilar with %vn=sgn(yn); as quantizing
            %if yn>0  % imilar with %vn=sgn(yn); as quantizing
            vn=1;
            else
            vn=-1;
            %vn=0;
            end
     en = yn - vn;
     en_delay=en;

     output_vn(aa)=vn;
end
diff = (un-output_vn);
average_power= vn ./ T_sample;

un1_fft = 20*log10(abs(fft(output_vn)));
un1_fft_fftshift =fftshift(un1_fft);
F= [-T_sample/2 : T_sample/2]/T_sample;

figure()
plot(F,un1_fft_fftshift)
xlabel('Frequency/fs')
ylabel('Ouput to Input Power Spectrum Frequency (dB)')
title ('Output spectrum of Delta-Sigma MOD1')
grid on
```

(Noise Transfer Function)

```
close all;
clear all;
clc;

%% generate input
ffc= 2.3552*10^9; %generate input frekuenesi carrier choose : 2.3
Ghz, 3.5 Ghz
ffs= 4.710*10^9; %Samplng Frequency
BBW= 36.8*10^6; % kenapa 16 MHz?

N=50;
OSR = 4;

 1/(OSR*N) : 1;

:length(fc);

(i) = 2.*sin(pi.*fc(i))^2;
```

```
end

% NTF_DS_out
figure()
plot (20.*log10 (abs(NTFMOD1))), 'k';

figure()
plot ( (abs(NTFMOD1))), 'r--';
```

(Deret Fourier Transform – Penentuan R (OSR 4,8,dan 16)

```
% DFT Np4
odd4nol = [0,0,0,0];
odd4pos = [1,1,1,1];
odd4neg = [-1,-1,-1,-1];

odd_c0  = max(abs(fft([0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,]))/256)*2


odd_c1  = max(abs(fft([1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     -1,-1,-1,-1,
-1,-1,-1,-1,           -1,-1,-1,-1, -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,           -1,-1,-1,-1, -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,           -1,-1,-1,-1, -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,           -1,-1,-1,-1, -1,-1,-1,-1, 0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,
       ,]))/256)*2
```

```
% DFT Np8
odd4nol = [0,0,0,0];
odd4pos = [1,1,1,1];
odd4neg = [-1,-1,-1,-1];

odd_c0  = max(abs(fft([0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,])/256)*2


odd_c1  = max(abs(fft([0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
1,1,1,1,  1,1,1,1,      1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,  1,1,1,1,      1,1,1,1,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      -1,-1,-1,-1, -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1, -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,            0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,])/256)*2


odd_c3  = max(abs(fft([1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,  1,1,1,1,      1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,  1,1,1,1,      1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,  1,1,1,1,      1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,  1,1,1,1,      1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1, -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1, -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1, -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1, -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1, -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,            -1,-1,-1,-1, -1,-1,-1,-1, 0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,])/256)*2
```

```
% DFT Np16
odd4nol = [0,0,0,0];
odd4pos = [1,1,1,1];
odd4neg = [-1,-1,-1,-1];

odd_c0  = max(abs(fft([0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,]))/256)*2


odd_c1  = max(abs(fft([1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0       0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, 0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,]))/256)*2


odd_c3  = max(abs(fft([1,1,1,1,      1,1,1,1,      1,1,1,1,

1,1,1,1,  1,1,1,1,      1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,  1,1,1,1,      1,1,1,1,      1,1,1,1,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,  0,0,0,0,      0,0,0,0,      0,0,0,0,      -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
       1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, 0,0,0,0,
       ,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
       ,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
       ,  0,0,0,0,      0,0,0,0,      0,0,0,0,      0,0,0,0,
       ,  0,0,0,0,      0,0,0,0,      0,0,0,0,]))/256)*2
```

```
odd_c5  = max(abs(fft([1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, 0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     0,0,0,0,
0,0,0,0,]))/256)*2


odd_c7  = max(abs(fft([1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,     1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,     1,1,1,1,     1,1,1,1,
0,0,0,0,  0,0,0,0,     0,0,0,0,     0,0,0,0,     -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,  -1,-1,-1,-1, 0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,]))/256)*2

% DFT Np32
odd4nol = [0,0,0,0];
odd4pos = [1,1,1,1];
odd4neg = [-1,-1,-1,-1];

odd_c0  = max(abs(fft([0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0 0 0 0    0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
       ,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
       ,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
       ,]))/256)*2
```

```
odd_c3  = max(abs(fft([1,1,1,1,      1,1,1,1,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      -1,-1,-1,-1,
-1,-1,-1,-1,           0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,])))/256)*2
```

```
odd_c3  = max(abs(fft([1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,           0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,])))/256)*2
```

```
odd_c5  = max(abs(fft([1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,   1,1,1,1,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,           0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0       0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,])))/256)*2
```

```
odd_c7  = max(abs(fft([1,1,1,1,      1,1,1,1,      1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,      1,1,1,1,      1,1,1,1,
           1,1,1,1,    1,1,1,1,      1,1,1,1,      1,1,1,1,
           0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
           0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
           0,0,0,0,    0,0,0,0,      0,0,0,0,      0,0,0,0,
           0,0,0,0,    0,0,0,0,      0,0,0,0,      -1,-1,-1,-1,
1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
```

```
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,           0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,
0,0,0,0,]))/256)*2


odd_c9  = max(abs(fft([1,1,1,1,      1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,      1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,      1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,      1,1,1,1,     1,1,1,1,
0,0,0,0,  0,0,0,0,     0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,  0,0,0,0,     0,0,0,0,      0,0,0,0,     -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,           0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,]))/256)*2


odd_c11 = max(abs(fft([1,1,1,1,      1,1,1,1,     1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,      1,1,1,1,     1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,      1,1,1,1,     1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,      1,1,1,1,     1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,      1,1,1,1,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,      0,0,0,0,     -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,           0,0,0,0,       0,0,0,0,    0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,       0,0,0,0,    0,0,0,0,
0,0,0,0,   0,0,0,0,]))/256)*2
odd_c13 = max(abs(fft([1,1,1,1,       1,1,1,1,     1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,       1,1,1,1,     1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,       1,1,1,1,     1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,       1,1,1,1,     1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,       1,1,1,1,     1,1,1,1,
1,1,1,1,   1,1,1,1,    1,1,1,1,       0,0,0,0,     0,0,0,0,
0,0,0,0,   0,0,0,0,    0,0,0,0,       0,0,0,0,    -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
1,-1,                 -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
1,-1,                 -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
1,-1,                 -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
1,-1,                 -1,-1,-1,-1,   -1,-1,-1,-1, -1,-1,-1,-1,
1,-1,                  0,0,0,0,       0,0,0,0,     0,0,0,0,
   0,0,0,0,    0,0,0,0,]))/256)*2
```

```
odd_c15 = max(abs(fft([1,1,1,1,        1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,        1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,        1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,        1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,        1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     1,1,1,1,        1,1,1,1,     1,1,1,1,
1,1,1,1,  1,1,1,1,     0,0,0,0,        0,0,0,0,     -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,    -1,-1,-1,-1,  -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,    -1,-1,-1,-1,  -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,    -1,-1,-1,-1,  -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,    -1,-1,-1,-1,  -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,    -1,-1,-1,-1,  -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,    -1,-1,-1,-1,  -1,-1,-1,-1,
-1,-1,-1,-1,          -1,-1,-1,-1,    -1,-1,-1,-1,  -1,-1,-1,-1,
-1,-1,-1,-1,           0,0,0,0,        0,0,0,0,]))/256)*2
```

(Function Theta, Quantisasi Odd OSR 4,8,16)

```
% odd
% steps_in_pi=2, NA=2, NP=OSR=4.
function [odd_r_pwm, odd_R, odd_theta]=
oddquan4(output_cp_yr,output_cp_ytheta);
%NA=2;
if (output_cp_yr <0.4502);
    odd_R=0;
    odd_r_pwm=0/2;

else
    odd_R=0.9003;
    odd_r_pwm=1/2;

end

%% NP=4;
if (output_cp_ytheta >= (0/32*2*pi))& (output_cp_ytheta
                   2*pi));
                   _theta=(4/32*2*pi);
                   (output_cp_ytheta >= (8/32*2*pi))& (output_cp_ytheta <=
                   2*pi));
                   _theta=(12/32*2*pi);
```

```
elseif (output_cp_ytheta >= (-16/32*2*pi))& (output_cp_ytheta < (-
8/32*2*pi));
    odd_theta=(-12/32*2*pi);
else (output_cp_ytheta >= (-8/32*2*pi))&(output_cp_ytheta <
(0/32*2*pi));
    odd_theta=(-4/32*2*pi);
end

end




% odd
% steps_in_pi=4, NA=3, NP=OSR=8.
function [odd_r_pwm, odd_R, odd_theta]= oddquan8(output_cp_yr,
output_cp_ytheta);
%NA=3;
if (output_cp_yr < 0.2437);
    odd_R=0;
    odd_r_pwm=0/4;

elseif (output_cp_yr >= 0.2437)&(output_cp_yr < 0.8318);
    odd_R=0.4873;
    odd_r_pwm=1/4;

else
    odd_R=1.1763;
    odd_r_pwm=3/4;

end

%% NP=8;
if (output_cp_ytheta >= (0/32*2*pi))& (output_cp_ytheta
<(4/32*2*pi));
    odd_theta=(2/32*2*pi);
elseif (output_cp_ytheta >= (4/32*2*pi))& (output_cp_ytheta <
(8/32*2*pi));
    odd_theta=(6/32*2*pi);
elseif (output_cp_ytheta >= (8/32*2*pi))& (output_cp_ytheta <
(12/32*2*pi));
    odd_theta=(10/32*2*pi);
elseif (output_cp_ytheta >= (12/32*2*pi))&(output_cp_ytheta <=
2*pi));
    _theta=(14/32*2*pi);
(output_cp_ytheta >= (-16/32*2*pi))& (output_cp_ytheta < (-
*pi));
_theta=(-14/32*2*pi);
```

```
elseif (output_cp_ytheta >= (-12/32*2*pi))& (output_cp_ytheta < (-
8/32*2*pi));
    odd_theta=(-10/32*2*pi);
elseif (output_cp_ytheta >= (-8/32*2*pi))& (output_cp_ytheta < (-
4/32*2*pi));
    odd_theta=(-6/32*2*pi);
else (output_cp_ytheta >= (-4/32*2*pi))& (output_cp_ytheta <
(0/32*2*pi));
    odd_theta=(-2/32*2*pi);
end

end




% odd
% steps_in_pi=8, NA=5, NP=OSR=16.
function [odd_r_pwm, odd_R, odd_theta]= oddquan16(output_cp_yr,
output_cp_ytheta);
%NA=5;
if (output_cp_yr < 0.1242);
    odd_R=0;
    odd_r_pwm=0/8;

elseif (output_cp_yr >= 0.1242)&(output_cp_yr < 0.4779);
    odd_R=0.2484;
    odd_r_pwm=1/8;

elseif (output_cp_yr >= 0.4779)&(output_cp_yr < 0.8831);
    odd_R=0.7074;
    odd_r_pwm=3/8;

elseif (output_cp_yr >= 0.8831)&(output_cp_yr < 1.1538);
    odd_R=1.0587;
    odd_r_pwm=5/8;

else
    odd_R=1.2488;
    odd_r_pwm=7/8;

end

5
s in fractional
put_cp_ytheta >= (0/32*2*pi))& (output_cp_ytheta <
*pi));
_theta=(1/32*2*pi);
```

```
elseif (output_cp_ytheta >= (2/32*2*pi))& (output_cp_ytheta <
(4/32*2*pi));
    odd_theta=(3/32*2*pi);
elseif (output_cp_ytheta >= (4/32*2*pi))& (output_cp_ytheta <
(6/32*2*pi));
    odd_theta=(5/32*2*pi);
elseif (output_cp_ytheta >= (6/32*2*pi)) & (output_cp_ytheta <
(8/32*2*pi));
    odd_theta=(7/32*2*pi);
elseif (output_cp_ytheta >= (8/32*2*pi))& (output_cp_ytheta <
(10/32*2*pi));
    odd_theta=(9/32*2*pi);
elseif (output_cp_ytheta >= (10/32*2*pi))& (output_cp_ytheta <
(12/32*2*pi));
    odd_theta=(11/32*2*pi);
elseif (output_cp_ytheta >= (12/32*2*pi))& (output_cp_ytheta <
(14/32*2*pi));
    odd_theta=(13/32*2*pi);
elseif (output_cp_ytheta >= (14/32*2*pi))& (output_cp_ytheta <=
(16/32*2*pi));
    odd_theta=(15/32*2*pi);
elseif (output_cp_ytheta >= (-16/32*2*pi))& (output_cp_ytheta < (-
14/32*2*pi));
    odd_theta=(-15/32*2*pi);
elseif (output_cp_ytheta >= (-14/32*2*pi))& (output_cp_ytheta < (-
12/32*2*pi));
    odd_theta=(-13/32*2*pi);
elseif (output_cp_ytheta >= (-12/32*2*pi))& (output_cp_ytheta < (-
10/32*2*pi));
    odd_theta=(-11/32*2*pi);
elseif (output_cp_ytheta >= (-10/32*2*pi))& (output_cp_ytheta < (-
8/32*2*pi));
    odd_theta=(-9/32*2*pi);
elseif (output_cp_ytheta >= (-8/32*2*pi))& (output_cp_ytheta < (-
6/32*2*pi));
    odd_theta=(-7/32*2*pi);
elseif (output_cp_ytheta >= (-6/32*2*pi))& (output_cp_ytheta < (-
4/32*2*pi));
    odd_theta=(-5/32*2*pi);
elseif (output_cp_ytheta >= (-4/32*2*pi))& (output_cp_ytheta < (-
2/32*2*pi));
    odd_theta=(-3/32*2*pi);
else (output_cp_ytheta >= (-2/32*2*pi))& (output_cp_ytheta <
(0/32*2*pi));
    _theta=(-1/32*2*pi);
```

To Polar

```
output_cp_yr = sqrt((vn_i ).^2 + (vn_q).^2);
output_cp_ytheta = atan2 (vn_q,vn_i);
```

Polar To Cartesian

```
en_delay_i = odd_R .*cos(odd_theta);
en_delay_q = odd_R .*sin(odd_theta);
```

Quantisation Odd

```
if      (steps_in_pi == 2)
            [odd_r_pwm, odd_R, odd_theta]= oddquan4(output_cp_yr,
            output_cp_ytheta);
elseif (steps_in_pi == 4)
            [odd_r_pwm, odd_R, odd_theta]= oddquan8(output_cp_yr,
            output_cp_ytheta);
elseif (steps_in_pi == 8)
            [odd_r_pwm, odd_R, odd_theta]= oddquan16(output_cp_yr,
            output_cp_ytheta);
end

R_feedback = odd_R;
theta_feedback = odd_theta;

R_feedback_array (i)        = R_feedback;      %feedback ke DS
theta_feedback_array (i)    = theta_feedback;  %feedback ke DS

odd_R_array (i)             = odd_R;
odd_theta_array (i)         = odd_theta;
odd_r_pwm_array (i)         = odd_r_pwm;

output_cp_yr_array (i)      = output_cp_yr;
output_cp_ytheta_array(i)   = output_cp_ytheta;
```

Keseluruhan Code Pemrograman (ODD MATLAB)

```
close all;
clear all;
clc;

%% generate input
fc= 2.355*10^9; %geerate input frekuenesi carrier 5G , 2355.2 MHz
atau 3584
fs= 4.710*10^9; %Samplng Frequency 4710.4 atau 7168
BW= 36.8*10^6; % fc/64   untuk 3,5 GHz = 56 MHz

steps_in_pi =2;  %choose: 2,4,8
NP=2*steps_in_pi; %OSR_r=RF = 4,8,16    cari rumus OSR_RF
fs_rf= NP*fs;

T_sample =4710; %N_sample = OSR   = fs
No_of_periods=1;
sample_max = (T_sample*No_of_periods);
offset=1; %Offset signal from carrier  choose: 1,.....

noise = (randn(1,1)+j*(randn(1,1)))*10^-4;
%noise= [-0.000120748692268504 + 7.17238651328839e-05i]; %test
only

max_num= offset/No_of_periods;
G=1;

% Baseband proses
for jarak=1:max_num;
i=1:sample_max;

p=1;
Amp(p)=0.5;
input_un=Amp(p)*exp(j*2*pi*i/T_sample*jarak)+ noise;  %Input
signal


%% input I dan Q dalam format Cartesian/Rectangular
input_i = real(input_un);
input_q = imag(input_un);

%% Modulasi Delta Sigma Orde 1

yn_delay_i=0;
en_delay_i=0;
yn_delay_q=0;
en_delay_q=0;

i=1:sample_max;


i = input_i(i)- en_delay_i;
i = yn_i + yn_delay_i;
```

```
    yn_delay_i = vn_i;

    yn_q  = input_q(i)- en_delay_q;
    vn_q = yn_q + yn_delay_q;
    yn_delay_q = vn_q;

    yn_i_array(i)        = yn_i; %Input signal of I-plane
    yn_q_array(i)        = yn_q; %Input signal of Q-plane

    en_delay_i_array(i)  = en_delay_i;
    en_delay_q_array(i)  = en_delay_q;

    vn_i_array (i)       = vn_i;
    vn_q_array (i)       = vn_q;

%% Cartesian to Polar
    output_cp_yr = sqrt((vn_i ).^2 + (vn_q).^2);
    output_cp_ytheta = atan2 (vn_q,vn_i);

%% Quantization: ODD

     if      (steps_in_pi == 2)
             [odd_r_pwm, odd_R, odd_theta]= oddquan4(output_cp_yr,
output_cp_ytheta);
     elseif (steps_in_pi == 4)
             [odd_r_pwm, odd_R, odd_theta]= oddquan8(output_cp_yr,
output_cp_ytheta);
     elseif (steps_in_pi == 8)
             [odd_r_pwm, odd_R, odd_theta]= oddquan16(output_cp_yr,
output_cp_ytheta);
     end

     R_feedback = odd_R;
     theta_feedback = odd_theta;

     R_feedback_array (i)        = R_feedback;           %feedback
ke DS
     theta_feedback_array (i)    = theta_feedback;       %feedback
ke DS

     odd_R_array (i)             = odd_R;
     odd_theta_array (i)         = odd_theta;
     odd_r_pwm_array (i)         = odd_r_pwm;

     output_cp_yr_array (i)      = output_cp_yr;
     output_cp_ytheta_array(i)   = output_cp_ytheta;

%% Polar to Cartesian
     n_delay_i = odd_R .*cos(odd_theta);
     n_delay_q = odd_R .*sin(odd_theta);

     PWM/PPM
     SIRMARF (odd_r_pwm_array,odd_theta_array,steps_in_pi);
```

```
%  FIGURE RESULT
figure();
stem(20*log10(abs(fft(pwm))),'.b','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Output spectrum RF Signal');
legend ('Odd (2.3GHz)');
xlim([500 4200]);
% ylim([0 100]);


figure();
plot(20*log10(abs(fft(pwm))),'b','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Output spectrum RF Signal');
legend ('Odd');
xlim([500 1600]);

figure();
plot(20*log10(abs(fft(vn_i_array))),'r','LineWidth',1)
plot(20*log10(abs(fft(vn_q_array))),'b','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Output spectrum Blok DS');
legend ('Odd');

figure();
plot(20*log10(abs(fft(output_cp_yr_array))),'b','LineWidth',1)
plot(20*log10(abs(fft(output_cp_ytheta_array))),'b','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Output spectrum Blok Q');
legend ('Odd');


%output-filter DS
fft_DS_out=fft(vn_i_array(sample_max/2+1:sample_max)+j.*
vn_q_array (sample_max/2+1:sample_max));
abs_DS_out=abs(fft_DS_out);
spectrum_DS_out=20*log10(abs_DS_out);
figure()
plot (fftshift(spectrum_DS_out));
xlabel('Frequency/fs');
ylabel('Output Power Spectrum Frequency (dB)');
title ('Output spectrum Blok DS');
legend ('Odd')

-Q
ut=fft(output_cp_yr_array(sample_max/2+1:sample_max)+j.*out
ytheta_array(sample_max/2+1:sample_max));
ut=abs(fft_Q_out);
n_Q_out=20*log10(abs_Q_out);
```

```
figure()
plot (fftshift(spectrum_Q_out));
xlabel('Frequency/fs');
ylabel('Output Power Spectrum Frequency (dB)');
title ('Output spectrum Blok Q');
legend ('Odd')

%figure input I dan Q
figure();
plot(20*log10(abs(fft(yn_i_array))),'b','LineWidth',1)
plot(20*log10(abs(fft(yn_q_array))),'b','LineWidth',1)
xlabel('Frequency/fs');
ylabel('Ouput Power Spectrum Frequency (dB)');
title ('Input I');
legend ('Odd');
% xlim([500 1600]);
```