APPENDIX A

```
%=========================================================
================
% The mfile investigates the effects of high power
amplifier on the ofdam
% signals. The effects bit error rat versus signal to noise
ratio for (TWTA&SSPA)
%=========================================================
================
% Initialization
    clear all;
    close all;
    clc;
%=========================================================
================
% Setting Parameters
%=================================================
close all
clear all
clc
%_____________________-%
% Initialization
%_____________________-%
drops = 1000; % Number of independent runs.
M = 16; % M-QAM modulation.
Number_of_subcarriers = 64; % Number of the OFDM subcarrier
Length_of_CP = 16; % the CP length
Number_of_bits =10000; % Number of bits
g_t =0.05; % I/Q imbalance factor of amplitude at Tx
fi_t = pi/36; % I/Q imbalance factor of phase at Tx
g_r =0.05; % I/Q imbalance factor of amplitude at Rx
fi_r = pi/36; % I/Q imbalance factor of phase at Rx
SNR_dB=-5:35; % channel SNR
A_sat=3; % input saturation voltage
Es=1;
T =4; % numbers of channel tap
variance_of_pn = 0.04; % variance of the phase noise
%_____________________-%
% Transmitter
%_____________________-%
% QAM modulation%
x = floor(rand(1,Number_of_bits)+0.5);
[x_qam,data] = qam(x,M);
% generate the OFDM signal%
x_ofdm = ofdm(x_qam,Number_of_subcarriers,Length_of_CP)*Es;
for jj=1:drops
for kk=1:length(SNR_dB)
snr(kk) = 10.^ (SNR_dB(kk)10);
% ___IQ Imbalance at the transmitter__-%
[x_ofdm_im,k1_t,k2_t]=iqimbalance(x_ofdm,g_t,fi_t);
%_____________________-%
% PA model with the memoryless e_ect
```

```
%_____________________-%
[x_ofdm_tx,OBO(kk),sigmaD(kk),k_o(kk)] = pa(x_ofdm_im,A_sat);
%_____________________-%
% no channel e_ect
%_____________________-%
% channel_output=x_ofdm_tx; sigmaW(kk)=0;
%_____________________-%
% AWGN channel
%_____________________-%
channel_output = awgn(x_ofdm_tx,SNR_dB(kk),'measured');
sigmaW(kk) = mean(abs(x_ofdm_tx).^ 2)snr(kk);
%_____________________-%
% Rayleigh Fading Channel
%_____________________-%
% [channel_output,h_channel] = rayleighfading(x_ofdm_tx,T,snr(kk));
% sigmaW(kk) = mean(abs(x_ofdm_tx)._ 2)snr(kk);
%_____________________-%
% Receiver
%_____________________-%
% ______I/Q Imbalance at the receiver_____%
[rx_im,k1_r,k2_r]=iqimbalance(channel_output,g_r,fi_r);
%_____Phase noise__________%
N = Number_of_subcarriers;
L = Length_of_CP;
v_pn = variance_of_pn;
[rx_phasenoise,w_angle] = phasenoise(rx_im,N,L,v_pn);
rx= rx_phasenoise;
% Match _lter for Rayleigh fading channel%
% H = _t(h_channel,length(rx_phasenoise));
% rx = i_t(_t(rx_phasenoise).H);
% rx = rx(1:length(x_ofdm_im));
%Demodulation of the OFDM signal%
z = de_ofdm(rx,Number_of_subcarriers,Length_of_CP);
z = z(1:length(x_qam))Es;
%_______-Evaluation_________-%
% demodulate the QAM signal%
y = qamdemod(z,M);
y_bi = de2bi(y);
y_bi = fliplr(y_bi);
[m,n] = size(data);
y_final = [];
for ii=1:m
y_final = [y_final,y_bi(ii,:)];
end
% The bit error probability %
[num(jj,kk),BER(jj,kk)]= symerr( y_final,x(1:length(y_final)));
end
end
ber_whole = mean(BER);
%____________qam.m_____________%
function [x_qam,data]=qam(x,M)
L_QAM=log2(M);
for ii=1:length(x)L_QAM
x1(ii,:)=x((ii-1)*L_QAM+1:(ii-1)*L_QAM+L_QAM);
x2(ii,:)=num2str(x1(ii,:));
end
```

```
data=bin2dec(x2)';
x_qam = qammod(data,M);
%____________ofdm.m_____________%
function [ x_ofdm] = ofdm(data,N,L )
%% This is OFDM modulation. CP is inserted.%%
c_prefix = [];
m_data = [];
T=N+L; %length of OFDM symbol
I=length(data);
if rem(I,M) =0
data = [data,zeros(1,M-rem(I,N))];
I = length(data);
end
% the modulation IFFT %
for n =1:N:I-N+1
ofdm_temp1(n:n+M-1) = sqrt(N)*i_t(data(n:n+N-1),N);
temp1 = ofdm_temp1(n:n+N-1);
m_data = [m_data,zeros(1,L),temp1];
end
% the cyclic pre_x %
for n =1:N:I-N+1
ofdm_temp2(n:n+L-1) = ofdm_temp1(n+N-L:n+N-1);
end
ofdm_temp2 = [ofdm_temp2,zeros(1,I-length(ofdm_temp2))];
% inserting cyclic pre_x %
for n =1:N:I-N+1
temp2 = ofdm_temp2(n:n+L-1);
c_prefix = [c_prefix,temp2,zeros(1,N)];
end
for n = 1:T:_oor(I/N)*L+I-(T)+1
x_ofdm(n:n+T-1) = [c_prefix(n:n+L-1),m_data(n+L:n+T-1)];
end
%____________-iqimbalance.m_____________%
function [y,k1,k2] = iqimbalance(a,g,_)
% g is the amplitude imbalance of the IQ imbalance
% _ is the phase imbalance of the IQ imbalance
k1 = cos(fi) +i*g*sin(fi);
k2 = g*cos(fi) - i*sin(fi);
for ii = 1:length(a)
y(ii) = k1*a(ii)+k2*conj(a(ii));
end
%_____________pa.m_____________%
function [y,OBO,var_noise,K_abs] = pa(x_ofdm,A_sat)
amplitude = abs(x_ofdm);
angle_x = angle(x_ofdm);
Pin = mean(abs(x_ofdm).^2)2;
i = sqrt(-1);
%_________TWTA___________-%
% for ii=1:length(amplitude)
% roo = amplitude(ii);
% Fa(ii) = A_sat_ 2*(roo.(roo._ 2+A_sat_ 2)); %AM-AM
% Fp(ii) = (pi3)*(roo._ 2(roo._ 2+A_sat_ 2)); %AM-PM
% % y(ii) = Fa(ii)*exp(i*(Fp(ii)+angle_x(ii)));
% % end
% Pout= mean(abs(y)._ 2)2; %output power
% % OBO = 10*log10((A_sat2)_ 2(2*Pout));
```

```
% % K = mean(y.*conj(x_ofdm))mean(x_ofdm.*conj(x_ofdm));
% K_abs = abs(K);
% var_noise = (Pout-(K_abs_ 2)*Pin)*2Pin; %variance of noise
%_________-SSPA___________-%
Ao = A_sat/2;
for ii=1:length(amplitude)
roo = amplitude(ii);
Fa(ii) = roo./sqrt(1+(rooAo).^ 2);
Fp(ii) = 0;
y(ii) = Fa(ii)*exp(i*(Fp(ii)+angle_x(ii)));
Sp(ii)=Fa(ii)*exp(i*(Fp(ii)));
end
Pout= mean(abs(y).^ 2)/2; %output power
OBO = 10*log10((A_sat2).^ 2(2*Pout));
K = mean(y.*conj(x_ofdm))mean(x_ofdm.*conj(x_ofdm));
K_abs = abs(K);
var_noise = (Pout-(K_abs.^ 2)*Pin)*2Pin; %variance of noise
%___________rayleighfading.m_____________%
function [y,h] = rayleighfading(x,t,snr)
Es = sum(abs(x).^ 2) length(x); % power of signal
channel = sqrt(12)*(randn(1,t)+i*randn(1,t)).*[0.5 1 2 4];
sigma_n = Es snr 2; %noise power
for ii=1:t
h(ii)=channel(ii) sum(channel); %channel tap
end
y1 = conv(h,x);
noise = sqrt(sigma_n)*(randn(1,length(y1))+i*randn(1,length(y1)));
y = y1 +noise;
%___________-phasenoise.m_________________%
function [y,w] = phasenoise(x,N,L)
% phase noise is modelled as Wiener process
% generate phase noise
dt = T(N+L);
dw = zeros(1,N); % preallocate arrays ...
w= zeros(1,N); % for e_ciency
dw(1) = sqrt(dt)*randn; % _rst approximation outside the loop ...
w(1) = dw(1);
for jj = 2:N
dw(jj) = sqrt(dt)*randn; % general increment
w(jj) = w(jj-1) + dw(jj);
end
% phase noise added to the input signal
for ii = L:N+L:length(x)
for jj = 1:N
z(ii+jj) = x(ii+jj)*exp(i*w(jj));
end
end
%_____________-de_ofdm.m_____________________%
function [y] = de_ofdm(x,N,L)
T = N+L ;
y_temp1 = [];
y_temp3=[];
% remove the cyclic pre_x
I1 = length(x);
for n = 1:T:I1-T+1
y_temp1 = [y_temp1,x(n+L:n+T-1)];
```

```
end
% FFT demodulation
I2 = length(y_temp1);
for n=1:N:I2-N+1
y_temp2(n:n+N-1) = _t(y_temp1(n:n+N-1),N)sqrt(N);
end
y=y_temp2;
=======================
    % OFDM System Parameters
    N           = 256;      % length of OFDM IFFT
(16,32,64,...,2^n)
    M           = 64;       % number of QAM constellation
points (4,16,64,256)
    numOfZeros  = N/4+1;    % numOfZeros must be an odd
number and lower
                            % than N. The zero padding
operation is
                            % necessarry in practical
implementations.
    GI          = 1/4;      % Guard Interval
(1/4,1/8,1/16,...,4/N)
    BW          = 20;       % OFDM signal Band width in MHz
    numOfSym    = 100;      % number of OFDM Symbols

    % Amplifire Parameters
    satLevel = 5;   % in dB , higher than the tx out mean
of voltage

%=============================================================
================
%   Main Program
%=============================================================
================
    txData     = randint(N-numOfZeros,numOfSym,M);  % data
generation

    % QAM modulation
    txDataMod  = qammod(txData,M);

    % zeros padding
    txDataZpad = [txDataMod((N-numOfZeros+1)/2:end,:);...
                  zeros(numOfZeros,numOfSym);...
                  txDataMod(1:(N-numOfZeros+1)/2+1,:)];
        % Note : in practice zero padding operation must be
followed by
        % a standard. Usually the last part of data frame
shifts to the first
        % part of zero padded frame.
```

```
   % IFFT
    txDataZpadIfft = sqrt(N)*ifft(txDataZpad,N);

    % Guard Interval Insertion
    txDataZpadIfftGI    = [txDataZpadIfft((1-
GI)*N+1:end,:);txDataZpadIfft];

    % Amplifier Model
    txDataZpadIfftGIAbs     = abs(txDataZpadIfftGI);
% tx data amplitude

    % tx data amplitude standard deviation and mean
    txDataZpadIfftGIAbsStd  =
mean(std(txDataZpadIfftGIAbs));
    txDataZpadIfftGIAbsMean =
mean(mean(txDataZpadIfftGIAbs));

    % tx data phase in radian
    txDataZpadIfftGIAng     = angle(txDataZpadIfftGI);

    % It is imagined that the amplifier has no effect on
the phase of the
    % signal. The solid state amplifier effects on signal
phase is about 5
    % degrees. The amplifier AM/AM response is followed by
x/sqrt(1+(x/k)^2)
    txDataZpadIfftGIAbsHPA = txDataZpadIfftGIAbs ./...

sqrt(1+(txDataZpadIfftGIAbs/(txDataZpadIfftGIAbsMean*10^(sa
tLevel/10))).^2);
    % no change in the phase
    txDataZpadIfftGIAngHPA = txDataZpadIfftGIAng;

    % mean of amplitude after amplification
    txDataZpadIfftGIAbsHPAmean =
mean(mean(txDataZpadIfftGIAbsHPA));
    % standard deviation after amplification
    txDataZpadIfftGIAbsHPAStd  =
mean(std(txDataZpadIfftGIAbsHPA));


    % polar to cartesian conversion
    txDataZpadIfftGIHPA    = txDataZpadIfftGIAbsHPA.* ...
                             exp(sqrt(-1) *
txDataZpadIfftGIAngHPA);
```

```
    % receiver part
    % Guard Interval removal
    rxDataZpadIfftHPA  = txDataZpadIfftGIHPA(GI*N+1 :
N+GI*N,:);
    % FFT operation
    rxDataZpadHPA      =
1/sqrt(N)*fft(rxDataZpadIfftHPA,N);
    % zero removal and rearrangement
    rxDataModHPA       = [rxDataZpadHPA((N-(N-numOfZeros-
1)/2+1):N,:);...
                          rxDataZpadHPA(1:(N-
numOfZeros+1)/2,:)];
    % demodulation
    rxDataHPA          =
qamdemod(rxDataModHPA/mean(std(rxDataModHPA))*mean(std(txDa
taMod)),M);

%=================================================================
===============
%       statistical computation
%=================================================================
===============
    % Mean Error Rate computation
    MER       =
10*log10(mean(var(rxDataModHPA./mean(std(rxDataModHPA))...
                - txDataMod./mean(std(txDataMod)))));
    % Bit Error Rate computation
    [num BER] = symerr(rxDataHPA,txData);
%=================================================================
===============
% graphical observation
%=================================================================
===============
   f1 = figure(1);
   set(f1,'color',[1 1 1]);
        subplot(2,2,1);

            spectrumFftSize = 2*N;
            % spectrum of signal befor High Power Amplifier
            txSpec  =
20*log10(mean(abs(fft(txDataZpadIfftGI(:,:)./ ...

mean(std(txDataZpadIfftGI)),spectrumFftSize)),2));
            % spectrum of signal after High Power Amplifier
            HpaSpec =
20*log10(mean(abs(fft(txDataZpadIfftGIHPA(:,:)./ ...
```

```
mean(std(txDataZpadIfftGIHPA)),spectrumFftSize)),2));
            % corresponding frequency
            Freq    = linspace(-BW/2,BW/2,length(txSpec));


plot(Freq,[txSpec(length(txSpec)/2:length(txSpec));...
                txSpec(1:(length(txSpec)/2-1))]);
            hold on
            grid on

plot(Freq,[HpaSpec(length(txSpec)/2:length(txSpec));...
                HpaSpec(1:(length(txSpec)/2-1))],'r');
            grid on;
            xlabel('representing frequency');
            ylabel('spectrum signals (first symbol)');
            title('Spectrum Effects')
            legend('Befor Amplifier','After Amplifier')

        subplot(2,2,2);
            plot(real(reshape(rxDataModHPA,1,numOfSym*(N-
numOfZeros)))/mean(std(rxDataModHPA)),...
                 imag(reshape(rxDataModHPA,1,numOfSym*(N-
numOfZeros)))/mean(std(rxDataModHPA)),'.r')
            hold on
            plot(real(reshape(txDataMod,1,numOfSym*(N-
numOfZeros)))/mean(std(txDataMod)),...
                 imag(reshape(txDataMod,1,numOfSym*(N-
numOfZeros)))/mean(std(txDataMod)),'.b')
            xlabel('I channel');
            ylabel('Q channel');
            title('signal Constelleations');
            legend('After Amplifier','Before Amplifier');

        subplot(2,2,3)
            % normalize amplitude
            txAmp = linspace(0,5,100);
            amAmp = txAmp./(1+(txAmp/satLevel).^2);
            plot(txAmp,txAmp,'b');
            hold on
            plot(txAmp,amAmp,'r');
            plot(1,1,'om');
            xlabel('Input Amplitude');
            ylabel('Output Amplitude');
            title('AM/AM response of power amplifier');
            legend('Linear Response','Amplifier
Response','Mean of OFDM Amplitude');
```

```
        subplotHandel = subplot(2,2,4);
            text(0,1 ,['Mean Error Rate     :
',num2str(MER),' dB']);
            text(0,.8,['Bit  Error Rate     :
',num2str(BER)]);

            text(0,.6,['Modulation          :
',num2str(M),' QAM']);
            text(0,.4,['IFFT Size           :
',num2str(N),' points']);
            text(0,.2,['Guard Interval Size :
',num2str(N),' points']);
            text(0,.0,['Saturation Level    :
',num2str(satLevel),' dB relative to AM Avg']);
            % setting the axes invisibale
            set(subplotHandel,'Xcolor',[1 1 1]);
            set(subplotHandel,'Ycolor',[1 1 1]);
```

matlab code for reduction PAPR In OFDM using SLM-PTS

```
clc;
clear all;
close all;
N = 128; % The number of carriers
OF = 8; % Oversampling factor
K = N*OF;
QPSK_Set = [1 -1 j -j]; % QPSK Constellation symbols
Phase_Set = [1 -1 j -j]; % Weighting
M = 4; % The number of branches in SLM method
V = 4; % The number of sub-blocks in PTS method
X1 = zeros(M,N); % Initialize the data matrix
Index1 = zeros(M,N);
X2 = zeros(1,N);
Index2 = zeros(1,N);
hwait = waitbar(0,'Please wait...'); % Creates and displays a waitbar
for i=1:4 % Generate all possible combinations of weighting factor set
in PTS method
    X(i,1:4^i) = [repmat(1,1,4^(i-1)),repmat(2,1,4^(i-
1)),repmat(3,1,4^(i-1)),repmat(4,1,4^(i-1))];
    Y = X(i,1:4^i);
    X(i,1:256) = repmat(Y,1,256/length(Y));
end
    X = X.';
    Choose = fliplr(X);
    Choose_Len = 256; % The total number of combinations or IFFT
operations in PTS method
    Max_Symbols = 1e3; % The number of generated OFDM symbols
    for nSymbol=1: Max_Symbols *10
        Index = randint(1,N,length(QPSK_Set))+1;
        X = QPSK_Set(Index(1,:)); % The QPSK modulation
        X = [X(1:N/2) zeros(1,K-N) X(N/2+1:N)]; % oversampling process
        x = ifft(X,[],2); % Signals in time domain after IFFT operation
```

```
        Signal_Power = abs(x.^2);
        Peak_Power = max(Signal_Power,[],2);
        Mean_Power = mean(Signal_Power,2);
        PAPR_Orignal(nSymbol) = 10*log10(Peak_Power./Mean_Power);
    end;
    step = Max_Symbols /100; % Set the parameters of waitbar
    for nSymbol=1:Max_Symbols
        if Max_Symbols-nSymbol<=50
            waitbar(nSymbol/ Max_Symbols,hwait,'Almost done!');
            pause(0.05);
        else PerStr=fix(nSymbol/step);
            str=['Process on going>>>',num2str(PerStr),'%'];
            waitbar(nSymbol/ Max_Symbols,hwait,str);
            pause(0.05);
        end
        %SLM
        Index1(1,:) = randint(1,N,length(QPSK_Set))+1;
        Index1(2:M,:) = randint(M-1,N,length(Phase_Set))+1;
        X1(1,:) = QPSK_Set(Index1(1,:)); % The QPSK modulation
        Phase_Rot = Phase_Set(Index1(2:M,:));
        X1(2:M,:) = repmat(X1(1,:),M-1,1).*Phase_Rot;
        X11 = [X1(:,1:N/2) zeros(M,K-N) X1(:,N/2+1:N)]; % oversampling
process
        x = ifft(X11,[],2); % Signals in time domain after IFFT
operation
        Signal_Power = abs(x.^2);
        Peak_Power = max(Signal_Power,[],2);
        Mean_Power = mean(Signal_Power,2);
        PAPR_temp = 10*log10(Peak_Power./Mean_Power);
        PAPR_SLM(nSymbol) = min(PAPR_temp);
        %PTS
        A = zeros(V,N); % Initial phase set to '0',exp(j*0)
        Index2 = randint(1,N,length(QPSK_Set))+1;
        X2 = QPSK_Set(Index2(1,:));
        Index= randperm(N);
        for v=1:V % Divided signals in frequency domain X into V non-
overlapping sub-blocks
            A(v,Index(v:V:N)) = X2(Index(v:V:N));
        end
        A1 = [A(:,1:N/2) zeros(V,K-N) A(:,N/2+1:N)];
        a = ifft(A1,[],2);
        min_value = 10; % Applying optimum algorithm
        for n=1:Choose_Len temp_phase = Phase_Set(Choose(n,:)).';
            temp_max = max(abs(sum(a.*repmat(temp_phase,1,K))));
            if temp_max<min_value min_value = temp_max;
                Best_n = n;
                end
        end
        aa = sum(a.*repmat(Phase_Set(Choose(Best_n,:)).',1,K)); %
Represent the accumulation process
        Signal_Power = abs(aa.^2);
        Peak_Power = max(Signal_Power,[],2);
        Mean_Power = mean(Signal_Power,2);
        PAPR_PTS(nSymbol) = 10*log10(Peak_Power./Mean_Power);
    end
    close(hwait);
```

```
    [cdf1, PAPR1] = ecdf(PAPR_Orignal);
    [cdf2, PAPR2] = ecdf(PAPR_SLM);
    [cdf3, PAPR3] = ecdf(PAPR_PTS);
    semilogy(PAPR1,1-cdf1,'linewidth',2)
    hold on;
    semilogy(PAPR2,1-cdf2,'linewidth',2)
    hold on;
    semilogy(PAPR3,1-cdf3,'linewidth',2)
    legend(' Orignal',' SLM',' PTS');
    xlabel('PAPR0 [dB]');
    ylabel('CCDF (Pr[PAPR>PAPR0])');
    axis([5 12 10e-4 1])
    grid on
```

code for PAPR simulation and aproxmiation

```
X=0:.5:10;
N=64;
Y=1-(1-exp(-X)).^N;
R=exp(-X)
xlabel('X(dB)')
ylabel('CCDF P(|s(t) |^2/Ps >x)')
title('The complementary Cumulative distribution function (CCDF),N=64')
plot(X,Y,'--rs','LineWidth',2,...
                'MarkerEdgeColor','B',...
                'MarkerFaceColor','g',...
                'MarkerSize',10)
hold on
plot(X,R,'--mo','LineWidth',.3,...
                'MarkerEdgeColor','y',...
                'MarkerFaceColor','b',...
                'MarkerSize',5)
            % Add a legend in the upper right
legend('approxmitted.','simulation.','Location','northeast')
```