Anexo I: Códigos

A continuación se expone el listado de funciones utilizadas para poder implementar en Matlab las simulaciones realizadas en este Proyecto Fin de Carrera. Se expone el código completo para las funciones, aunque hay fragmentos de código opcionales que se pueden calcular o no de forma opcional con solo poner o quitarlo como comentario.

```
function OFDMlineal(N)

% Transmisor

%       Generador de bits y mapeador 4QAM

datos=randsrc(N*1000,1,[1,-1])+j*randsrc(N*1000,1,[1,-1]);
B=1;
plot(real(datos),imag(datos),'.');
axis([-2,2,-2,2])

for EbNo=0:0.5:20
   totErr=0;
   numBits=2*N*1000;
   maxNumErrs=50;
   maxNumBits=N*10000000;
   while((totErr < maxNumErrs) && (numBits < maxNumBits))

%       Modulador OFDM

   for K=1:1000
       datos_transmitidos=datos(((K-1)*N+1):(K*N));
       transf_datos(((K-1)*N+1):(K*N),1)=ifft(datos_transmitidos);
   end


% Canal


   energia_s=0;
   for H=1:N
   energia_s=energia_s+(abs(transf_datos(H,1)))^2;
   end
   potencia_db=10*log10(energia_s/N);
   SNR=0;
   SNR=EbNo+10*log10(2);
   rx=awgn(transf_datos,SNR,potencia_db);


% Receptor


%       Demodulador OFDM

   for G=1:1000
        datos_recibidos=rx(((G-1)*N+1):(G*N));
        datos_final(((G-1)*N+1):(G*N),1)=fft(datos_recibidos);
   end
```

```
   plot(real(datos_final),imag(datos_final),'.');
   axis([-2,2,-2,2])


%       Decisor 4QAM

   receptor=sign(real(datos_final))+j*sign(imag(datos_final));


%       Cálculo de la probabilidad de error


   aux=receptor-datos;
   cont=0;
   for i=1:N*1000
        if aux(i)~=0
            if real(aux(i))~=0
                cont=cont+1;
            end
            if imag(aux(i))~=0
                cont=cont+1;
            end
        end
   end
   totErr=totErr+cont;
   numBits=numBits+2*N*1000;
   end
ber(B)=totErr/(numBits-2*N*1000);
B=B+1;
end
EbNo(1:41)=[0:0.5:20];
semilogy(EbNo,ber)
grid



function OFDMnolinealSSPA(N,Ao)


% Transmisor


%       Generador de bits y mapeador 4QAM

datos=randsrc(N*1000,1,[1,-1])+j*randsrc(N*1000,1,[1,-1]);
B=1;
plot(real(datos),imag(datos),'.');
axis([-2,2,-2,2])

for EbNo=0:0.5:30
   totErr=0;
   numBits=2*N*1000;
   maxNumErrs=50;
   maxNumBits=N*10000000;
   while((totErr < maxNumErrs) && (numBits < maxNumBits))

%       Modulador OFDM

   for K=1:1000
       datos_transmitidos=datos(((K-1)*N+1):(K*N));
       transf_datos(((K-1)*N+1):(K*N),1)=ifft(datos_transmitidos);
```

```
   end

%     Amplificador SSPA

   modulo=abs(transf_datos);
   fase=angle(transf_datos);
   for M=1:N*1000
       modulo_NL(M,1)=modulo(M,1)/(sqrt(1+((modulo(M,1)/Ao)^2)));
   end
   senal_NL=modulo_NL.*exp(j*fase);

%  alfa=sum(((conj(senal_NL)).*transf_datos))/(N*1000*potencia_senal);
%    media_senal_NL=mean(senal_NL);
%    media_senal=mean(alfa.*transf_datos);
%    ruido=senal_NL-((real(alfa)).*transf_datos);
%    media_ruido=mean(ruido);
%    potencia_ruido=sum((abs(ruido)).^2)/(N*1000);
%    potencia_util=((real(alfa)).^2)*2/N;
%    potencia_total=((real(alfa)).^2)*2/N+potencia_ruido;
   OBO=10*log10(Ao^2/(potencia));


% Canal


   energia=sum((abs(senal_NL)).^2);
   potencia=energia/(N*1000);
   potencia_db=10*log10(potencia);
   SNR=EbNo+10*log10(2);
   rx=awgn(transf_datos,SNR,potencia_db);


% Receptor


%       Demodulador OFDM

   for G=1:1000
        datos_recibidos=rx(((G-1)*N+1):(G*N));
        datos_final(((G-1)*N+1):(G*N),1)=fft(datos_recibidos);
   end
   plot(real(datos_final),imag(datos_final),'.');
   axis([-2,2,-2,2])


%       Decisor 4QAM

   receptor=sign(real(datos_final))+j*sign(imag(datos_final));


%       Cálculo de la probabilidad de error


   aux=receptor-datos;
   cont=0;
   for i=1:N*1000
        if aux(i)~=0
            if real(aux(i))~=0
                cont=cont+1;
            end
            if imag(aux(i))~=0
```

```
                cont=cont+1;
            end
        end
   end
   totErr=totErr+cont;
   numBits=numBits+2*N*1000;
   end
ber(B)=totErr/(numBits-2*N*1000);
B=B+1;
end
EbNo(1:61)=[0:0.5:30];
semilogy(EbNo,ber)
grid



function OFDMnolinealTWT(N,Ao)


% Transmisor


%       Generador de bits y mapeador 4QAM

datos=randsrc(N*1000,1,[1,-1])+j*randsrc(N*1000,1,[1,-1]);
B=1;
plot(real(datos),imag(datos),'.');
axis([-2,2,-2,2])

for EbNo=0:0.5:30
   totErr=0;
   numBits=2*N*1000;
   maxNumErrs=50;
   maxNumBits=N*10000000;
   while((totErr < maxNumErrs) && (numBits < maxNumBits))

%       Modulador OFDM

   for K=1:1000
       datos_transmitidos=datos(((K-1)*N+1):(K*N));
       transf_datos(((K-1)*N+1):(K*N),1)=ifft(datos_transmitidos);
   end

%     Amplificador TWT

   Asat=2*Ao;
   modulo=abs(transf_datos);
   fase=angle(transf_datos);

   for M=1:N*1000

   modulo_NL(M,1)=((Asat)^2)*modulo(M,1)/(((modulo(M,1))^2)+(Asat)^2);
   fase_NL(M,1)=(pi/3)*((modulo(M,1))^2)/(((modulo(M,1))^2)+(Asat)^2);

   end

   senal_NL=modulo_NL.*exp(j*fase+fase_NL);

%  alfa=sum(((conj(senal_NL)).*transf_datos))/(N*1000*potencia_senal);
%    media_senal_NL=mean(senal_NL);
%    media_senal=mean(alfa.*transf_datos);
```

```
%    ruido=senal_NL-((real(alfa)).*transf_datos);
%    media_ruido=mean(ruido);
%    potencia_ruido=sum((abs(ruido)).^2)/(N*1000);
%    potencia_util=((real(alfa)).^2)*2/N;
%    potencia_total=((real(alfa)).^2)*2/N+potencia_ruido;
   OBO=10*log10(Ao^2/(potencia));


% Canal


   energia=sum((abs(senal_NL)).^2);
   potencia=energia/(N*1000);
   potencia_db=10*log10(potencia);
   SNR=EbNo+10*log10(2);
   rx=awgn(transf_datos,SNR,potencia_db);


% Receptor


%       Demodulador OFDM

   for G=1:1000
        datos_recibidos=rx(((G-1)*N+1):(G*N));
        datos_final(((G-1)*N+1):(G*N),1)=fft(datos_recibidos);
   end
   plot(real(datos_final),imag(datos_final),'.');
   axis([-2,2,-2,2])


%       Decisor 4QAM

   receptor=sign(real(datos_final))+j*sign(imag(datos_final));


%       Cálculo de la probabilidad de error


   aux=receptor-datos;
   cont=0;
   for i=1:N*1000
        if aux(i)~=0
            if real(aux(i))~=0
                cont=cont+1;
            end
            if imag(aux(i))~=0
                cont=cont+1;
            end
        end
   end
   totErr=totErr+cont;
   numBits=numBits+2*N*1000;
   end
ber(B)=totErr/(numBits-2*N*1000);
B=B+1;
end
EbNo(1:61)=[0:0.5:30];
semilogy(EbNo,ber)
grid
```