

ABODE DANIEL  
Course Work Report  
Comparison of OFDM and GFDM for 5G

## 1. Background

5G is required to meet the low latency, high data rate, connection density and other demands of applications that requires enhanced mobile broadband, ultra-reliable and low latency connection and massive machine type connection [1]. The capacity of the channel required for such communication is limited by noise, attenuation, distortion and intersymbol interference, which can make the transmitted information irrecoverable at the receiver. In order to compensate for channel capacity, multicarrier signaling techniques such as OFDM replaced CDMA in 4G systems [2]. 5G systems have higher expectation in terms of spectral efficiency and data rate, hence new multicarrier signaling techniques such as GFDM are being investigated because of its advantages of lower out of band radiation and better spectral efficiency while retaining the flexibility and simplicity of OFDM [3].

A typical communication system shown in figure 1 consist of the transmitter which contains components that helps in coding, modulating and shaping a signal, a channel through which the signal is transported and a receiver which helps to demodulate, equalize and decode the signal back to the generated data.

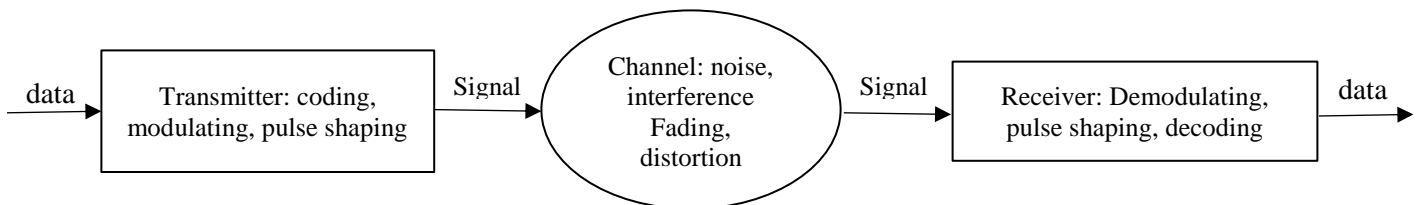


Figure 1: Simplified Physical Layer of Communication System

In this project, the modelling and simulation of OFDM and GFDM system was carried out using MATLAB. The two system were compared by analyzing their signal spectrum, constellation diagram, eye diagram and comparing their bit error rate and symbol error rate at different signal to noise ratio. The result showed that GFDM is better in terms of spectral efficiency, though OFDM performed better in terms of BER, it was observed that a better equalization method would improve GFDM BER. The major disadvantage of GFDM is the increased complexity in design.

## 2. Methodology

### 2.1 GFDM and OFDM Transmitter

The designed transmitter block diagram for the OFDM and GFDM system is shown in figure 2.1a and 2.1b respectively. The function of each block and implementation is given in the table below

Components	Function and Implementation
Binary Source	The binary source generates random bits of 1 and 0s. For the project, we use the randsrc() function of MATLAB the binary sequence of length $L = 2^{16}$ .
Mapper	The mapper converts the binary sequence to symbols, where a symbol is 1 or more binary data. For the project, we use the qammod() function of MATLAB and user can choose to use QPSK (2 bits per symbol) or 16QAM (4 bits per symbol). The length of the symbol sequence is $L/\text{number of bits per symbol}$ . The result is parallel into N number of subcarriers and K number of blocks for OFDM, meanwhile for GFDM, it is converted to N, M number of symbols and K using the reshape() function. For both cases $N = 256$ , with 128 null subcarriers while for OFDM $M = 1$ , $K = \text{number of symbols} / N$ . For GFDM, K and M are variable and can be set by user.
IFFT Block	The IFFT block helps to modulate each subcarriers on a different frequency. This was implemented using the IFFT() function of MATLAB.
Digital Pulse Shaping	The GFDM design includes a Digital Pulse Shaping block. For the project, I use a Raised Root Cosine Filter to reduce the out of band radiation.
Cyclic Prefix Adder	The CP block adds a variable length of CP to the signal, for OFDM it adds this to each blocks of 1 subsymbol, while for GFDM it adds to each block of M subsymbols. The CP acts as guard band to reduce intersymbol interference and to make it easy for the receiver to perform single-tap equalization.

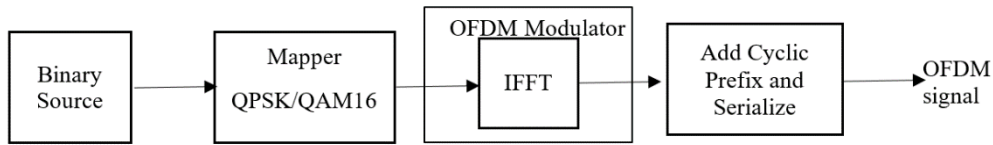


Figure 2.1a OFDM Transmitter

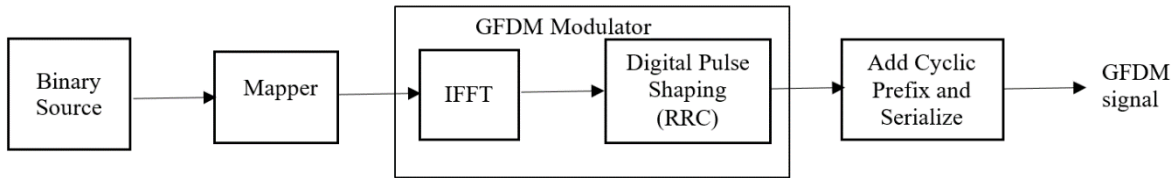


Figure 2.1 b GFDM Transmitter

The signal format for the OFDM and the GFDM block before serializing for transmission is shown in figure 2.1c,d below, N is the number of subcarriers, K is the number of blocks and M is the number of sub symbols, CP. denotes the length of cyclic prefix.



Figure 2.1c GFDM Block

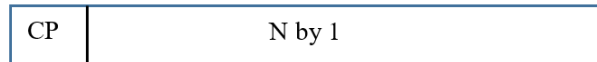
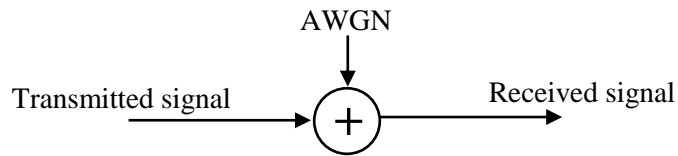


Figure 2.1d OFDM Block

## 2.2 Channel Model

For this project, and Adaptive White Gaussian Noise (AWGN) was used to model the channel. AWGN models the random process of noisy signals that affects desired signal in channel. The noise was generated using matlab randn() function. The effect of this channel is the linear addition of white noise to the signal.

The channel capacity of an AWGN channel is given by  $C = \frac{1}{2} \log(1 + P/N)$  where  $N$  is noise density and  $P$  is signal power.



## 2.3 GFDM and OFDM Receiver

The designed receiver for the OFDM and GFDM signal is shown in figure 2.3a and 2.3b respectively. The function and implementation of each block is discussed in the table below.

Components	Function and Implementation
Remove CP block	The remove CP block remove CP from the input signal.
Matched Filter	This performs equalization on the signal to reverse the effect of the pulse shaping done in the transmitter. The matched filter coefficients is the conjugate of the coefficient of the RRC filter used at the transmitter.
FFT	The FFT block demodulates the signal by reversing the effect of the IFFT block in the transmitter. It was implemented using the FFT() of matlab.
De-mapper	This de-maps the symbol using the qamdemod() function of matlab.
Binary Sink	The binary sequence produce by the De-mapper is stored in the binary sink.

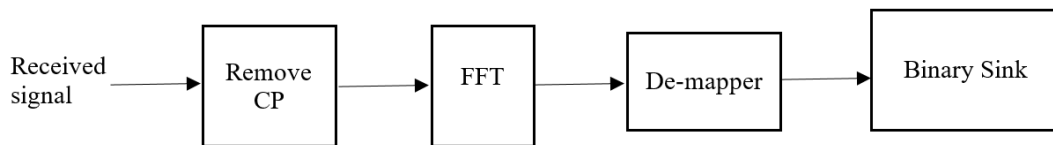


Figure 2.3a OFDM Receiver

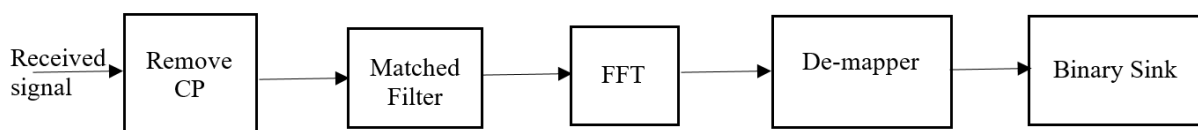


Figure 2.3 b GFDM Receiver

### 3. Result and Analysis

A comprehensive analysis and comparison of the OFDM system and the GFDM system was carried out by several methods as described below.

#### 3.1 Signal Spectrum Comparison of the generated OFDM and GFDM signals

Figure 3.1a shows the signal spectrum of the OFDM and GFDM signal. It can be observed that the side bands energy is greatly reduced in GFDM than in OFDM with about 15dB difference, the effect of this is a reduction in Out of Band (OOB) radiation, and hence multiple carriers can be orthogonally multiplexed with minimum interference between adjacent carriers. Figure 3.1b shows the effect of the number of blocks on the spectrum of the GFDM signal without CP, from the plot we can observe that the OOB radiation is lesser for 2 blocks than for 32 blocks. This is an advantage for GFDM, whose number of blocks can be varied. For OFDM, the number of blocks is maximum.

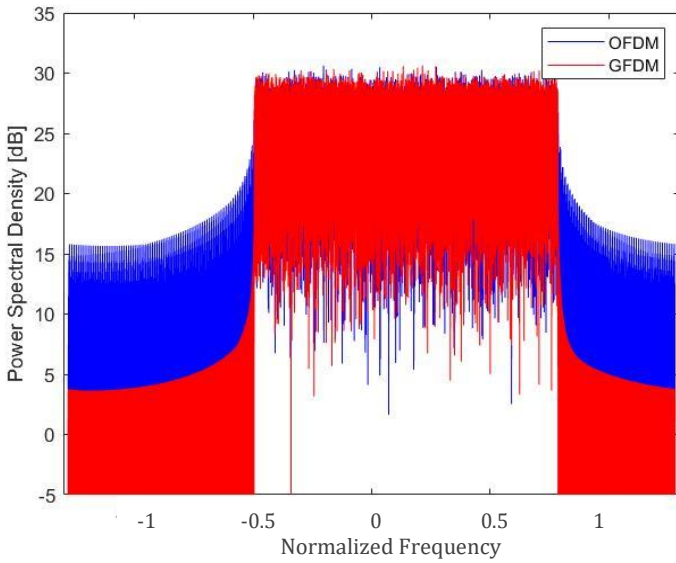


Figure 3.1a GFDM vs OFDM spectrum

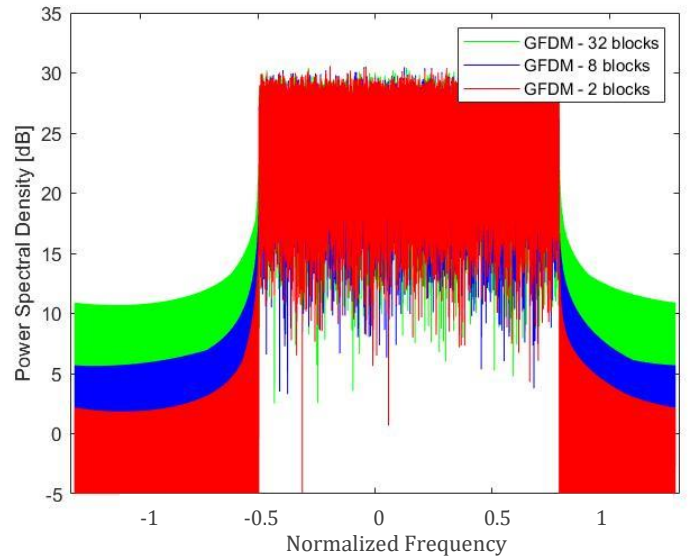


Figure 3.1b Spectrum of GFDM of different blocks

#### 3.2 Effect of Cyclic Prefix on Spectral Efficiency of the OFDM and GFDM signal

The spectral efficiency indicator for OFDM and GFDM is less than 1 considering their requirement for cyclic prefix which introduce additional overhead. Figure 3.2 shows the spectral efficiency indicator for OFDM and GFDM signals. It can be observed that for both SEI reduced as CP length increases, but this variation is higher in OFDM than in GFDM, hence, GFDM has a better spectrum efficiency than OFDM in agreement with 3.1.

### 3.3 Bit Error Rate (BER) comparison of OFDM and GFDM signal

Figure 3.3 shows the BER performance of OFDM and GFDM signal for QAM4 modulation format. It can be observed that OFDM perform better in this case as its BER plot is close that the ideal case of the theoretical BER, while GFDM performed worst as it shows a large deviation from the ideal case. This is due to the complex process in generating the signal, such that the recovery of the information through equalization becomes more difficult.

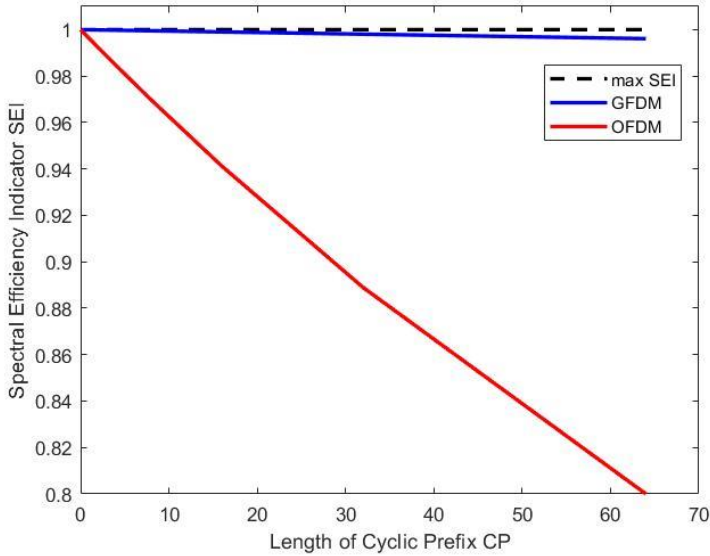


Figure 3.2 SEI Comparison vs length of CP

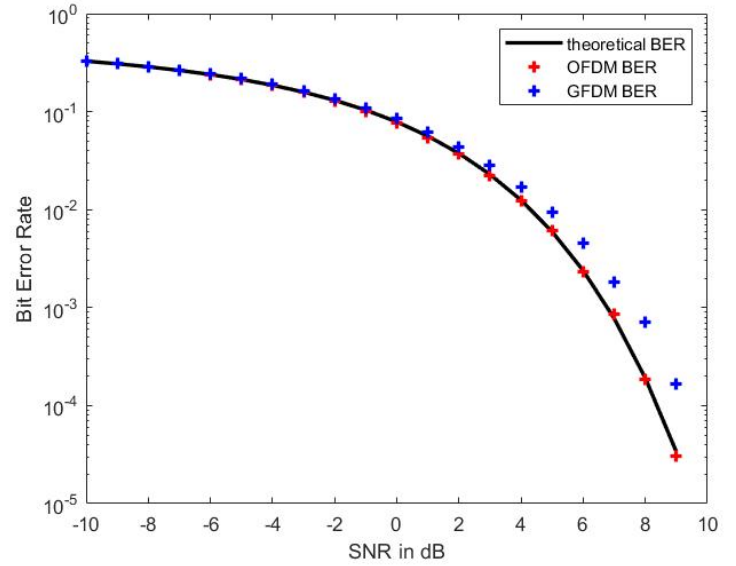


Figure 3.3 QPSK BER comparison for OFDM and GFDM

### 3.4 Constellation Comparison of the received OFDM and GFDM signal at SNR of 10dB

The constellation diagram in figure 3.4 confirms the difficulty posed to the demodulator in making the hard decision of the GFDM symbols compare to OFDM symbols. It can be observed that more GFDM symbols appear closer to the decision boundaries and could have crossed the decision boundary compared to OFDM.

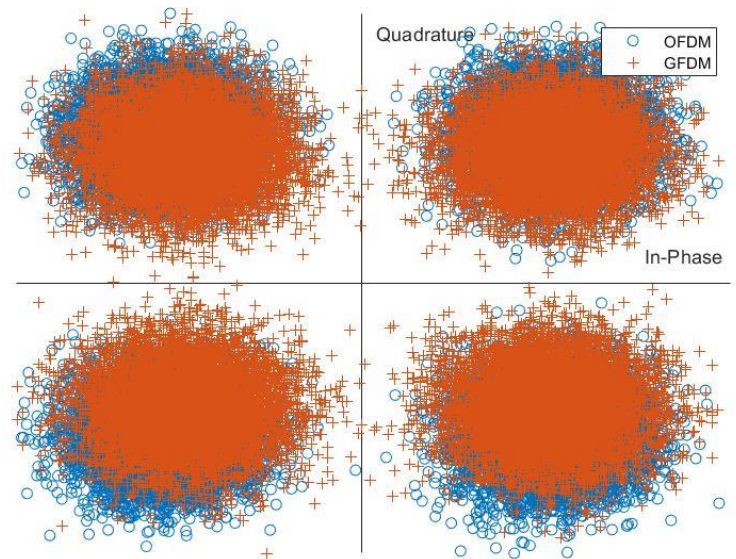


Figure 3.4 Constellation diagram comparison

### 3.6 Complexity Comparison of OFDM and GFDM generation and recovery

From figure 2.1a,b and 2.3a,b of the designed systems, it can be seen that GFDM transmitter and receiver has an additional block of pulse shaping filter and matched filter respectively compared to OFDM transmitter and receiver. Hence, GFDM system is more complex than OFDM. The main part of the OFDM modulation is the IFFT block which has a complexity of  $O(N\log N)$ . The GFDM demodulation involved circular shift of complexity  $O(N)$ , IFFT of complexity  $O(N\log N)$  and convolution with the filter with complexity  $O(N^2)$ .

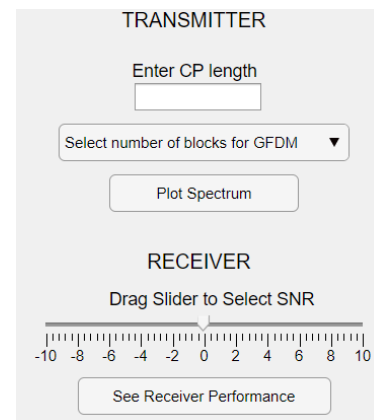
### 3.7 User Interface Result

A GUI shown below was built which allows user to enter some parameters and simulate the described results.

## 4. Conclusion and Recommendation

In this project, the design and simulation of generation and recovery of OFDM and GFDM signal was carried out and they were compared base on signal spectrum, spectral efficiency, bit error rate and constellation. The comparison result is given in the table below; the best for each comparison is ticked.

Performance Metric	OFDM	GFDM
Out of Band Radiation		✓
Spectral Efficiency Indicator		✓
Bit Error Rate Performance	✓	
System Complexity	✓	



The BER of GFDM can be improved by improving the equalization method. Otherwise, the trade off is majorly between system complexity and performance. GFDM would be highly useful in systems where we require a higher spectral efficiency.

### References

- 1) H.S. et al "Machine-type Communication," IEEE Communication Magazine, pp.10-17,2015
- 2) M.S. et.al, "Comparative analysis of OFDM and GFDM," IJACTA, vol 4, no. 2, 2016.
- 3) G.F, M.K and S.B, "GFDM – Generalized Frequency Division Multiplexing," IEEE, 2009.

# USER MANUAL

The code is divided into 4 m files, transmitter.m, OFDMreceiver.m, GFDMreceiver.m, and user\_interface.m. All four files must be inserted into 1 folder.

Transmitter.m : This is an object oriented file, the transmitter was defined as a class, with properties and methods such as the constructor, the mapper, the RRCPulseShape, the modulate. The code can be seen below;

```
classdef transmitter < handle
% This class implements a transmitter
properties
    no_of_subcarriers; no_of_subsymbols; no_of_blocks; no_of_total_subcarriers;
    no_of_bin_data; len_of_mapped_signal; len_of_CP; binary_source;
    mapped_signal; no_of_bits_per_symbol; modulation_type;
    Pulse_Shaping_Filter_Type; filter_pulse; signal_type; transmitted_signal;
end
methods
function obj = transmitter()
%This is the class constructor
    obj.no_of_bin_data = 2^16;
    obj.binary_source = randsrc(1,obj.no_of_bin_data,[0 1]);
    obj.no_of_subcarriers = 128;
    obj.no_of_total_subcarriers = 256;
end
function mapper(obj)
% function generates QPSK symbols
    obj.no_of_bits_per_symbol = 2;
    mapped_signal_ = qammod(obj.binary_source', 2^obj.no_of_bits_per_symbol, 'InputType','bit','UnitAveragePower',true);
    obj.mapped_signal = mapped_signal_;
    obj.modulation_type = 'QPSK';
    obj.len_of_mapped_signal = length(mapped_signal_);
end
function res = RRCPulseShape(obj)
% This generates the RRC filter response
    M = obj.no_of_subsymbols; K = obj.no_of_total_subcarriers; a = 0.1;
    t = linspace(-M/2, M/2, M*K+1); t = t(1:end-1); t = t';
    g = (sin(pi*t*(1-a))+4*a.*t.*cos(pi*t*(1+a)))/(pi.*t.*(1-(4*a*t).^2));
    g(find(t==0)) = 1-a+4*a/pi;
    g(find(abs(t) == 1/(4*a))) = a/sqrt(2)*((1+2/pi)*sin(pi/(4*a)))+(1-2/pi)*cos(pi/(4*a));
    g = fftshift(g); res = g / sqrt(sum(g.*g)); obj.filter_pulse = res;
end

function gen_signal = modulate(obj, signal_type_, cp_len, num_of_subsymbols)
    obj.len_of_CP = cp_len;
    mapped_sig = obj.mapped_signal;
    obj.signal_type = signal_type_;
    if (signal_type_ == 'OFDM')
        obj.no_of_subsymbols = 1;
        obj.no_of_blocks = obj.len_of_mapped_signal/obj.no_of_subcarriers;
        obj.Pulse_Shaping_Filter_Type = '';
        reshaped_signal = reshape(mapped_sig, [],obj.no_of_blocks);
        ofdm_signal = [zeros(50,obj.no_of_blocks);reshaped_signal;zeros(78,obj.no_of_blocks)];
        ofdm_signal = ifft(ofdm_signal);
        ofdm_signal_cp = [ofdm_signal(:,end - obj.len_of_CP+1 : end), ofdm_signal]; %add cp
        obj.transmitted_signal = reshape(ofdm_signal_cp,1,[]);
        obj.transmitted_signal = obj.transmitted_signal/std(obj.transmitted_signal);
        gen_signal = obj.transmitted_signal;
    end
    if (signal_type_ == 'GFDM')
        obj.Pulse_Shaping_Filter_Type = 'Raised Root Cosine';
        obj.no_of_subsymbols = num_of_subsymbols;
        obj.no_of_blocks = obj.len_of_mapped_signal/(obj.no_of_subcarriers*obj.no_of_subsymbols);
        block_len = obj.no_of_subsymbols * obj.no_of_subcarriers;
        total_block_len = obj.no_of_total_subcarriers * obj.no_of_subsymbols;
        GFDM_signal = zeros((obj.no_of_blocks*total_block_len)+cp_len,1); %add cp
        for i = 0:obj.no_of_blocks-1
            sig = mapped_sig(block_len*i+1:end-(block_len*(obj.no_of_blocks - 1 - i)));
            sig_resaped = reshape(sig, obj.no_of_subcarriers, obj.no_of_subsymbols);
            sig_total = zeros(obj.no_of_total_subcarriers,obj.no_of_subsymbols);
```



```

sig_total(51:obj.no_of_subcarriers+50,:)= sig_resaped;
filter_value = RRC_PulseShape(obj);
sig_rep = repmat(obj.no_of_total_subcarriers*ifft(sig_total),obj.no_of_subsymbols,1);
sig_sum = zeros(total_block_len, 1);
for j = 1:obj.no_of_subsymbols
    sym = sig_rep(:,j).*filter_value;
    sym = circshift(sym, obj.no_of_total_subcarriers*(j-1));
    sig_sum = sig_sum + sym;
end
sig_cp = [sig_sum(end-cp_len+1:end); sig_sum];
GFDM_signal(i*(total_block_len+cp_len) + (1:total_block_len + cp_len)) = sig_cp;
end
obj.transmitted_signal = GFDM_signal';
obj.transmitted_signal = obj.transmitted_signal/std(obj.transmitted_signal);
gen_signal = obj.transmitted_signal;
end
end
end
end

```

To use the class, you can create an object from it by;

```

>> t = transmitter(); %creates and object of the transmitter
>> t.mapper(); %maps with QPSK
>> ofdm_signal = t.modulate('OFDM',cp_len,number_of_subsymbols) %generate OFDM signal
>> gfdm_signal = t.modulate('GFDM',cp_len,number_of_subsymbols) %generate GFDM signal

```

OFDMreceiver.m: This is a function implements the OFDM receiver and generates the performance plot.

```

function [SER, BER] = OFDMreceiver(SNR,cp)
trans = transmitter(); trans.mapper(); trans.modulate('OFDM',cp,1);
OFDM_signal = trans.transmitted_signal;
snr_dB = [-10:9]; errors = zeros(1,length(snr_dB)); berD = zeros(1,length(snr_dB));
n = 1/sqrt(2)*(randn(1,length(OFDM_signal))+ 1i*randn(1,length(OFDM_signal))); %Gaussian noise

for i = 1:length(snr_dB)
    r_OFDM_signal = OFDM_signal + 10^(-snr_dB(i)/20)*n;
    r_OFDM_signal_matrix = reshape(r_OFDM_signal,trans.no_of_total_subcarriers,[]);
    r_OFDM_signal_ = r_OFDM_signal_matrix(:,trans.len_of_CP+1:end); %remove CP
    fft_OFDM_signal = fft(r_OFDM_signal_);
    fft_OFDM_signal_ = fft_OFDM_signal(51:trans.no_of_subcarriers+50,:); % remove null subcarriers
    rcv_mapped_signal = reshape(fft_OFDM_signal_,1,[]);
    if snr_dB(i) == SNR, s_snr = rcv_mapped_signal; end
    rcv_bin_data_ = qamdemod(rcv_mapped_signal,2^trans.no_of_bits_per_symbol,'OutputType','bit','UnitAveragePower',true);
    rcv_bin_data = reshape(rcv_bin_data_,1,[]);
    errors(i) = length(find(trans.binary_source-rcv_bin_data));
    berD(i) = berawgn(snr_dB(i),'qam',4);
end

ber = errors/trans.no_of_bin_data;
ser = ber / trans.no_of_bits_per_symbol;
SER = ser(snr_dB == SNR); BER = ber(snr_dB == SNR);
snrlin=10.^(snr_dB./10);
formula_ber = 0.5 * erfc(sqrt(snrlin));
r_rcv_ofdm = normalize(real(s_snr),'range',[-1 1]);
i_rcv_ofdm = normalize(imag(s_snr),'range',[-1 1]);
a = gca;
scatter(r_rcv_ofdm,i_rcv_ofdm) %plot constellation
a.YAxisLocation = 'origin'; a.XAxisLocation = 'origin'; a.XTick = []; a.YTick = [];
xlabel('In-Phase'), ylabel('Quadrature'), title(strcat('Constellation of OFDMreceiver at SNR=',num2str(SNR),'dB'))
figure
eyediagram(r_rcv_ofdm + i_rcv_ofdm*1j,2); %plot eye diagram
figure
plot(snr_dB, log10(ber), '-x');
hold on
plot(snr_dB, log10(formula_ber),'-o');
ylabel('Bit Error Rate');xlabel('SNR in dB');

```



end

To use this function, you don't need to run the transmitter class separately but rather call the function by giving a variable for the SNR between (-10 9) and a cyclic prefix length typically less than 128.

[SER, BER] = OFDMreceiver(SNR,cp)

GFDMreceiver() function implements the GFDM receiver and the code is shown below

```
function [SER, BER] = GFDMreceiver(SNR,cp,n_subsym)
trans = transmitter(); trans.mapper(); trans.module('GFDM',cp,n_subsym);
GFDM_signal = trans.transmitted_signal; sen = trans.binary_source;g = trans.filter_pulse;
snr_dB = -10:10;
errors = zeros(length(snr_dB), 1);
s_ = zeros(trans.no_of_total_subcarriers,trans.no_of_subsymbols,trans.no_of_blocks);
sf = zeros(trans.no_of_subcarriers,trans.no_of_subsymbols,trans.no_of_blocks);
s2 = zeros(trans.no_of_subcarriers*trans.no_of_subsymbols,trans.no_of_blocks);
s3 = zeros(1,trans.no_of_subcarriers*trans.no_of_subsymbols*trans.no_of_blocks);
n = 1/sqrt(2)*(randn(1,length(GFDM_signal))+ 1i*randn(1,length(GFDM_signal)));
for si = 1:length(snr_dB)
    rcv_signal = GFDM_signal + 10^((-snr_dB(si)/20)*n); %pass through channel
    x = circshift(rcv_signal.*exp(1i*2*pi*0*(1:length(rcv_signal))), 0);
    y = reshape(x,[],trans.no_of_blocks);
    y = y(trans.len_of_CP+1:end,:); % remove CP
    IQ = zeros(trans.no_of_total_subcarriers,trans.no_of_subsymbols,trans.no_of_blocks);
    D = zeros(trans.no_of_total_subcarriers,trans.no_of_subsymbols,trans.no_of_blocks);
    s = zeros(trans.no_of_total_subcarriers*trans.no_of_subsymbols,trans.no_of_blocks);
    for b = 1:trans.no_of_blocks
        yhat = y(:,b);
        G = conj(fft(g));
        L = length(G)/trans.no_of_subsymbols;
        xhat = fft(yhat);
        Dhat = zeros(trans.no_of_total_subcarriers,trans.no_of_subsymbols);
        for k = 1:trans.no_of_total_subcarriers
            carrier = circshift(xhat,ceil(L*trans.no_of_subsymbols/2) - trans.no_of_subsymbols*(k-1));
            carrier = fftshift(carrier(1:L*trans.no_of_subsymbols));
            carrierMatched = carrier .* G;
            dhat = ifft(sum(carrierMatched,trans.no_of_subsymbols,L),2)/L;
            D(k,:,b) = dhat;
        end
        s(:,b) = reshape(D(:,b), numel(D(:,b)),1);
        s_(:,b) = reshape(s(:,b),trans.no_of_total_subcarriers,trans.no_of_subsymbols);
        sf(:,b) = s_(51:trans.no_of_subcarriers+50,:); %remove null carriers
        s2(:,b) = reshape(sf(:,b),[],1);
    end
    s3 = reshape(s2,1,[]);
    if snr_dB(si) == SNR, s_snr = s3; end
    sk = qamdemod(s3, 2^2,'gray','OutputType','bit','UnitAveragePower',true);
    sx = reshape(sk,1,[]);
    errors(si) = length(find(sx' - sen));
end
ber2 = errors/length(sx);
ser = ber2 / trans.no_of_bits_per_symbol;
SER = ser(snr_dB == SNR); BER = ber2(snr_dB == SNR);
plot(snr_dB, log10(ber2), '-x');
legend({'OFDM BER','theoretical BER','GFDM BER'});
hold off
figure
a = gca;
r_rcv_gfdm = normalize(real(s_snr),'range',[-1 1]);
i_rcv_gfdm = normalize(imag(s_snr),'range',[-1 1]);
scatter(r_rcv_gfdm,i_rcv_gfdm,'+')
a.YAxisLocation = 'origin'; a.XAxisLocation = 'origin'; a.XTick = []; a.YTick = [];
xlabel('In-Phase'), ylabel('Quadrature'), title(strcat('Constellation of GFDMreceiver at SNR=',num2str(SNR),'dB'))
figure
eyediagram(r_rcv_gfdm + i_rcv_gfdm*1j,2)
end
```

To use the function, you just need to call it as shown below, input the value of the SNR between (-10 to 9), give the CP length of less than 128, and the number of subsymbols which must be less than 256 and can be a base of two, typical values for subsymbols (256,128,64,32,16,8,4), this will give you number of blocks (1,2,4,,8,16,32,64).

[SER, BER] = GFDMreceiver(SNR,cp,n\_subsym)

Userinterface.m > This code makes it easy for the user to input the variables and view the plots and performance all with a Graphical User Interface. Running the code will generate a GUI, that allows user to enter length of CP, select Block length, plot OFDM and GFDM spectrum, select receiver SNR and view receiver performance with eye diagram, constellation diagram, BER plot and the result for the BER and SER at the SNR selected will be printed on the command window for the user.

```
close all
fig = uifigure;
trans_lbl = uilabel(fig,'Position',[210 380 120 22],'Text','TRANSMITTER', 'FontSize',16);
cp_lbl = uilabel(fig,'Position',[218 340 120 22],'Text','Enter CP length', 'FontSize',14);
cp_txt = uitextarea(fig,'Position',[220 320 100 22],'Value',' ');
blk_dd = uidropdown(fig,'Position',[160 280 230 30],'Items',{'Select number of blocks for GFDM','2','4','8','16','32'},...
    'Value','Select number of blocks for GFDM');
ts_btn = uibutton(fig,'Position',[200 240 150 30],'Text','Plot Spectrum','ButtonPushedFcn', @(ts_btn,event)
    plotSpectrum(ts_btn,str2num(cell2mat(cp_txt.Value)),256/(str2double(get(blk_dd, 'Value'))));
rcv_lbl = uilabel(fig,'Position',[230 190 120 22],'Text','RECEIVER', 'FontSize',16);
blk_lbl = uilabel(fig,'Position',[200 160 250 22],'Text','Drag Slider to Select SNR', 'FontSize',14);
snr_slider = uislider(fig,'Position',[150 150 250 30],'Limits',[-10 10]);
rcv_btn = uibutton(fig,'Position',[175 80 200 30],'Text','See Receiver Performance','ButtonPushedFcn',@(rcv_btn,event)
    Rec_performance(rcv_btn, str2num(cell2mat(cp_txt.Value)),256/(str2double(get(blk_dd, 'Value'))),ceil(snr_slider.Value)));
function plotSpectrum(ts_btn, cp_, sym_num)
cp = cp_; no_of_sym = sym_num;
trans = transmitter();
trans.mapper();
ofdm_sig = trans.modulate('OFDM',cp,1);
gfdm_sig = trans.modulate('GFDM',cp,no_of_sym);
close all
figure
f1 = linspace(-trans.no_of_total_subcarriers/2, trans.no_of_total_subcarriers/2, 2*length(ofdm_sig)+1);
f1 = f1(1:end-1); f1 = normalize(f1,'range', [-1 1]);
f2 = linspace(-trans.no_of_total_subcarriers/2, trans.no_of_total_subcarriers/2, 2*length(gfdm_sig)+1);
f2 = f2(1:end-1); f2 = normalize(f2, 'range', [-1 1]);
plot(f1, mag2db(flip(abs(fft(ofdm_sig, 2*length(ofdm_sig)))/2, 'b');
title('OFDM Spectrum'),
xlabel('normalize frequency'), ylabel('Power Spectral Density [dB]');ylim([-10 30])
figure
plot(f2, mag2db(flip(abs(fft(gfdm_sig, 2*length(gfdm_sig)))/2, 'b');
title('GFDM Spectrum'),
xlabel('normalize frequency'), ylabel('Power Spectral Density [dB]'); ylim([-10 30])
end
function Rec_performance(rcv_btn, cp_, sym_num, snr)
cp = cp_; no_of_sym = sym_num; user_snr = snr;
close all;
[oser, ober] = OFDMreceiver(user_snr,cp);
[gser, gber] = GFDMreceiver(user_snr,cp,no_of_sym);
fprintf('OFDM BER = %f at SNR = %d \n',ober,snr); fprintf('OFDM SER = %f at SNR = %d\n',oser,snr);
fprintf('GFDM BER = %f at SNR = %d \n',gber,snr); fprintf('GFDM SER = %f at SNR = %d\n',gser, snr);
end
```