Table of Contents

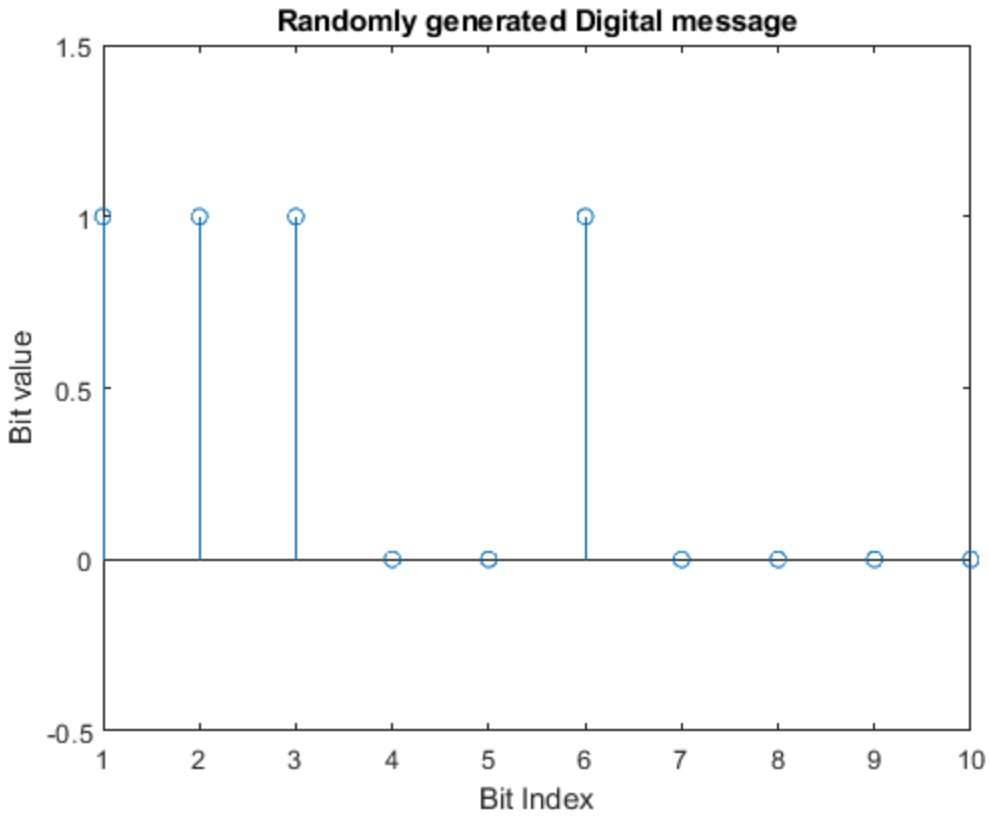
setting parameters 1
Message source ------> generates a rondom stream of 0s and 1s 1
BFSK modulator ---> send a different carrier depending on the bit sent 2
Constellation of transsmitted BFSK ---> two basis functions 3
Adding whie additive Gussian noise from a normal distribution 4
Constellation of recieved BFSK 5
Signal transimission decoder -----> using ML rule 6
PSD of the transimmited baseband BFSK 7

setting parameters

```
clear
clc

Tb = 5; %% bit duration in secs
Eb = 5; %% Energy per bit
N_bit = 500; %%number of samples per bit
t_bit = linspace(0,Tb,N_bit); %% time base for each bit
msg_l = 10 ; %% number of bits sent
t_signal = linspace(0,msg_l*Tb,msg_l*N_bit); %% total duration of the
 messag
N0 = 5;

% carrier frequencies
f1 =(2+1)/Tb;
f2 =(2+2)/Tb;
fc =2/Tb;
```

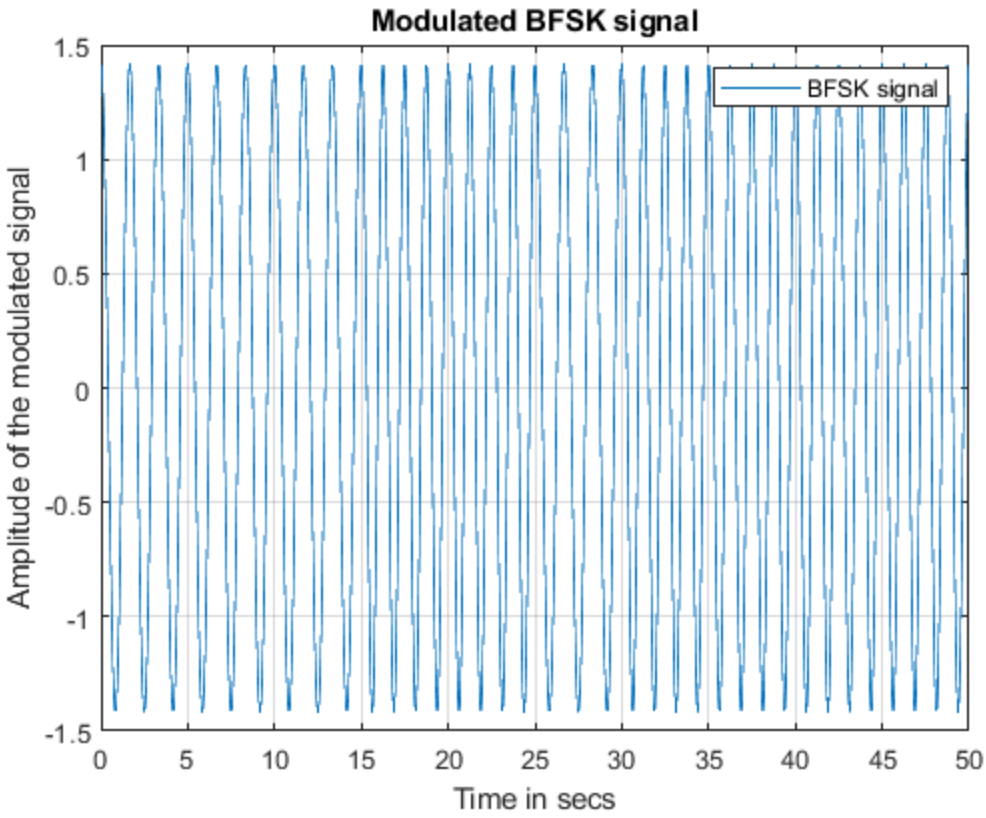
Message source ------> generates a rondom stream of 0s and 1s

```
message = randi([0 1],1,msg_l);
figure
stem(message)
title('Randomly generated Digital message')
ylabel('Bit value')
xlabel('Bit Index')
ylim([-0.5 1.5])
```



BFSK modulator ---> send a different carrier depending on the bit sent

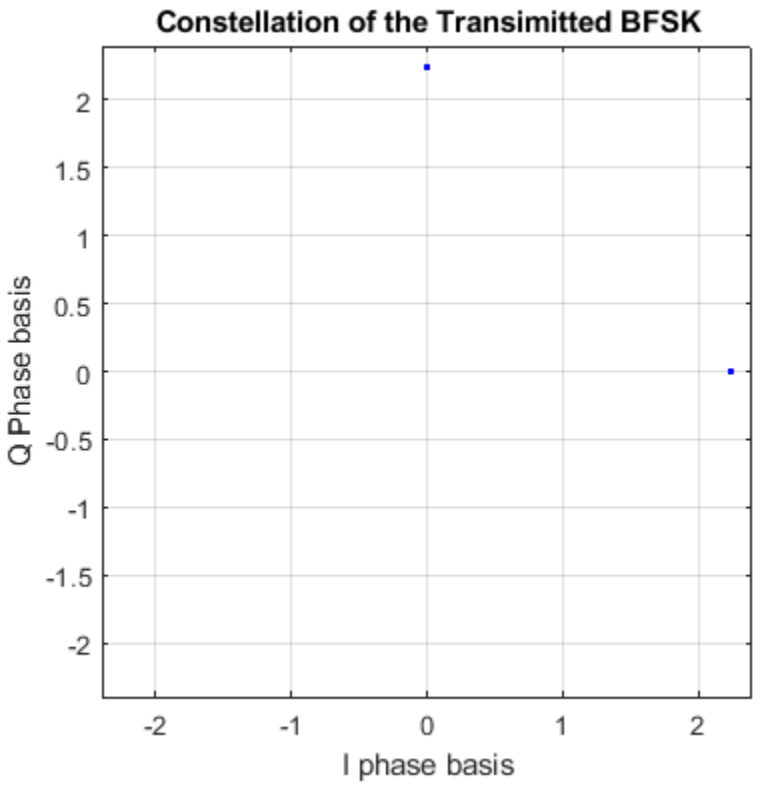
```
carrier_I = sqrt(2*Eb/Tb) * cos(2*pi*f1*t_bit);
carrier_Q = sqrt(2*Eb/Tb) * cos(2*pi*f2*t_bit);

modulatedSignal = [];

for i = 1:1:length(message)
    if message(i) == 1
       modulatedSignal= [modulatedSignal carrier_I];
    elseif message(i) == 0
        modulatedSignal= [modulatedSignal carrier_Q];
    end
end

figure
plot(t_signal,modulatedSignal)
title('Modulated BFSK signal')
ylabel('Amplitude of the modulated signal')
xlabel('Time in secs')
grid on
legend('BFSK signal')
```



Constellation of transsmitted BFSK ---> two basis functions

```
basis_func_i = sqrt(2/Tb)*cos(2*pi*f1*t_bit);
basis_func_Q = sqrt(2/Tb)*cos(2*pi*f2*t_bit);

% Since it is a two dimentional signal space, each message point is
% represented by a pair of numbers
si1_vector=[];
si2_vector=[];

% projection on Inphase
for i = 1:N_bit:length(modulatedSignal)
    vec = modulatedSignal(i:i+N_bit-1);
    vec = vec.*basis_func_i;
    intg = trapz(t_bit,vec); %% seperation is tb
    si1_vector=[si1_vector intg];
end

% projection on quadratue

for i = 1:N_bit:length(modulatedSignal)
    vec = modulatedSignal(i:i+N_bit-1);
    vec = vec.*basis_func_Q;
```

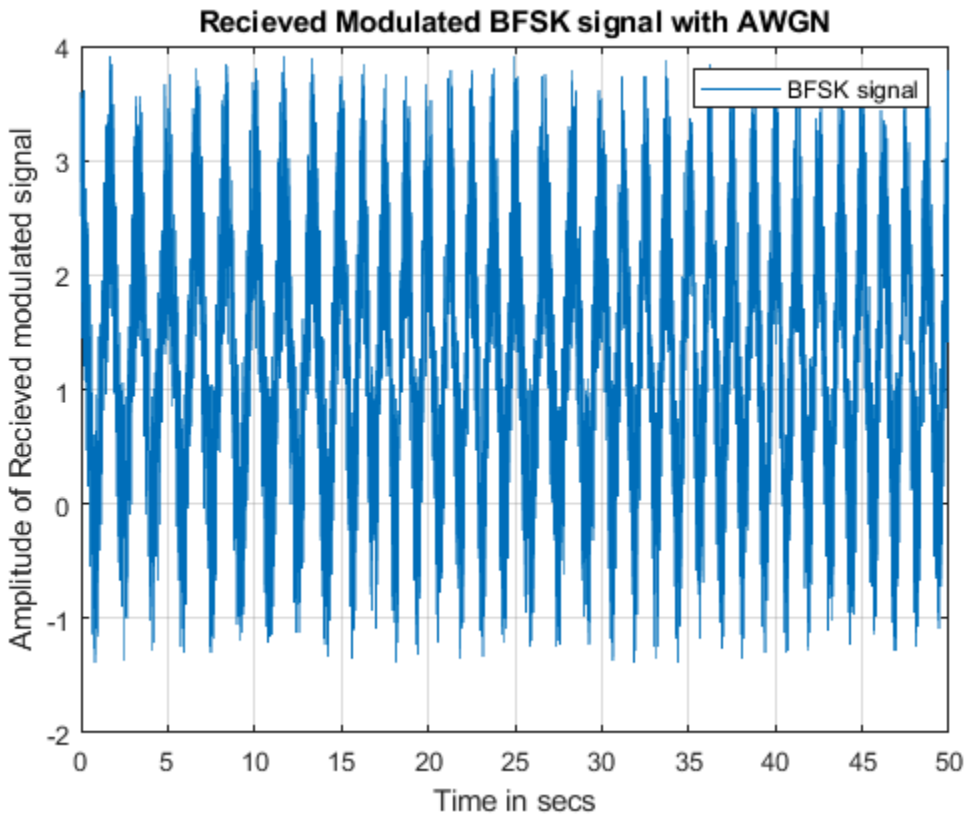
```
    intg = trapz(t_bit,vec); %% seperation is tb
    si2_vector=[si2_vector intg];
end

si_vector = [si1_vector ; si2_vector];

scatterplot(transpose(si_vector));

title('Constellation of the Transimitted BFSK')
ylabel('Q Phase basis')
xlabel('I phase basis')
grid on
```



Adding whie additive Gussian noise from a normal distribution

```
recievedSignal = modulatedSignal +
 unifrnd(0,N0/2,1,length(modulatedSignal));

figure()
plot(t_signal,recievedSignal)

title(' Recieved Modulated BFSK signal with AWGN ')
ylabel(' Amplitude of Recieved modulated signal ')
xlabel(' Time in secs ')
```

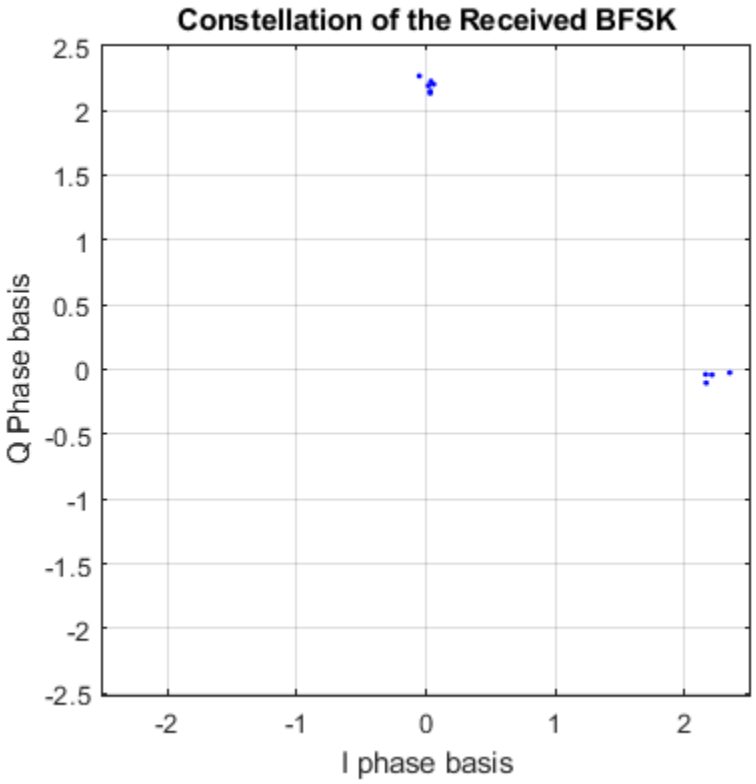
```
grid on
legend('BFSK signal')
```



Constellation of recieved BFSK

```
basis_func_i = sqrt(2/Tb)*cos(2*pi*f1*t_bit);
basis_func_Q = sqrt(2/Tb)*cos(2*pi*f2*t_bit);

xi1_vector=[];
xi2_vector=[];

for i = 1:N_bit:length(recievedSignal)
    vec = recievedSignal(i:i+N_bit-1);
    vec = vec.*basis_func_i;
    intg = trapz(t_bit,vec); %% seperation is tb
    xi1_vector = [xi1_vector intg];
end

for i = 1:N_bit:length(recievedSignal)
    vec = recievedSignal(i:i+N_bit-1);
    vec = vec.*basis_func_Q;
    intg = trapz(t_bit,vec); %% seperation is tb
    xi2_vector = [xi2_vector intg];
end

xi_vector=[xi1_vector ; xi2_vector];
```

```
scatterplot(transpose(xi_vector))
title('Constellation of the Received BFSK')
ylabel('Q Phase basis')
xlabel('I phase basis')
grid on
```



Signal transimission decoder -----> using ML rule

```
rec_signal_Decoded = [];

d1=[]; % distance from 1
d2=[]; % distance from 0

for i = 1:1:length(xi_vector)

x1 = sqrt((xi_vector(1,i) - sqrt(Eb))^2 + (xi_vector(2,i) - 0)^2);
x2 = sqrt((xi_vector(1,i) - 0)^2 + (xi_vector(2,i) - sqrt(Eb))^2);

d1=[d1 x1];
d2=[d2 x2];
end
d1=abs(d1);
d2=abs(d2);
```

```
for i=1:1:length(xi2_vector)
    if d1(i)<d2(i)
        rec_signal_Decoded=[rec_signal_Decoded 1];
    else
        rec_signal_Decoded=[rec_signal_Decoded 0];
    end

end
```

PSD of the transimmited baseband BFSK

```
base_I = sqrt(2*Eb/Tb)* cos(pi*t_bit/Tb);
base_Q = sqrt(2*Eb/Tb)* sin(pi*t_bit/Tb);
base_band = [];

for i = 1:1:length(message)
    seg = base_I + (-1)^message(i)*base_Q;
   base_band = [base_band seg];
end

ts = Tb/N_bit;
[psd,f]= periodogram(base_band,[],[],1/ts*30);

f = f/Tb;
psd = 10*log10(psd/(2*Eb));

% figure
% plot(f,psd)
% xlim([0 5])
% title('PSD of the Transimitted BFSK')
% ylabel('Spectral power Density in dB ')
% xlabel('Normalized Frequency')
% grid on
% legend('BFSK PSD')
```

Published with MATLAB® R2019b