1 Adaptive Modulation

1.(a)

The first step is to generate simulated signals under each modulation. As the range of simulated Signal-to-Noise Ratio (SNR) is set between 0 and 40, the number of bits is set as 2^20, whose power is equal to the average value of SNR. The rest initial config is placed in Fig. 1, in which the target bit error ratio (BER) is required as 3×10^{-3}, and Rayleigh random number is generated for simulation of channel.

```
%% initialize Data
N=2^20;                    %Number of bits to simulate
target_BER = 3e-3;         %Define the thresholds
Mo1=2;                         %BPSK
Mo2=4;                         %QPSK
Mo3=16;                        %16QAM
x1=randi([0,Mo1-1],1,N);       %Produce the random signal
x2=randi([0,Mo2-1],1,N);
x3=randi([0,Mo3-1],1,N);
R=raylrnd(0.5,1,N);          %Produce the Rayleigh signal
Adaptive_Modulation_THROUGHPUT=[];
```

Fig. 1. Initial configuration.

The following step is to integrate Rayleigh random number into random signals of each modulation, which presents the situation of Rayleigh channels. Then, by loop of each SNR, after demodulation steps to simulate receiving signal, the bit error is extracted for further calculations. The extracted ratio is formed into corresponding modulation type. The codes for generating fading channel and corresponding demodulation are illustrated in Fig. 2.

```
%% Modulation and transmit over a Rayleigh-distributed narrowband fading channel.
h1=pskmod(x1,Mo1);           %BPSK Modulation
h2=pskmod(x2,Mo2);           %QPSK Modulation
h3=qammod(x3,Mo3);           %16QAM Modulation
H1=h1.*R;                  %BPSK with Rayleigh Channel
H2=h2.*R;                  %QPSK with Rayleigh Channel
H3=h3.*R;                  %16QAM with Rayleigh Channel

%% AWGN and Demodulation for each modulation
for SNR=1:1:40

    y_RE_n1=R.\awgn(H1,SNR,'measured');
    y_RE_1=pskdemod(y_RE_n1,Mo1);
    [bit_RE1,ratio1]=biterr(x1,y_RE_1);
    BPSK_Ray(SNR)=ratio1;

    y_RE_n2=R.\awgn(H2,SNR,'measured');
    y_RE_2=pskdemod(y_RE_n2,Mo2);
    [bit_RE2,ratio2]=biterr(x2,y_RE_2);
    QPSK_Ray(SNR)=ratio2;

    y_RE_n3=R.\awgn(H3,SNR,'measured');
    y_RE_3=qamdemod(y_RE_n3,Mo3);
    [bit_RE3,ratio3]=biterr(x3,y_RE_3);
    QAM_Ray(SNR)=ratio3;

end
```

Fig. 2. Generation of signals in each modulation.

Based on generated data, the codes to present BER vs SNR curves are placed in Fig. 3, in which the target BER is configured for estimation of thresholds. The produced figure is placed in Fig. 4, and the points of thresholds are marked with black circle. According to that, the thresholds should be 20, 23, and 29.

```
%% Plot figure
figure(1)
SNR=1:1:40;                 %Range of SNR in dB
axis([1 40 10^-5 1]);
semilogy(SNR,BPSK_Ray,':rx');
hold on;
semilogy(SNR,QPSK_Ray,':gx');
semilogy(SNR,QAM_Ray,':bx');
grid on;
line([0 40],[target_BER target_BER],'Color','red','LineStyle','--')
semilogy(thresholds1,BPSK_Ray(thresholds1),'ko');
semilogy(thresholds2,QPSK_Ray(thresholds2),'ko');
semilogy(thresholds3,QAM_Ray(thresholds3),'ko');
legend('BPSK Ray Equalize','QPSK Ray Equalize','16QAM Ray Equalize');
title('the BER curves for each modulation technique');
xlabel('SNR (dB) ');ylabel('BER');
hold off;
```

Fig. 3. Codes for plotting figures.

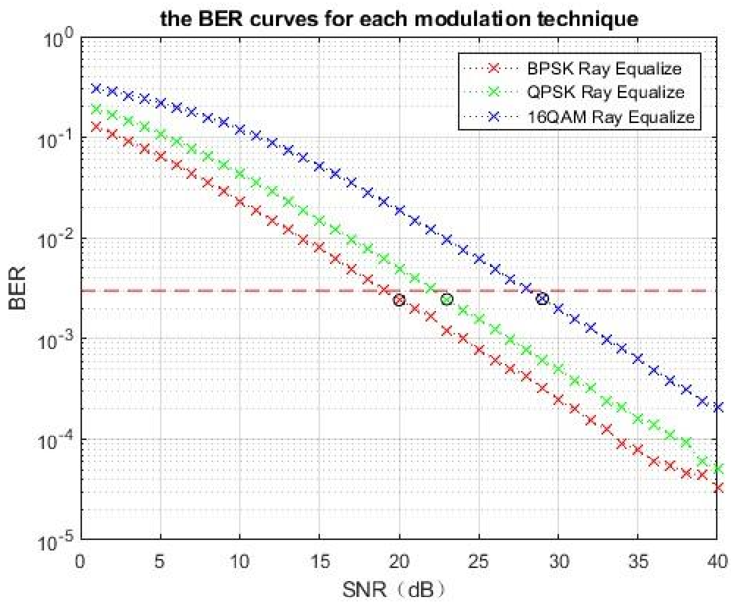


Fig. 4. BER curves for each modulation.

1.(b)

As the thresholds have been decided, the adaptive modulation system could be produced based on existing data. In this case, the transmitted data is consumed as a frame for each SNR with 2^20 bits. It makes the frame-SNR similar to average SNR. However, if the transmitted data is divided into frames and each have its own SNR, to attain corresponding number of error bits, each frame will have demodulation with its SNR. Then, to calculate BER, the number of error bits will be divided by frame length, and it will take average value for frames use equal frame-SNR. It makes the result similar to BER calculated by average SNR, but may cause less accuracy, because of fewer samples. Therefore, it is sensible to assume that length of all frames in each frame-SNR is equal to 2^20, and it transmit same message with different channel situation.

The simulation of adaptive system could be implemented by demodulation, because the transmitter could be assumed has knowledge of the frame-SNR. The codes of that are placed in

Fig. 5, In the range of SNR divided by thresholds, the transmitted signals will be demodulated by corresponding modulation type or no operation that means no transmission.

```
%% Adaptive Demodulation
for SNR=1:1:40
    if SNR <thresholds1                         %Choose Modulation method
       continue;                          %Don't transmit data
    end
    if SNR>=thresholds1 && SNR<thresholds2
        y_RE_n1=R.\awgn(H1,SNR,'measured');
        y_RE_1=pskdemod(y_RE_n1,Mo1);
        [bit_RE1,ratio]=biterr(x1,y_RE_1);
        Adaptive_Modulation(SNR)=ratio;
    elseif SNR>=thresholds2 && SNR<thresholds3
        y_RE_n2=R.\awgn(H2,SNR,'measured');
        y_RE_2=pskdemod(y_RE_n2,Mo2);
        [bit_RE2,ratio]=biterr(x2,y_RE_2);
        Adaptive_Modulation(SNR)=ratio;
    elseif SNR>=thresholds3
        y_RE_n3=R.\awgn(H3,SNR,'measured');
        y_RE_3=qamdemod(y_RE_n3,Mo3);
        [bit_RE3,ratio]=biterr(x3,y_RE_3);
        Adaptive_Modulation(SNR)=ratio;
    end
end
```

Fig. 5. Codes for adaptive demodulation.

The codes to generate figure is placed in Fig. 6 that refers to the comparison between three simple modulation methods and adaptive modulation. The corresponding produced figure is illustrated by Fig. 7.

```
figure(2)
SNR=1:1:40;                  %Range of SNR in dB
axis([1 40 10^-5 1]);
semilogy(SNR,BPSK_Ray,':rx');
hold on;
semilogy(SNR,QPSK_Ray,':gx');
semilogy(SNR,QAM_Ray,':bx');
grid on;
line([0 40],[target_BER target_BER],'Color','red','LineStyle','--')
semilogy(SNR,Adaptive_Modulation,':ko');
legend('BPSK Ray','QPSK Ray','16QAM Ray','Adaptive Modulation');
title('adaptive modulation system-BER vs SNR');
xlabel('SNR (dB) ');ylabel('BER');
hold off;
```

Fig. 6. Codes for plotting figures.

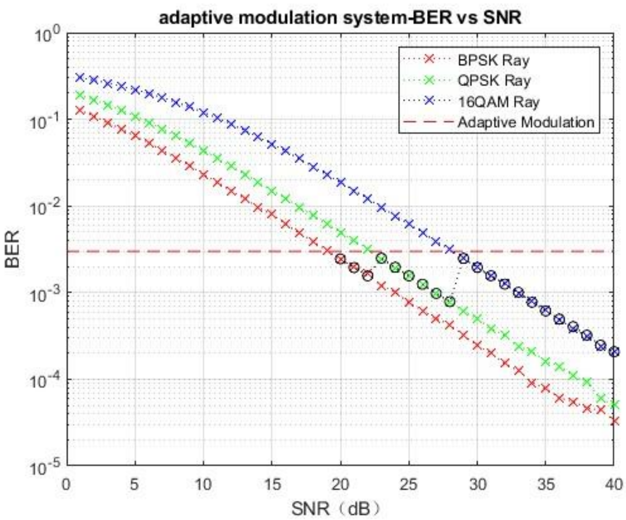


Fig. 7. Comparison between three simple modulation methods and adaptive modulation.

The codes to generate throughput data is illustrated by Fig. 8, which contains throughput of fixed one modulation method and that of corresponding range in adaptive system. The Fig. 9 present codes for figure plot. The comparison of throughput is presented in Fig. 10 with both fixed modulation method and adaptive modulation method.

```
%% Throughput
NO_trans = 0*ones(thresholds1-1,1);
BPSK_trans = 2*ones(thresholds2-thresholds1,1);
QPSK_trans = 4*ones(thresholds3-thresholds2,1);
QAM_trans = 16*ones(40-thresholds3+1,1);
Adaptive_Modulation_THROUGHPUT = [NO_trans;BPSK_trans;QPSK_trans;QAM_trans];

Fixed_BPSK_trans = 2*ones(40-thresholds1+1,1);
Fixed_QPSK_trans = 4*ones(40-thresholds2+1,1);
Fixed_QPSK_NO_trans = 0*ones(thresholds2-1,1);
Fixed_QAM_trans = 16*ones(40-thresholds3+1,1);
Fixed_QAM_NO_trans = 0*ones(thresholds3-1,1);

Fixed_BPSK_THROUGHPUT = [NO_trans;Fixed_BPSK_trans];
Fixed_QPSK_THROUGHPUT = [Fixed_QPSK_NO_trans;Fixed_QPSK_trans];
Fixed_QAM_THROUGHPUT = [Fixed_QAM_NO_trans;Fixed_QAM_trans];
```

Fig. 8. Codes for generation of throughput.

```
%% Plot
figure(3)
axis([0 40 0 20]);
plot(SNR,Adaptive_Modulation_THROUGHPUT,'Linewidth',2);
hold on;
plot(SNR,Fixed_BPSK_THROUGHPUT,'--',SNR,Fixed_QPSK_THROUGHPUT,'--',SNR,Fixed_QAM_THROUGHPUT,'--');
legend({'Adaptive Modulation','Fixed BPSK','Fixed QPSK','Fixed 16QAM'},'Location','northwest');
title('Throughput vs SNR');
xlabel('SNR (dB) ');ylabel('Throughput');
hold off;
```

Fig. 9. Codes for plotting figures.

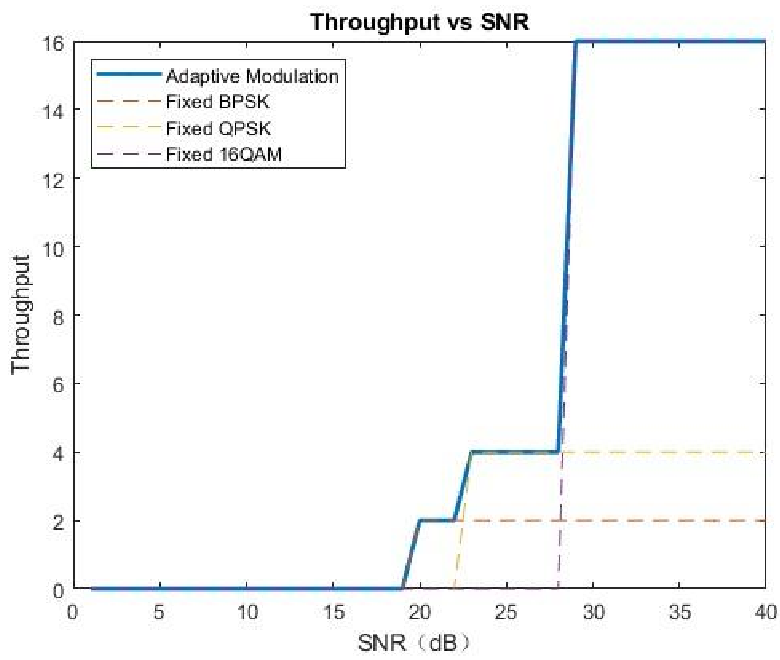


Fig. 10. Comparison of throughput vs SNR in each modulation system.

2 K-nearest neighbors (KNN) learning

2.(a)

The generation of training data with BER and SNR is similar to previous method, which will not be repeated here. In this case, considering the performance of computer, the data is reduced to 2^14 by testing of both performance and accuracy.

To deploy KNN algorithm, the essential step is to generate training data. In this case, to make better output, the dataset is produced by two sets of Rayleigh random number to simulate different channel state. The details of codes are illustrated in Fig. 11, and the distribution of training data is placed in Fig. 12.

```
%% Generate Train Data
for iter=1:1:max_iter

    if iter<=7
        R=raylrnd(0.5,1,N);          %Produce the Rayleigh signal
    else
        R=raylrnd(0.8,1,N);          %Produce another Rayleigh signal
    end
```

Fig. 11. Codes for Rayleigh random signals in different channels.

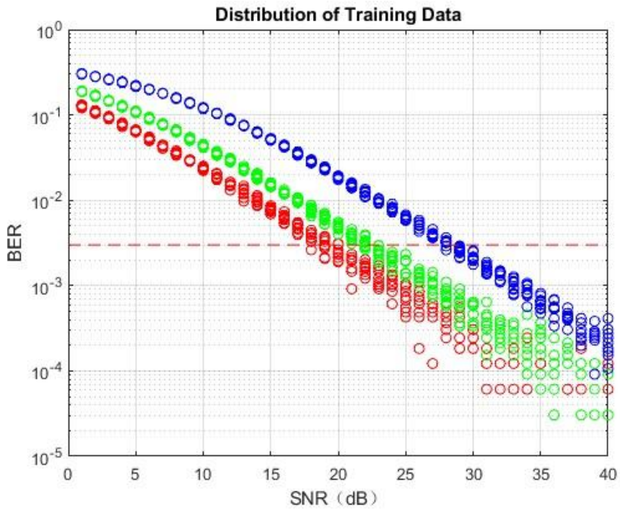


Fig. 12. Distribution of training data.

The next stage is to reform training data with class label. The codes of that are placed in Fig. 13, which reform data into column of BER and SNR with corresponding label. An index is declared here to figure out expected label based on given BER and SNR.

```
%% Reform and process data

for i=1:40
    Output_Data_BPSK_BER=[Output_Data_BPSK_BER;Data_BPSK(:,i)];
    Output_Data_BPSK_SNR=[Output_Data_BPSK_SNR;i.*ones(10,1)];
    Output_Data_QPSK_BER=[Output_Data_QPSK_BER;Data_QPSK(:,i)];
    Output_Data_QPSK_SNR=[Output_Data_QPSK_SNR;i.*ones(10,1)];
    Output_Data_16QAM_BER=[Output_Data_16QAM_BER;Data_16QAM(:,i)];
    Output_Data_16QAM_SNR=[Output_Data_16QAM_SNR;i.*ones(10,1)];
end
%Output it in BER,SNR ~ CLASS style
Output_Data_BPSK=[Output_Data_BPSK_BER,Output_Data_BPSK_SNR];
Output_Data_QPSK=[Output_Data_QPSK_BER,Output_Data_QPSK_SNR];
Output_Data_16QAM=[Output_Data_16QAM_BER,Output_Data_16QAM_SNR];
Output_Data_BER_SNR=[Output_Data_BPSK;Output_Data_QPSK;Output_Data_16QAM];
Output_Data_Class=[repmat({'BPSK'},400,1);repmat({'QPSK'},400,1);repmat({'16QAM'},400,1)];

index=find(Output_Data_BER_SNR(:,1)<=target_BER);
Require_Train_Data=Output_Data_BER_SNR(index,:);
Require_Train_Class=Output_Data_Class(index);
```

Fig. 13. Codes for reforming dataset with label.

2.(b)

Because there will be no transmission if BER is larger than target BER, the index declared in codes presented in Fig. 13 will filter points that means no-transmission in that condition. The KNN algorithm is implemented directly by function “fitcknn” provided by MATLAB. The config of K-value, number of neighbors, is set to 20, which is calculated by commonly 10 points for a particular SNR adding two times of half points number, 2*5=10, which belongs to two near SNRs. It is placed in Fig. 14 that codes to accomplish it and corresponding process to attain prediction.

```
%% Train Data and make predictions on different SNR
KNNC = fitcknn(Require_Train_Data,Require_Train_Class,'NumNeighbors',20,'Standardize',1)
New_Sample=[]

for Pred_SNR=1:1:40
    %Make predictions raw data set

    New_Sample = [New_Sample;target_BER Pred_SNR];
end
%Get predictions
[label,score,cost] = predict(KNNC,New_Sample);

```

Fig. 14. Codes for KNN and corresponding process.

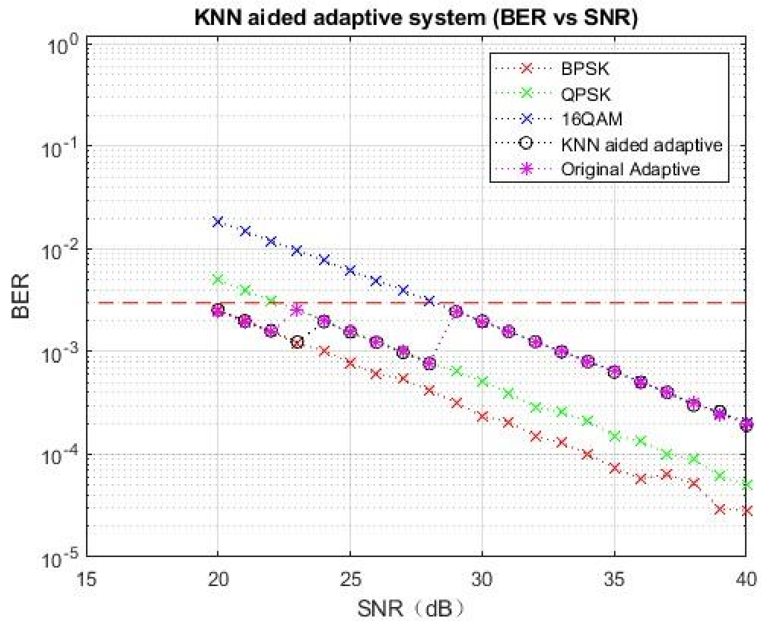
The process to attain signal is similar to previous method that will not be repeated here. There is a little difference could be presented that how KNN prediction work in demodulation. Partial code to accomplish it is placed in Fig. 15. It presents a higher throughput method would be prior choice, but it will choose a lower method if prior choice contains BER higher than target BER.

```
%% KNN-AWGN and Demodulation
for Practical_SNR=thresholds1:1:40
    mod=label{Practical_SNR}
    if strcmp(mod,'BPSK')
        y_RE_n1=R.\awgn(H1,Practical_SNR,'measured');
        y_RE_1=pskdemod(y_RE_n1,Mo1);
        [bit_RE1,ratio]=biterr(x1,y_RE_1);
        KNN_BER(Practical_SNR-(thresholds1-1))=ratio;
        KNN_BPSK_counter=[KNN_BPSK_counter;1];
        continue
    elseif strcmp(mod,'QPSK')
        y_RE_n2=R.\awgn(H2,Practical_SNR,'measured');
        y_RE_2=pskdemod(y_RE_n2,Mo2);
        [bit_RE2,ratio]=biterr(x2,y_RE_2);
        if ratio > target_BER
            y_RE_n1=R.\awgn(H1,Practical_SNR,'measured');
            y_RE_1=pskdemod(y_RE_n1,Mo1);
            [bit_RE1,ratio]=biterr(x1,y_RE_1);
            KNN_BER(Practical_SNR-(thresholds1-1))=ratio;
            KNN_BPSK_counter=[KNN_BPSK_counter;1];
            continue
        end
        KNN_BER(Practical_SNR-(thresholds1-1))=ratio;
        KNN_QPSK_counter=[KNN_QPSK_counter;1];
        continue
    elseif strcmp(mod,'16QAM')
        y_RE_n3=R.\awgn(H3,Practical_SNR,'measured');
        y_RE_3=qamdemod(y_RE_n3,Mo3);
```

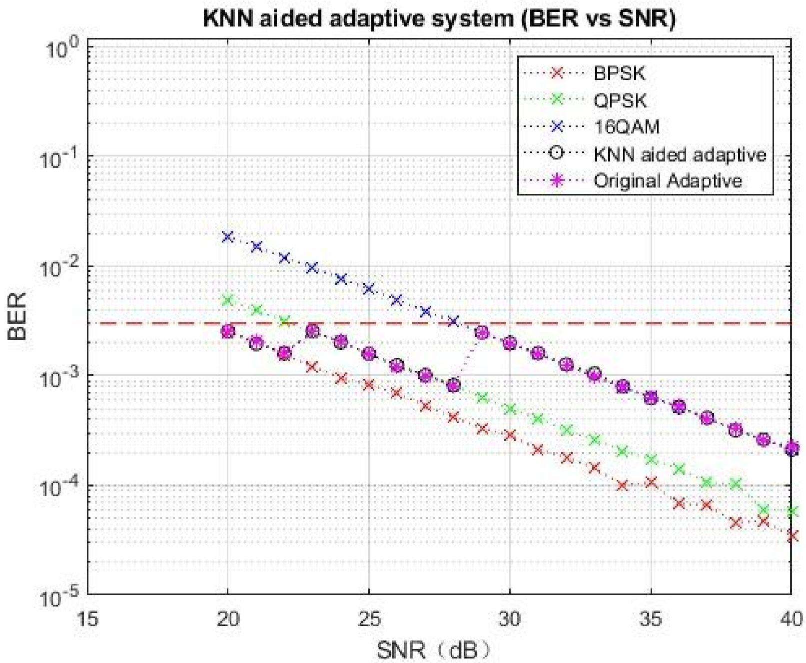
Fig. 15. Codes for KNN prediction work in demodulation.

Another problem needs to be discussed that, sometimes, there will be some controversial points near thresholds. As shown in Fig. 16. (a) and 16. (b), the controversial point occurred near the point where is thresholds between BPSK and QPSK. The problem is caused by a relatively but insufficient higher BER for QPSK at this point. It leads to a possibility that transmission based on

QPSK at such SNR will attain BER larger than target BER, and this possibility will influence training dataset and result. The corresponding throughput is placed in Fig. 17. (a) and (b).



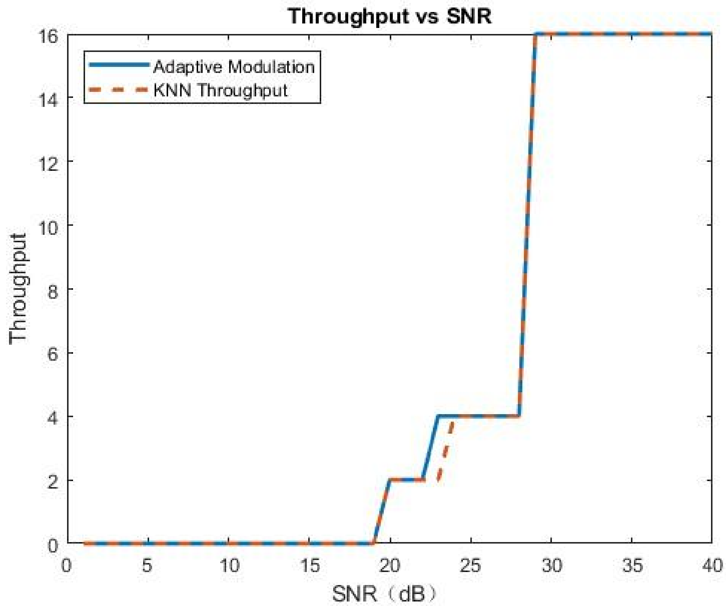
16. (a) Solution with BPSK.



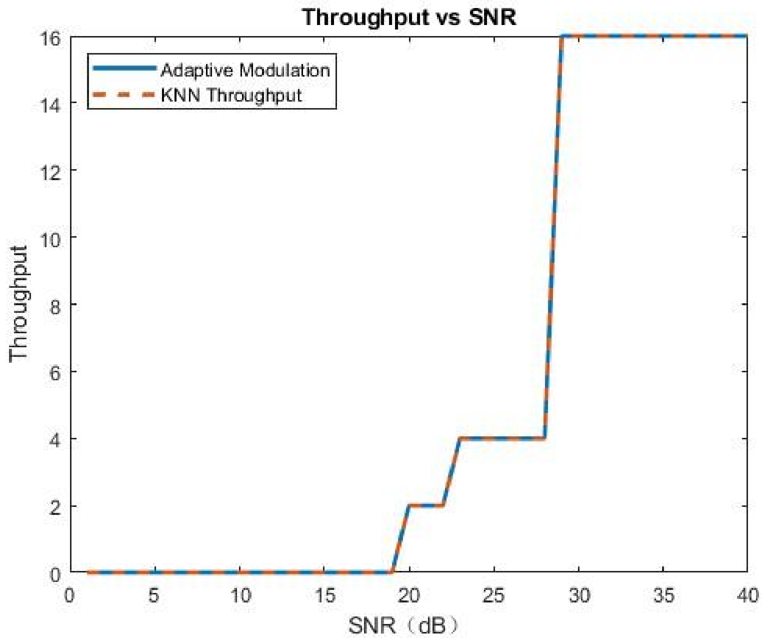
16. (b) Solution with QPSK.

Fig. 16. BER vs SNR of KNN aided adaptive system.

There are several sensible solutions for this problem. From view of training dataset, more samples could provide more precise prediction model, but also require more computational resource and higher expense. Optimization of algorithm design will work but make more requirement for designer. Exploring another algorithm have some potential, as there are various algorithms, it will not be discussed here.



17. (a) Solution with BPSK.



17. (b) Solution with QPSK.

Fig. 17. Throughput of KNN aided adaptive system.