Lab 4

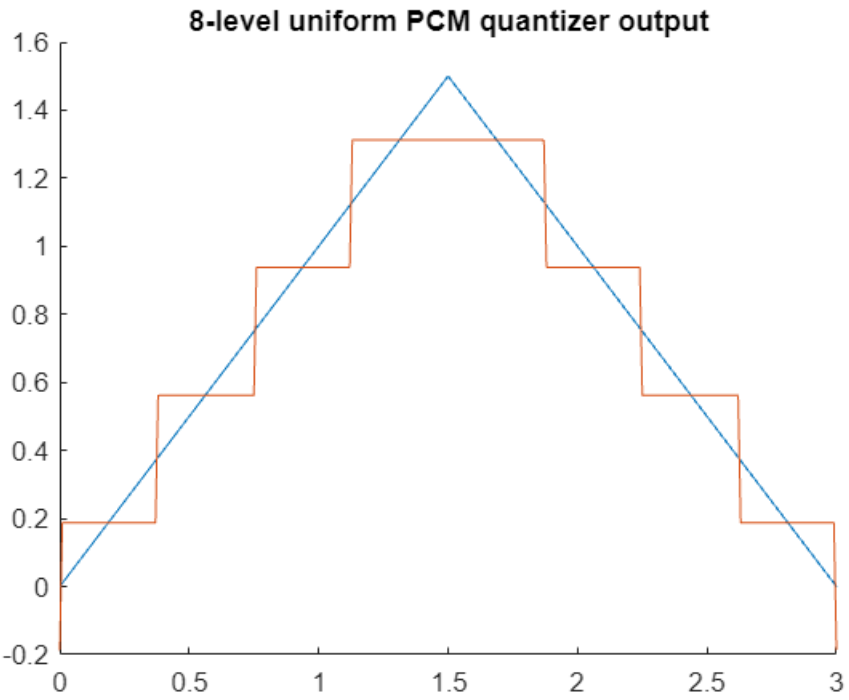
Manthan Parmar 202201416

Rakshit Pandhi 202201426

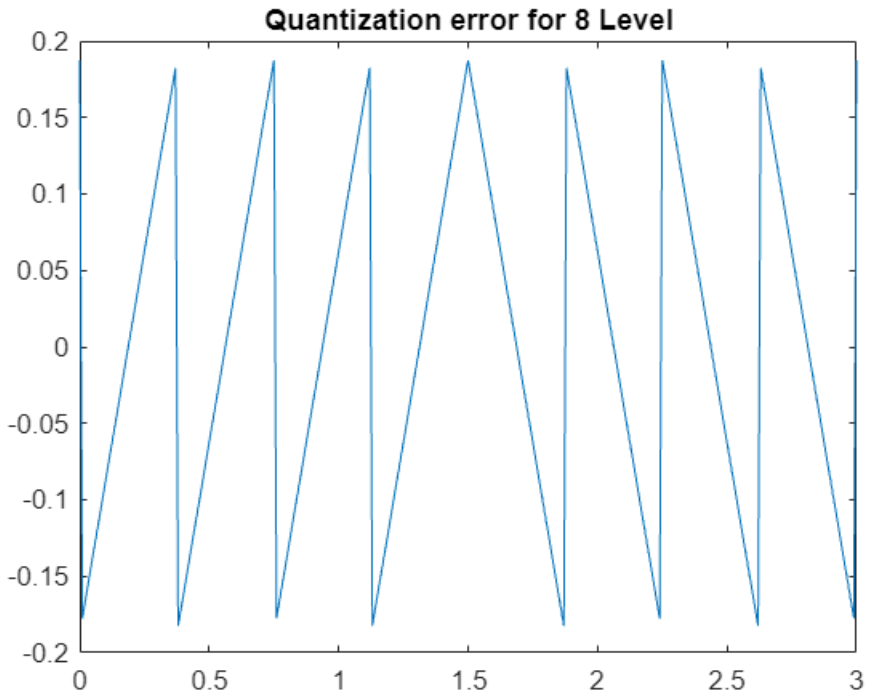
Question 1

(a) 4.17

```
t = 0:0.01:3;
a = t.*(t<1.5)+(-t+3).*(t>=1.5);
mu = 255;
[sqnr8,a_quan8,code8] = u_pcm(a,8);
q_error8 = a - a_quan8;
figure;
hold on;
plot(t,a)
plot(t,a_quan8)
title('8-level uniform PCM quantizer output')
```



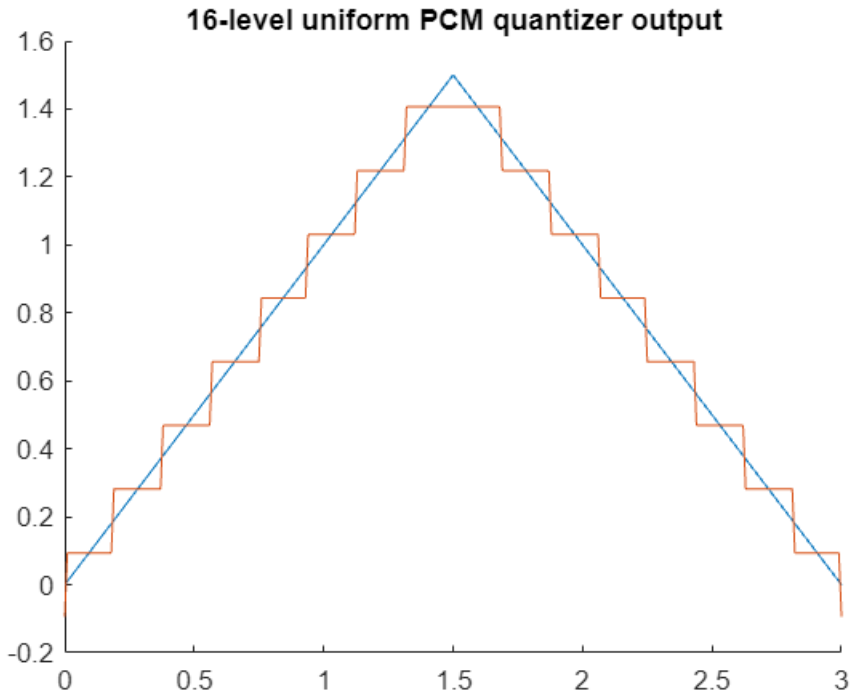

```
figure;
plot(t,q_error8)
title('Quantization error for 8 Level')
```



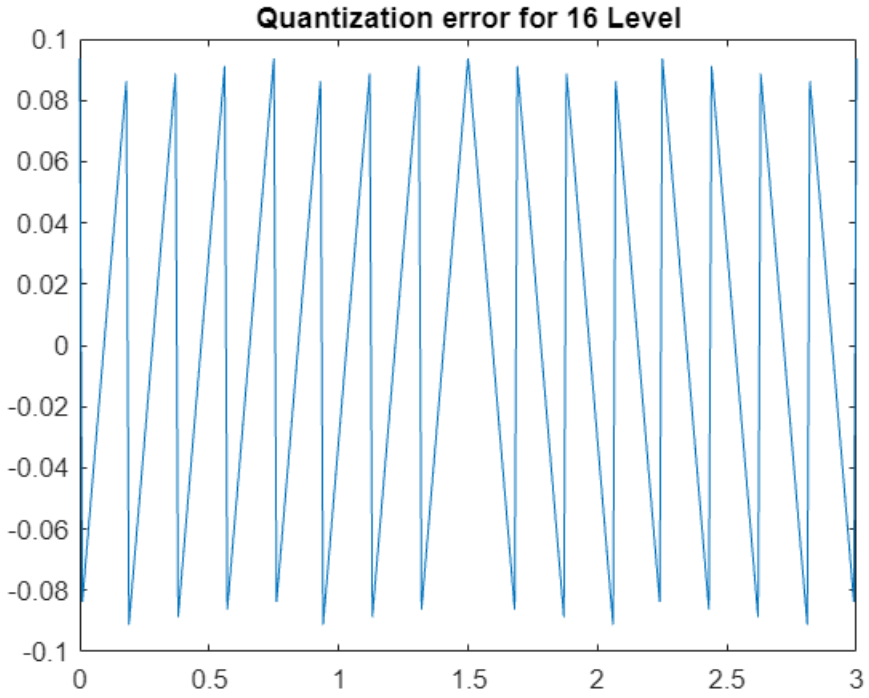

```
fprintf('SQNR for 8 Level Encoder is: %f dB\n', sqnr8);
```

```
SQNR for 8 Level Encoder is: 18.017154 dB
```

```
[sqnr16,a_quan16,code16] = u_pcm(a,16);
q_error16 = a - a_quan16;
figure;
hold on;
plot(t,a)
plot(t,a_quan16)
title('16-level uniform PCM quantizer output')
```




```
figure;
plot(t,q_error16)
title('Quantization error for 16 Level')
```

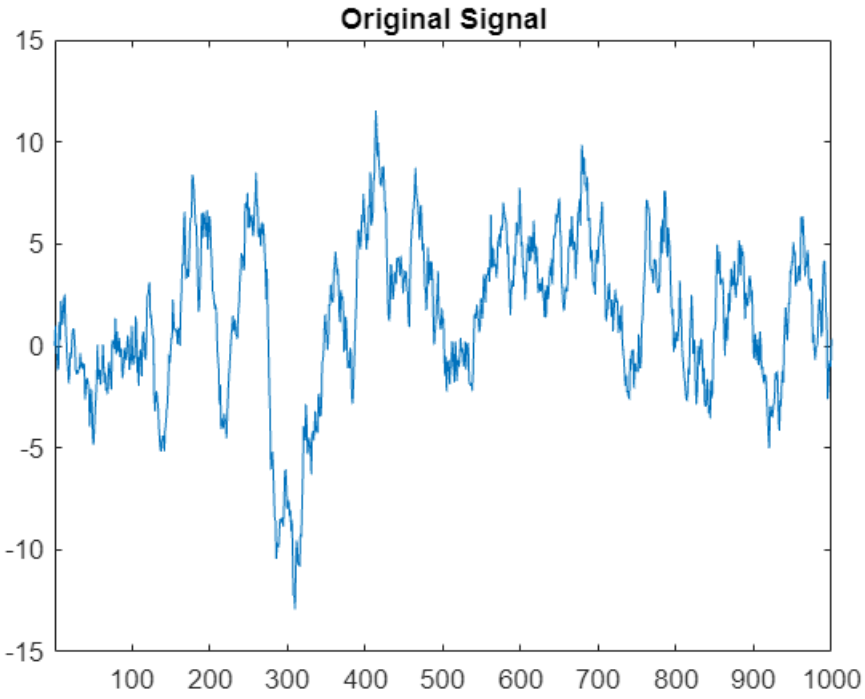



```
fprintf('SQNR for 16 Level Encoder is: %f dB\n', sqnr16);
```

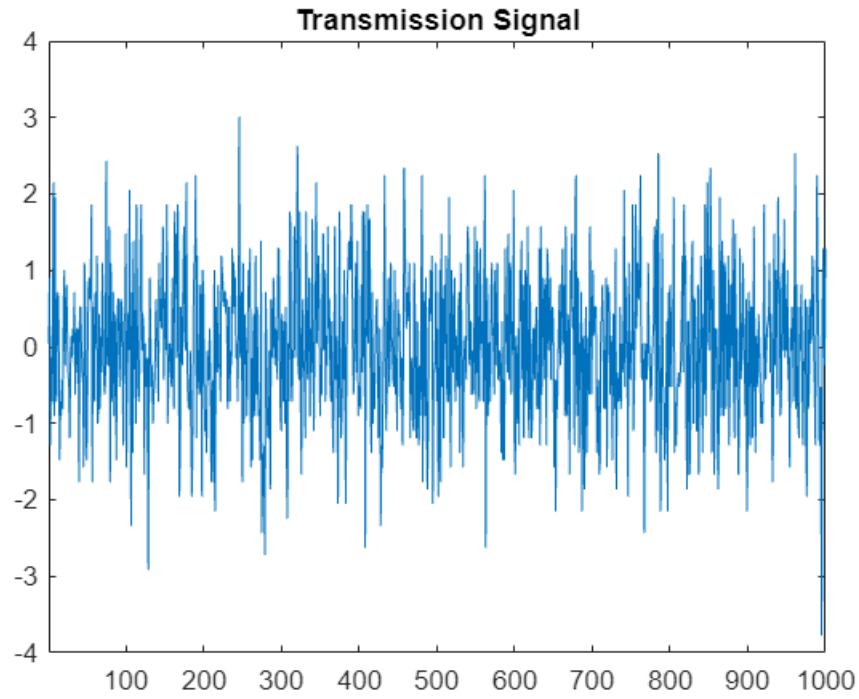
```
SQNR for 16 Level Encoder is: 24.037754 dB
```

(a) 4.24

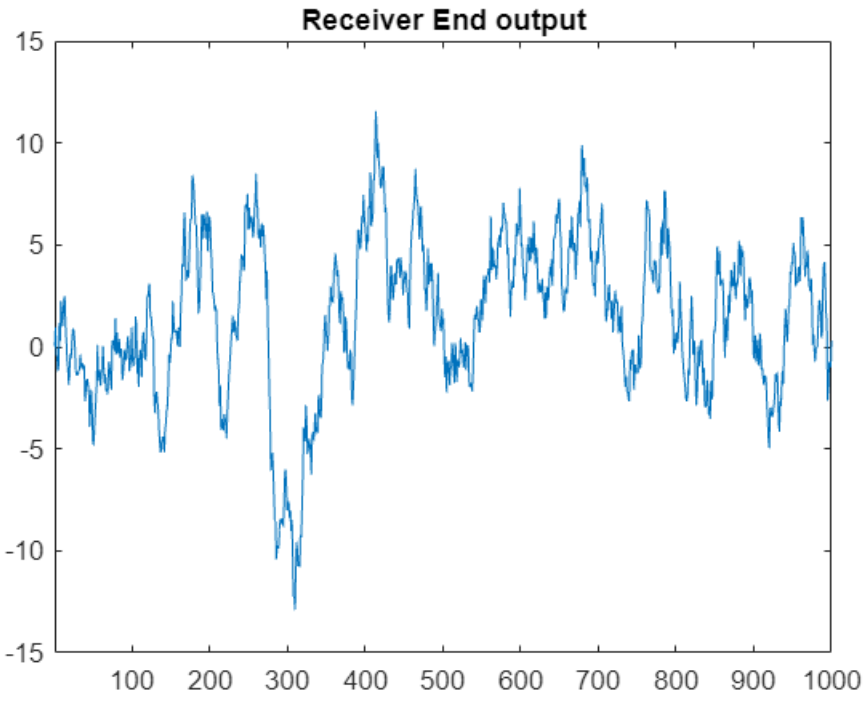
```
t = 1:1:1000;
elements = 1000;
X = zeros(1,elements);
w = randn(1,elements);
X(1) = 0;
for i = 2:elements
    X(i) = 0.98*X(i-1)+w(i);
end
mu =255;
n=256;

[sqnr_dpcm, X_quan_dpcm,TX_end] = d_pcm(X,n);

figure;
plot(X);
title('Original Signal')
xlim([1, 1000]);
```




```
figure;
plot(TX_end);
title('Transmission Signal')
xlim([1, 1000]);
```

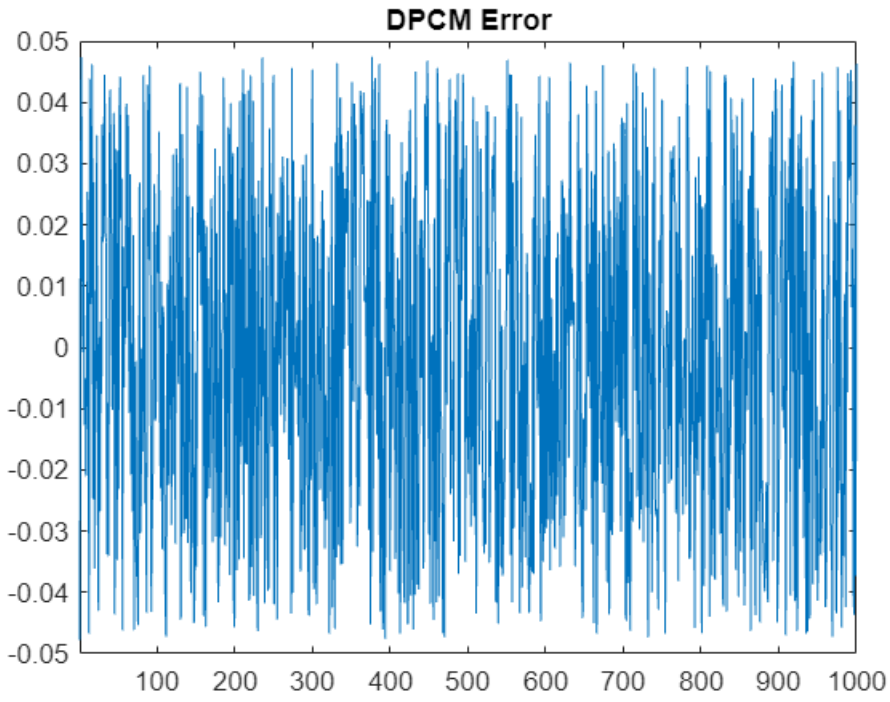



```
figure;
plot(X_quan_dpcm)
title('Receiver End output')
xlim([1, 1000]);
```



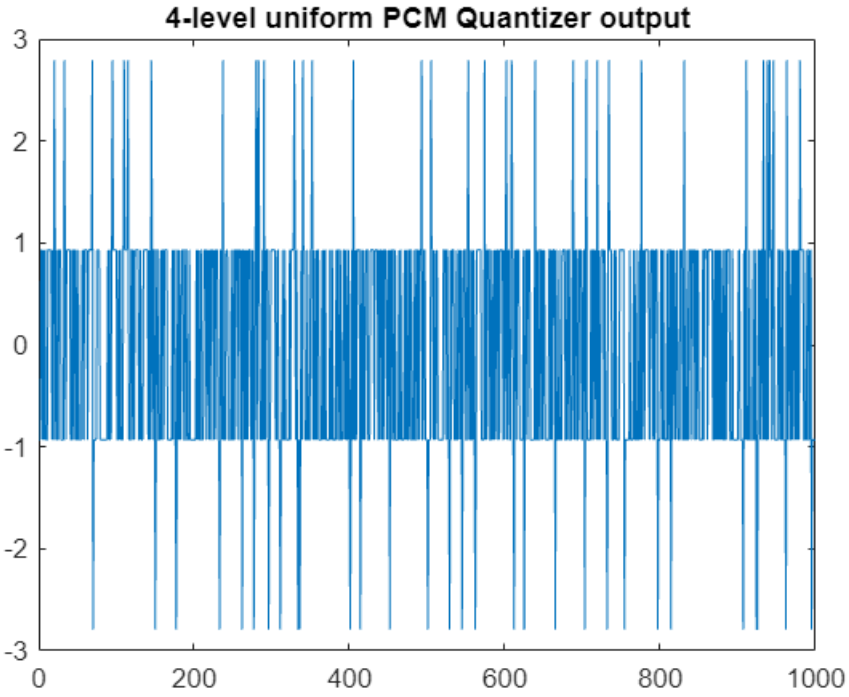

```
figure;
plot(X-X_quan_dpcm)
```

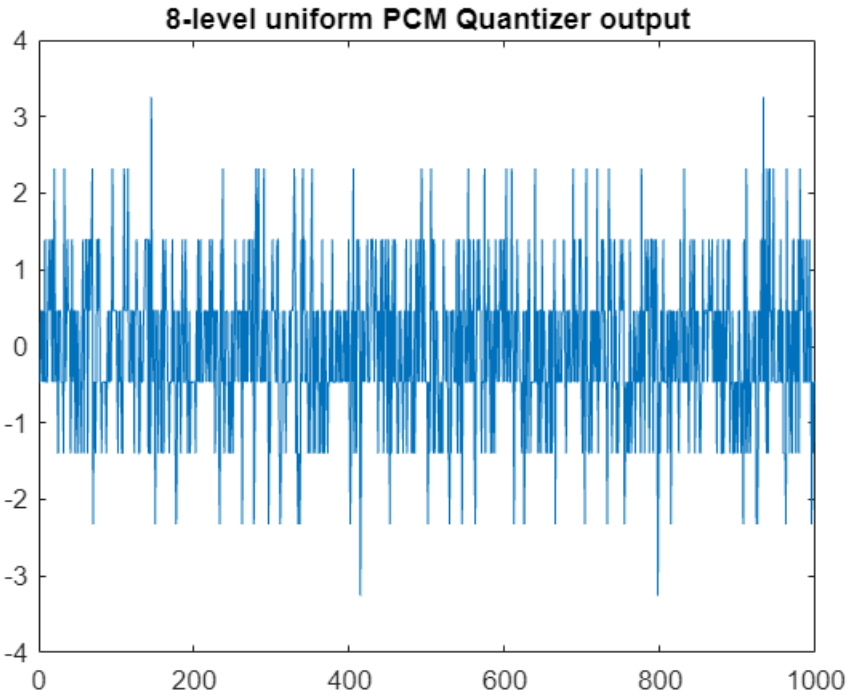
```
title('DPCM Error')
xlim([1, 1000]);
```

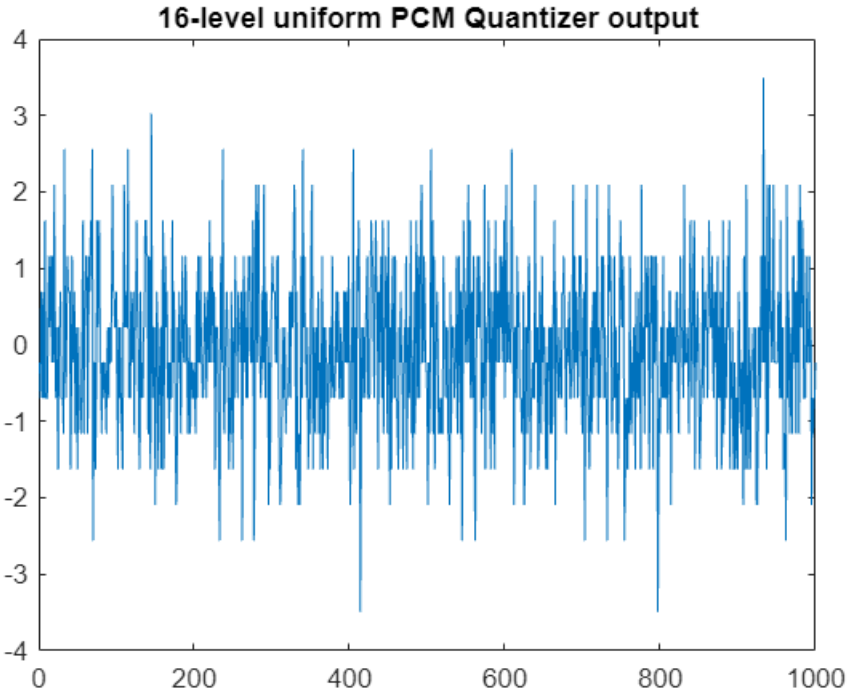


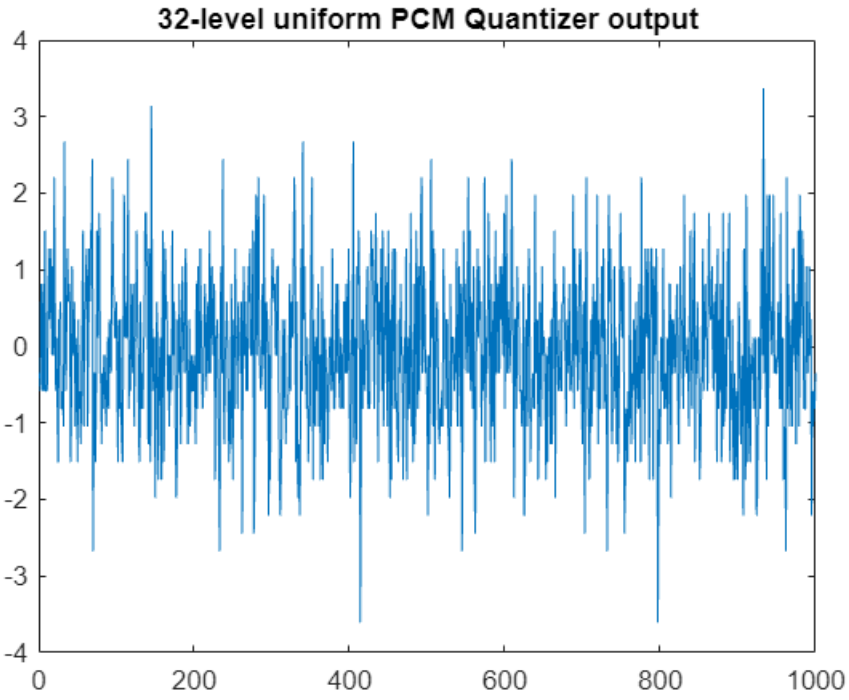
(b) 4.18

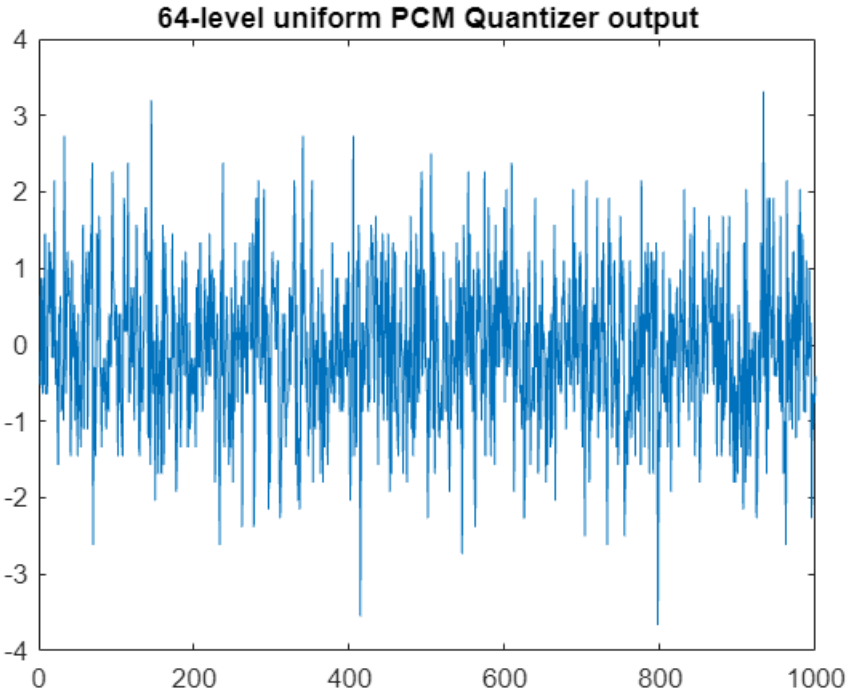
```
t = 1:1:1000;
elements = 1000;
gaussian = randn(1,elements);
level = [4,8,16,32,64,128,256,512,1024];
sqnrl = size(level);
sqnrl_mu_law = size(level);
a_quan = zeros(length(level),length(gaussian));
u = 255;

for i = 1:length(level)
    [sqnrl(i),a_quan,code] = u_pcm(gaussian,level(i));
    figure;
    plot(a_quan)
    title([num2str(level(i)),'-level uniform PCM Quantizer output'])
end
```

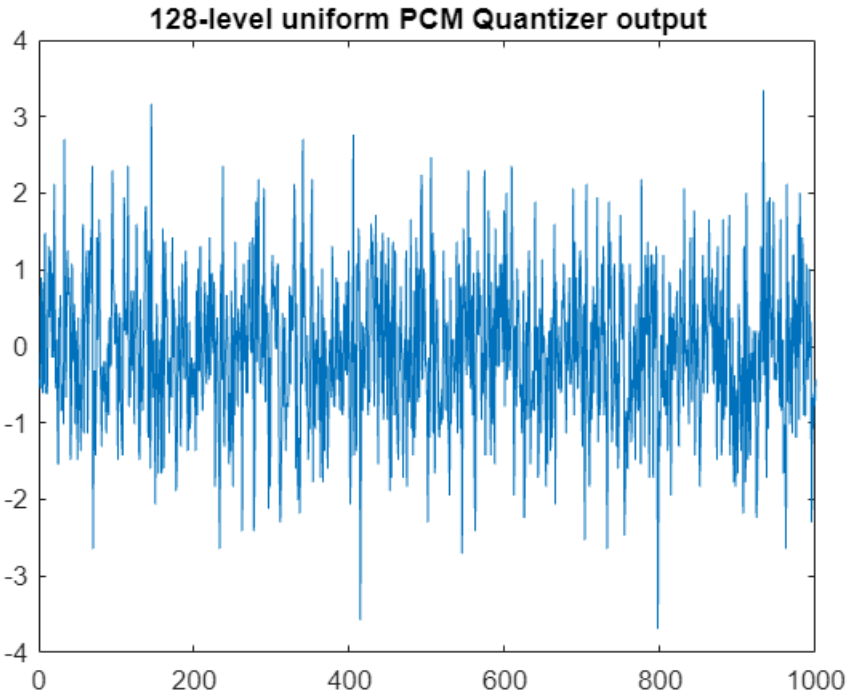

4-level uniform PCM Quantizer output
3
2
1
0
-1
-2
-3
0
200
400
600
800
1000

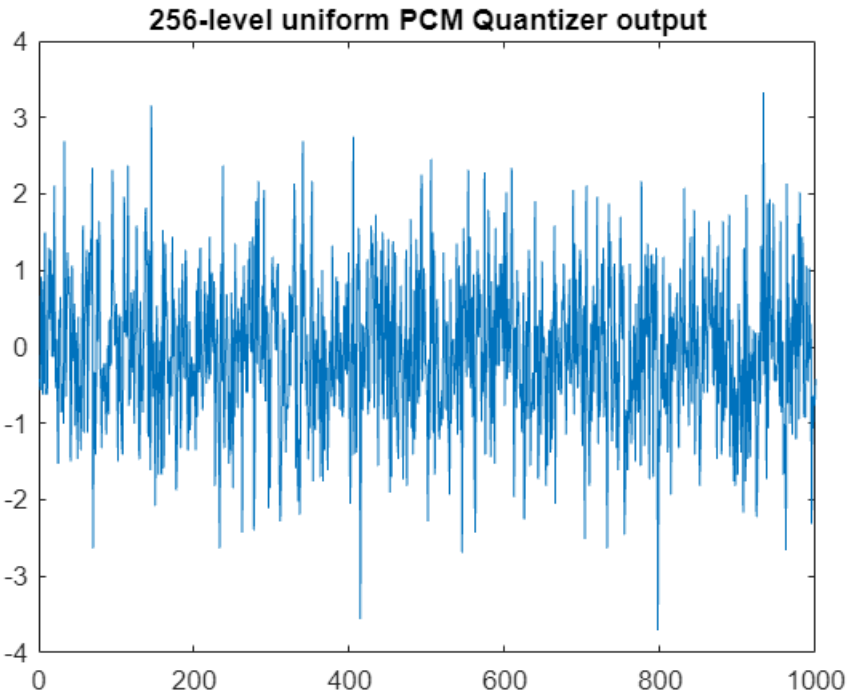

8-level uniform PCM Quantizer output
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000

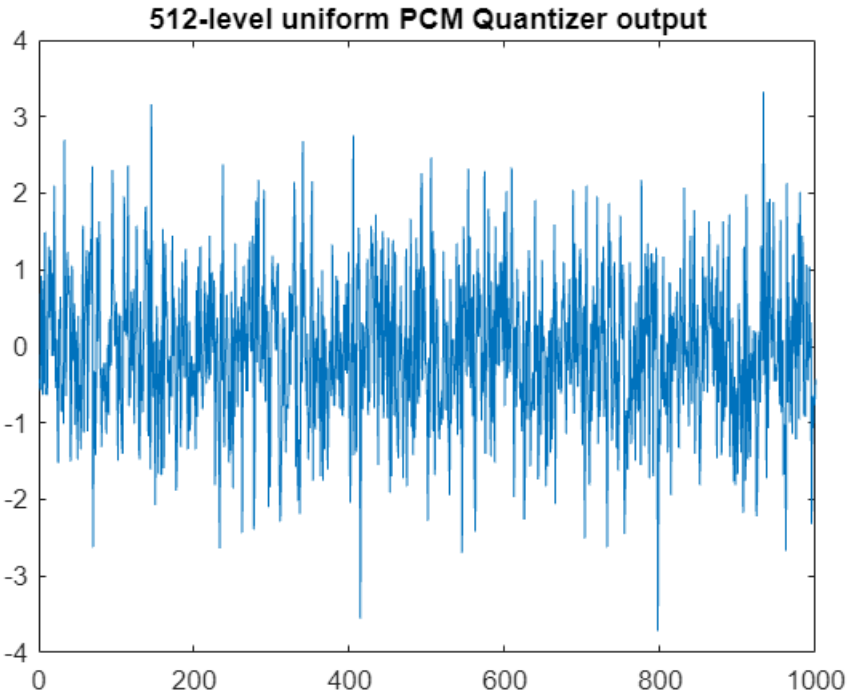

16-level uniform PCM Quantizer output
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000

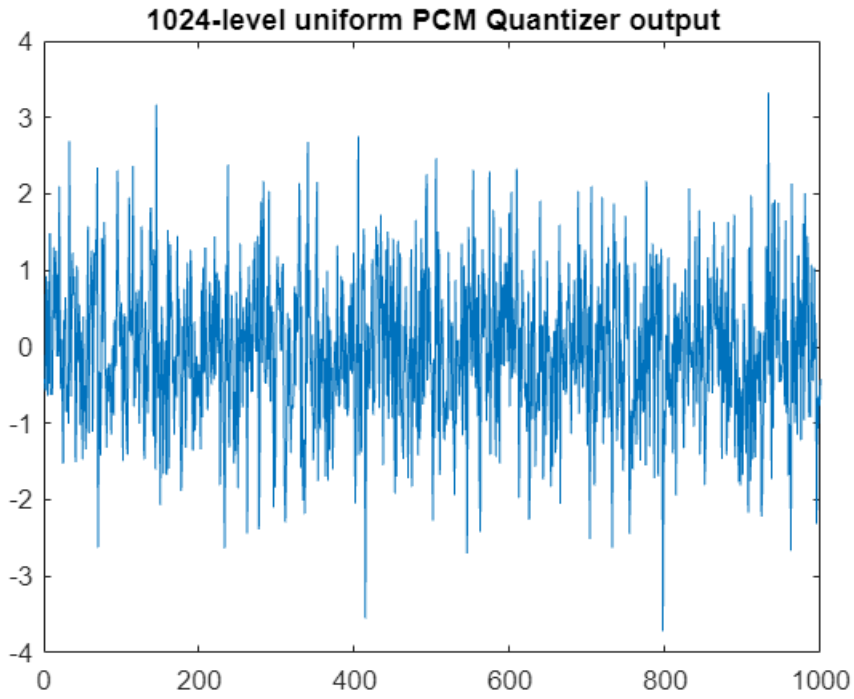

32-level uniform PCM Quantizer output
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000


64-level uniform PCM Quantizer output
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000

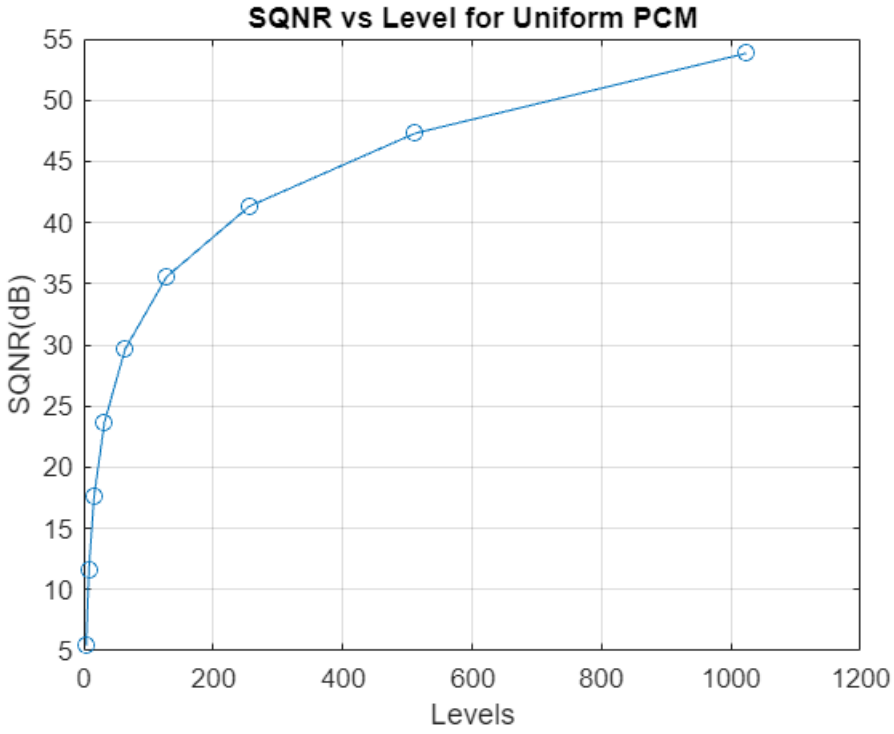

128-level uniform PCM Quantizer output
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000


256-level uniform PCM Quantizer output
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000


512-level uniform PCM Quantizer output
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000

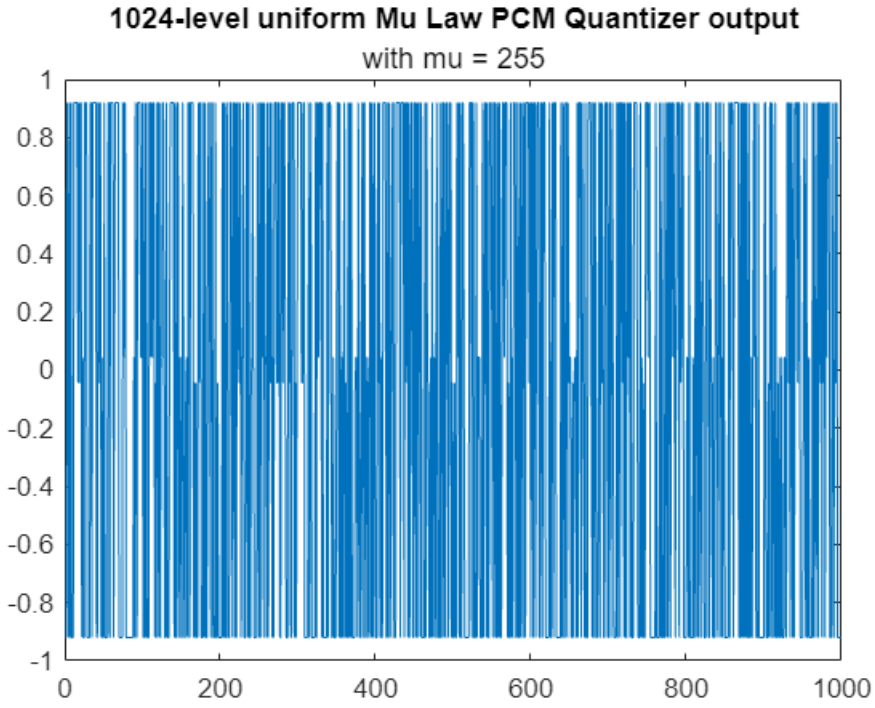


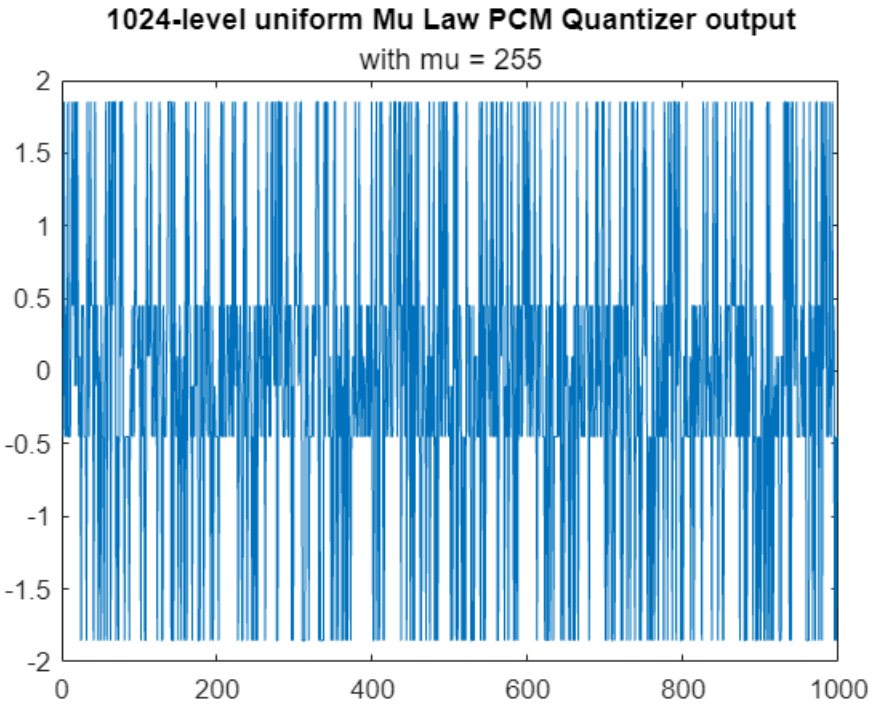

```
figure;
plot(level,sqnrl,'o-')
xlabel('Levels');
ylabel('SQNR(dB)');
title('SQNR vs Level for Uniform PCM');
grid on;
```

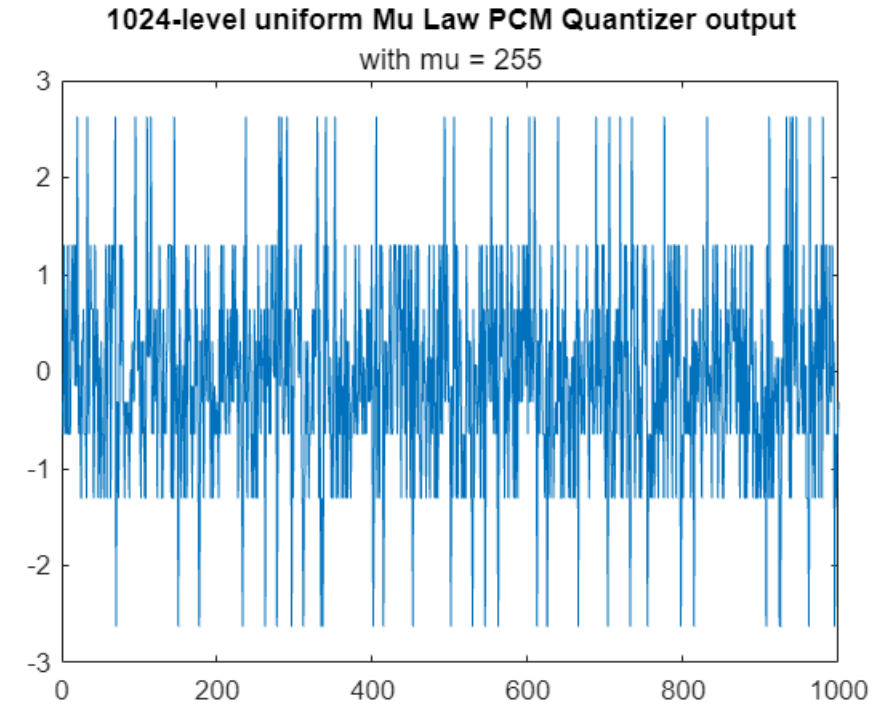


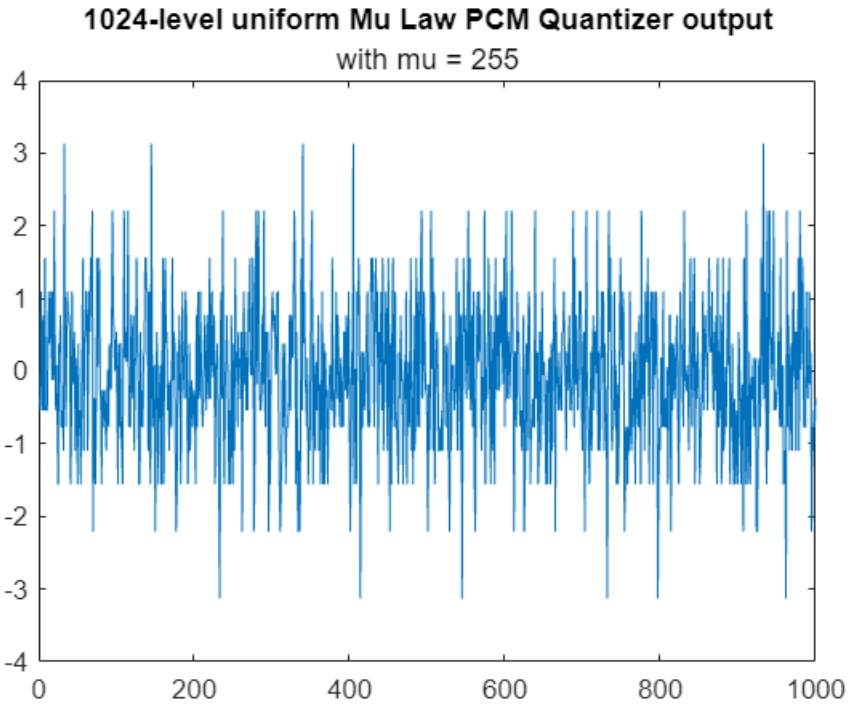
(b) 4.21

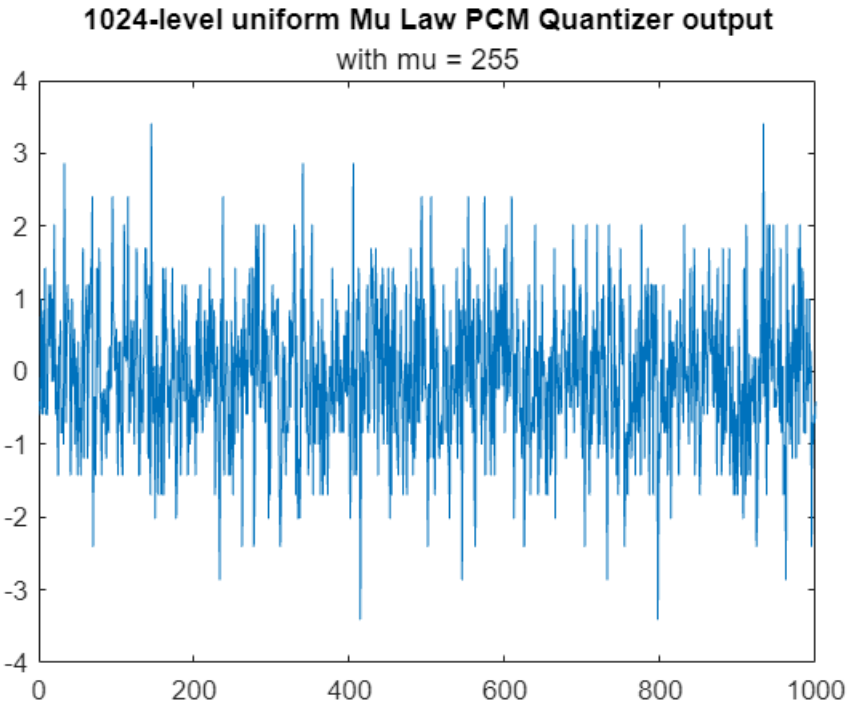
```
for j=1:length(level)
    [sqnrl_mu_law(j),a_quan,code] = mula_pcm(gaussian,level(j),u);
    figure;
    plot(a_quan)
    title([num2str(level(i)),'-level uniform Mu Law PCM Quantizer output'],'with mu
= 255')
end
```

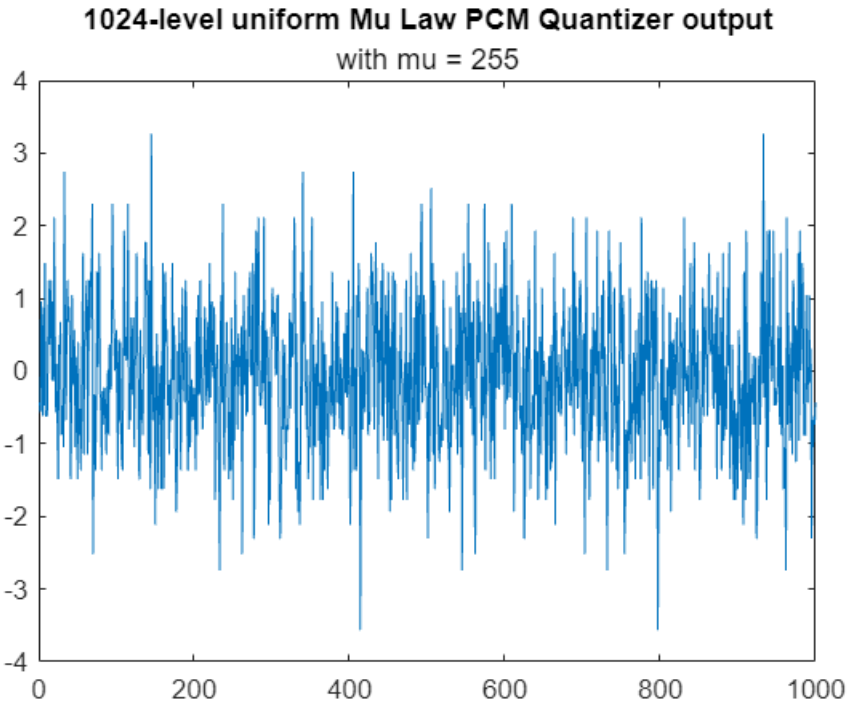


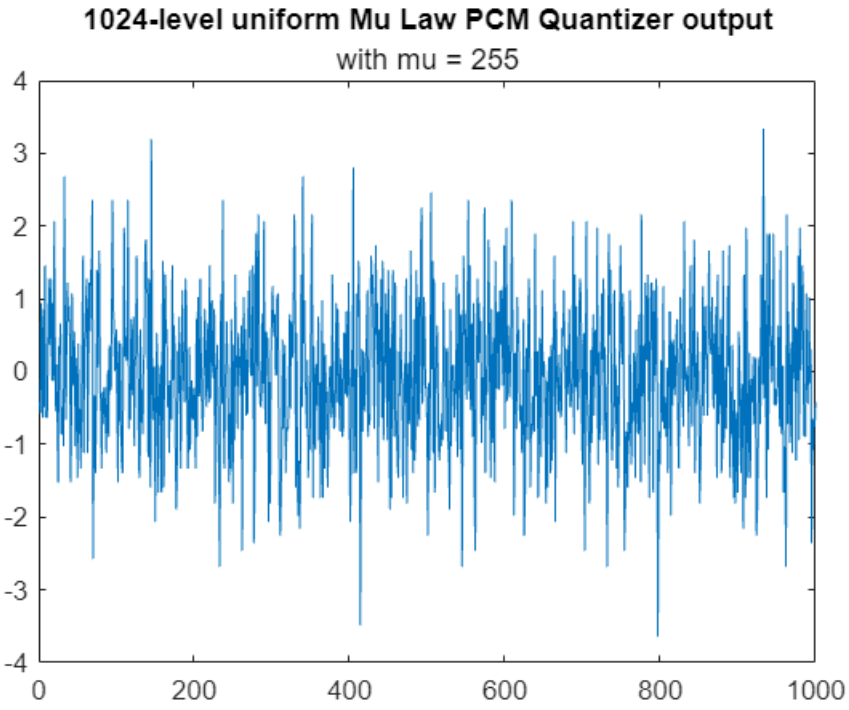

1024-level uniform Mu Law PCM Quantizer output
with mu = 255
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
200
400
600
800
1000

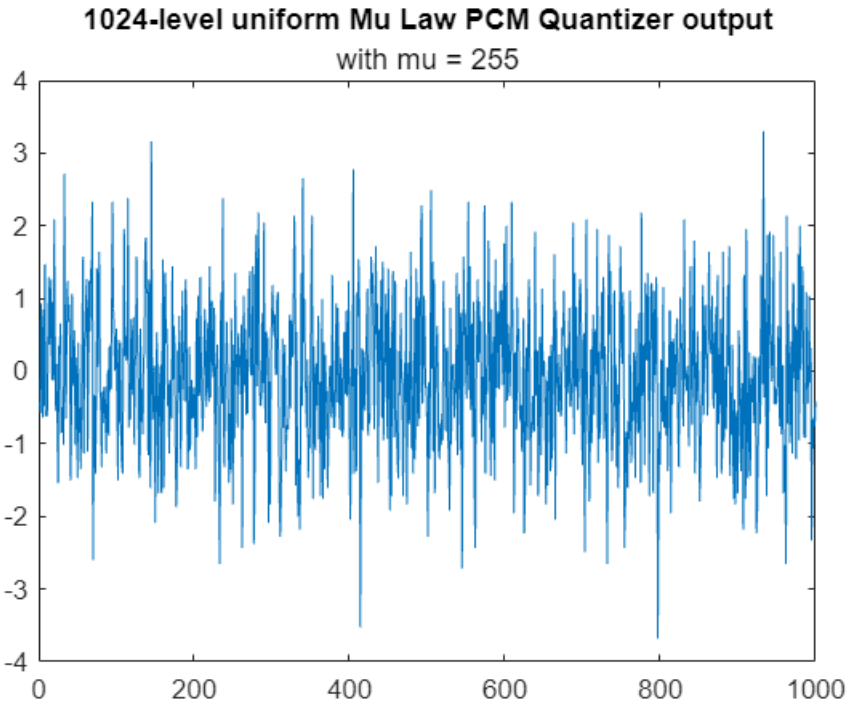

1024-level uniform Mu Law PCM Quantizer output
with mu = 255
3
2
1
0
-1
-2
-3
0
200
400
600
800
1000

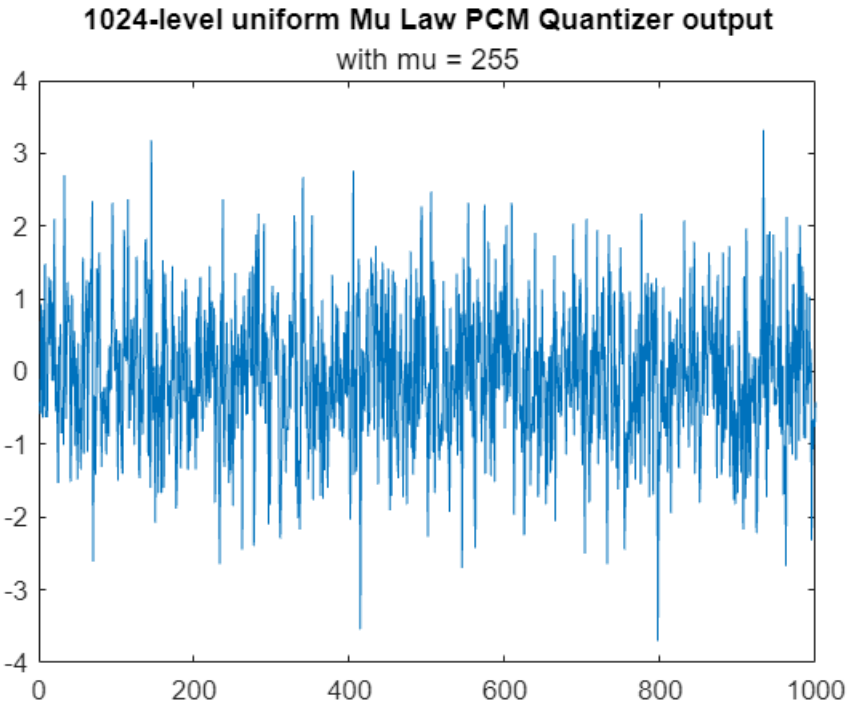

1024-level uniform Mu Law PCM Quantizer output
with mu = 255
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000


1024-level uniform Mu Law PCM Quantizer output
with mu = 255
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000

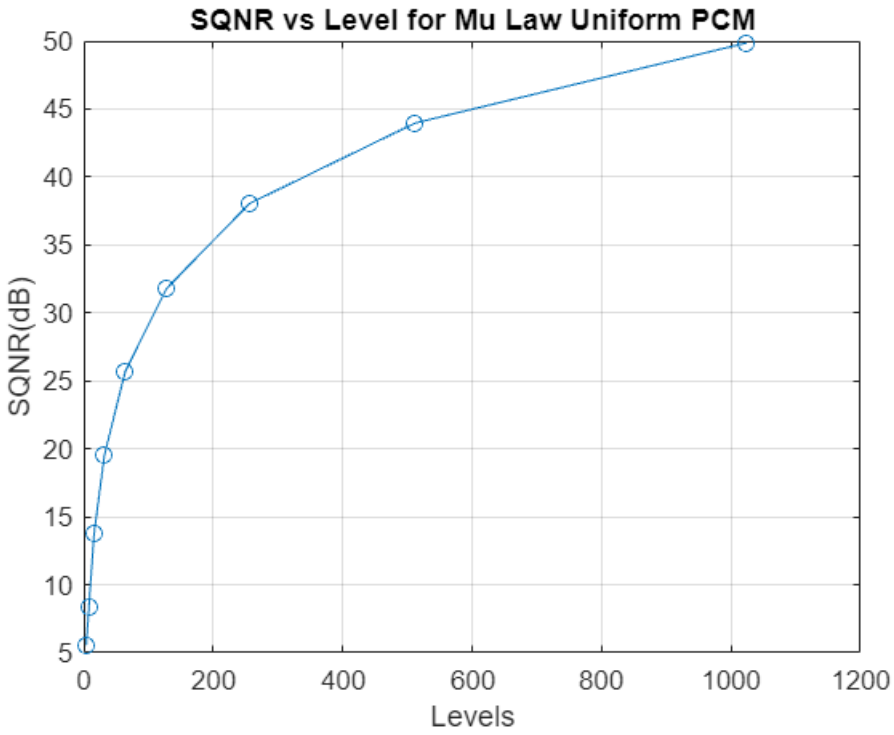

1024-level uniform Mu Law PCM Quantizer output
with mu = 255
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000


1024-level uniform Mu Law PCM Quantizer output
with mu = 255
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000






```
figure;
plot(level,sqnrl_mu_law,'o-')
xlabel('Levels');
ylabel('SQNR(dB)');
title('SQNR vs Level for Mu Law Uniform PCM');
grid on;
```



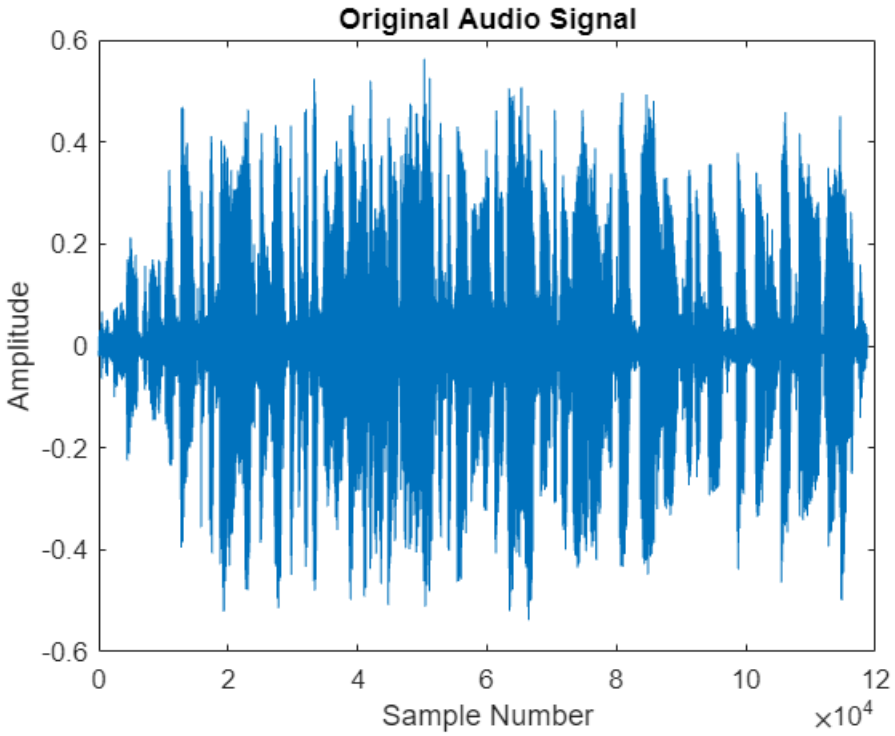
Conclusion:

- The SQNR increases as the number of quantization levels increases from 8 to 16, improving the quality of the encoded signal.
- The 8-level PCM quantizer produces higher quantization error compared to the 16-level PCM quantizer.
- DPCM reduces redundancy in the signal by predicting future values, which creates a more efficient encoding process than regular PCM encoding.
- Mu-law compression improves SQNR for signals with larger dynamic ranges compared to uniform PCM, especially at higher quantization levels.

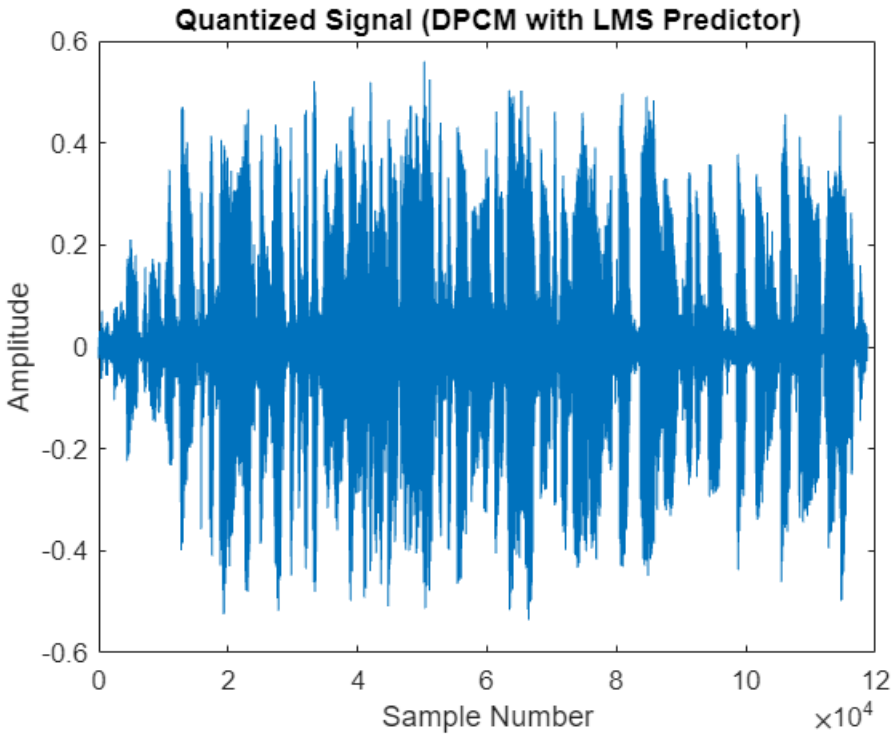
Question 2

```
[audio_signal, fs] = audioread('sample.wav');
audio_signal = audio_signal';

bits = 8;
levels = 2^bits;
p = 5;
mu = 0.0001;


N = length(audio_signal);
e = zeros(1, N);
x_hat = zeros(1, N);
w = zeros(p, N);
quantized_signal = zeros(1, N);
```

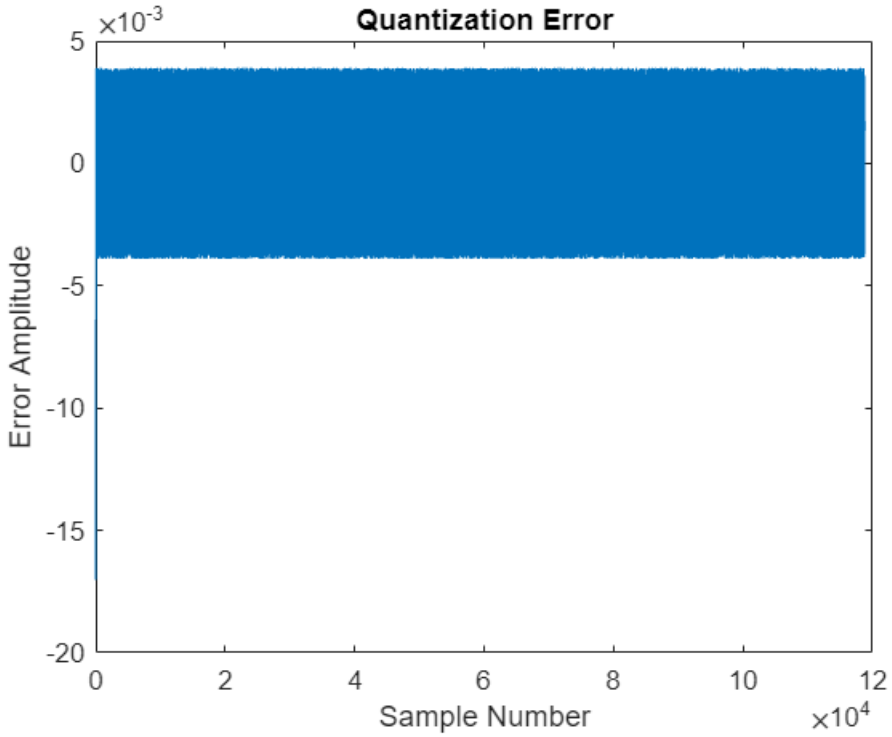
```
for k = 1:p
    w(k, 1) = 0;
end

for n = p+1:N

    x_hat(n) = sum(w(:, n)' .* audio_signal(n-1:-1:n-p));

    e(n) = audio_signal(n) - x_hat(n);

    delta = 2 / levels;
    quantized_error = round(e(n) / delta) * delta;
    quantized_signal(n) = x_hat(n) + quantized_error;

    for k = 1:p
        w(k, n+1) = w(k, n) + (mu / (sum(audio_signal(n-1:-1:n-p).^2))) *
audio_signal(n-k) * e(n);
    end
end

quantization_error = audio_signal - quantized_signal;
sqnr = 20 * log10(norm(audio_signal) / norm(quantization_error));

figure;
plot(audio_signal);
title('Original Audio Signal');
xlabel('Sample Number');
ylabel('Amplitude');
```



```
figure;
plot(quantized_signal);
title('Quantized Signal (DPCM with LMS Predictor)');
xlabel('Sample Number');
ylabel('Amplitude');
```




```
figure;
plot(quantization_error);
title('Quantization Error');
xlabel('Sample Number');
ylabel('Error Amplitude');
```




```
disp(['SQNR for mu = ', num2str(mu), ' is: ', num2str(sqnr), ' dB']);
```

```
SQNR for mu = 0.0001 is: 35.4043 dB
```

```
final_w = w(:, end);
disp('Final DPCM Predictor Coefficients (w):');
```

```
Final DPCM Predictor Coefficients (w):
```

```
disp(final_w);
```

```
    0.5917
    0.3135
    0.2326
   -0.0767
   -0.1575
```

Conclusion:

- The DPCM with Least Mean Square Predictor provides efficient compression.
- It has a good trade off between accuracy and compression.
- The quantization error is minimized through adaptive coefficient updates in the LMS predictor.
- The longer the signal, the more accurate is the LMS predictor since we have more value to update the coefficients with.
- The larger the prediction order, the more accurate is the LMS predictor since it takes into account multiple previous values, and makes a better judgement of the future values.

Differential PCM Encoding Function

```
function [sqnr_dpcm,X_quan_dpcm,TX_end] = d_pcm(X,n)
    d = zeros(1,length(X)); % Difference between signal and predicted value
    dq = zeros(1,length(X)); % Quantized difference signal
    mq = zeros(1,length(X)); % Predicted value for each sample

    D = (max(X)-min(X))/n;
    q = D.*(0:n-1);
    q = q-((n-1)/2)*D;

    for k = 1:length(X)
        if k==1
            d(k) = X(k); % First Sample
        else
            d(k) = X(k) - mq(k-1);
        end

        %Quantization
        if d(k)>max(X)
            d(k) = max(X);
        elseif d(k)<min(X)
            d(k) = min(X);
        end

        q_val = 1;
        for i = 1:n
            if (q(i)-D/2<=d(k)) && (d(k)<=q(i)+D/2)
                q_val = i;
            end
        end
        dq(k) = q(q_val);

        if k == 1
            mq(k) = dq(k);
        else
            mq(k) = dq(k) + mq(k-1);
        end
    end
    TX_end = dq;

    %Reconstruct
    R_out = zeros(1,length(X));
    for k = 1:length(X)
        if k ==1
            R_out(k) = dq(k);
        else
            R_out(k) = dq(k) + R_out(k-1);
        end
    end
```

```
    X_quan_dpcm = R_out;
    sqnr_dpcm = 20*log10(norm(X)/norm(X-R_out));
end
```

Mu Law PCM Encoding Function

```
function [sqnr,a_quan,code] = mula_pcm(a,n,mu)
    [y ,maximum]=mulaw(a,mu);
    [sqnr,y_q,code]=u_pcm(y,n);
    a_quan=invmulaw(y_q,mu);
    a_quan=maximum*a_quan;
    sqnr=20*log10(norm(a)/norm( a-a_quan));
end
```

Inverse Mu Law Function

```
function x = invmulaw(y,mu)
    x=(((1 +mu).^(abs(y))-1 )./mu).*sign(y);
end
```

Mu Law Function

```
function [y,a] = mulaw(x,mu)
    a=max(abs(x));
    y=(log(1 +mu*abs(x/a))./log(1 +mu)).*sign(x);
end
```

Uniform PCM Encoding Function

```
function [ sqnr, a_quan,code,b_quan] = u_pcm(a,n)
    amax=max(abs(a));
    a_quan=a/amax;
    b_quan=a_quan;
    d=2/n;
    q=d.*[0:n-1];
    q=q-((n-1 )/2) *d;
    for i=1 :n
        a_quan(find((q(i)-d/2 <= a_quan) & (a_quan <= q(i)+d/2)))=
q(i).*ones(1,length(find((q(i)-d/2 <= a_quan) & (a_quan <= q(i)+d/2))));
        b_quan(find( a_quan==q(i) ))=(i-1 ).*ones(1,length(find( a_quan==q(i) )));
    end
    a_quan=a_quan * amax;
    nu=ceil(log2(n));
    code=zeros(length(a),nu);
    for i=1 :length(a)
        for j=nu:-1 :0
            if ( fix(b_quan(i)/(2^j)) == 1)
                code(i,(nu-j)) = 1;
                b_quan(i) = b_quan(i) - 2^j;
            end
```

```
        end
    end
    sqnr=20*log10(norm(a)/norm(a-a_quan));
end
```