clear all

clc

close all

%preprocessing of the audio data

nBits = 8;

nchannels = 1;

nID = -1;

Fs = 200000;

Fc = 4000;

time = 5;

% recObj = audiorecorder(Fs,nBits,nchannels,nID);

% recObj.StartFcn = 'disp(''Start speaking.'')';

% recObj.StopFcn = 'disp(''End of recording.'')';

% recordblocking(recObj,time);

% x = getaudiodata(recObj);

% y = lowpass(x,Fc,Fs);

load ('x.mat');

x1 = x;

x1\_hist = x1;

figure;

plot(1:length(x1),x1);

title('Speech Audio Data');

xlabel('Data Points');

ylabel('Audio Signal Amplitude');

grid on;

x1 = nonzeros(x1);

figure;

plot(1:length(x1),x1)

title('Optimized Speech Audio Data');

xlabel('Data Points');

ylabel('Audio Signal Amplitude');

grid on;

y1 = lowpass(x1,Fc,Fs);

y1 = normalize(x1,'range',[-1,1]);

figure;

h1 = histogram(x1,'normalization','probability');

title('Normalized Histogram to Estimate PDF');

grid on;

pdf\_x1 = h1.Values;

x1 = linspace(h1.BinLimits(1),h1.BinLimits(2),h1.NumBins);

pol1 = polyfit(x1,pdf\_x1,10);

polx1 = linspace(h1.BinLimits(1),h1.BinLimits(2));

figure;

plot(polx1,polyval(pol1,polx1));

title('Fitted and Normalized Polynomial');

xlabel('x1');

ylabel('f\_x(x1)');

grid on

for i = 1:11

if(pol1(i) < 0)

pol1(i) = 0;

end

end

%Part I Question 2

N = [16 64 128]; %number of quantization levels

sqnr1\_uni\_vec = zeros(1,3);

sqnr1\_nonuni\_vec = zeros(1,3);

distortion1\_uni\_vec = zeros(1,3);

distortion1\_nonuni\_vec = zeros(1,3);

sqnr1\_uni\_vec\_abs = zeros(1,3);

sqnr1\_nonuni\_abs\_vec = zeros(1,3);

distortion1\_nonuni\_abs\_vec = zeros(1,3);

distortion1\_uni\_abs\_vec = zeros(1,3);

for nn = 1:3

%UNIFORM QUANTIZATION

a = zeros(1,N(nn)+1);

a(1) = -1;

a(N(nn)+1) = 1;

log\_vec = zeros(1,length(a));

uniqua\_error\_sum1 = 0;

uniqua\_error\_sum1\_abs = 0;

MSE\_y1 = 0;

for i = 1:N(nn)+1

a(i) = a(1) + (a(N(nn)+1)-a(1))/N(nn)\*(i-1);

end

level\_dif1 = a(2)-a(1);

for j = 1:N(nn)

x\_quan1(j) = (a(j) + a(j+1))/2;

end

for ally = 1:length(x1)

log\_vec = y1(ally)- a < level\_dif1;

MSE\_y1 = MSE\_y1 + y1(ally).^2;

uniqua\_error\_sum1 = uniqua\_error\_sum1 + (y1(ally) - x\_quan1(first\_one(log\_vec))).^2;

uniqua\_error\_sum1\_abs = uniqua\_error\_sum1\_abs + abs(y1(ally) - x\_quan1(first\_one(log\_vec)));

end

%error calculation for uniform quantizer

D1 = uniqua\_error\_sum1/length(x1);

D1\_abs = uniqua\_error\_sum1\_abs/length(x1);

P1 = MSE\_y1/length(x1);

distortion1\_uni\_vec(nn) = D1;

sqnr1\_uni = P1/D1;

sqnr1\_uni\_abs = P1/D1\_abs;

distortion1\_uni\_abs\_vec(nn) = D1\_abs;

sqnr1\_uni\_vec(nn) = sqnr1\_uni;

sqnr1\_uni\_vec\_abs(nn) = sqnr1\_uni\_abs;

%NONUNIFORM QUANTIZATION

a1\_nonqua = a; %initializing with uniform quantizer ai's

x1\_nonqua = zeros(N(nn),1);

y1\_nonqua = zeros(length(y1),1);

%integral of x times probability density function of x

numer\_pdf1= @(x\_poly1) pol1(1)\*(x\_poly1.^10)+pol1(2)\*(x\_poly1.^9)+pol1(3)\*(x\_poly1.^8)+pol1(4)\*(x\_poly1.^7)+pol1(5)\*(x\_poly1.^6)+pol1(6)\*(x\_poly1.^5)+pol1(7)\*(x\_poly1.^4)+pol1(8)\*(x\_poly1.^3)+pol1(9)\*(x\_poly1.^2)+pol1(10)\*(x\_poly1)+pol1(11);

%integral of probability density function of x

denom\_x\_pdf1 = @(x\_poly1) (x\_poly1.\*(pol1(1)\*(x\_poly1.^10)+pol1(2)\*(x\_poly1.^9)+pol1(3)\*(x\_poly1.^8)+pol1(4)\*(x\_poly1.^7)+pol1(5)\*(x\_poly1.^6)+pol1(6)\*(x\_poly1.^5)+pol1(7)\*(x\_poly1.^4)+pol1(8)\*(x\_poly1.^3)+pol1(9)\*(x\_poly1.^2)+pol1(10)\*(x\_poly1)+pol1(11)));

D1\_nonqua = 0.8;

%calculates xi's using the ratio of integrals over two consecutive ai levels

while D1\_nonqua > 0.002 %loops until distortion changes more than 0.001 threshold value. Stops if distortion does not change more than 0.001

%calculates xi's using the ratio of x\*pdf integral and pdf integral

for i= 1:N(nn)

x1\_nonqua(i) = integral(denom\_x\_pdf1,a1\_nonqua(i),a1\_nonqua(i+1))/integral(numer\_pdf1,a1\_nonqua(i),a1\_nonqua(i+1));

end

%ai's are midpoints of two consecutive xi's

for j = 2:N(nn)

a1\_nonqua(j)=(x1\_nonqua(j)+x1\_nonqua(j-1))/2;

end

for j = 1:length(y1)

for lev = 1:N(nn)

if y1(j) > a1\_nonqua(lev) && y1(j) < a1\_nonqua(lev+1)

y1\_nonqua(j) = x1\_nonqua(lev);

end

end

end

D1\_nonqua = mean((y1-y1\_nonqua).^2);

P1\_nonqua = mean((y1).^2);

sqnr1\_nonuni\_vec(nn) = P1\_nonqua/D1\_nonqua;

distortion1\_nonuni\_vec(nn) = D1\_nonqua;

end

%NONUNIFORM QUANTIZATION WITH ABSOLUTE VALUE

a1\_init\_abs = a; %initializing with uniform quantizer ai's

x1\_i\_hat = zeros([1,N(nn)]);

y1\_nonqua\_abs = zeros(length(y1),1);

abs1\_integral = @(x1\_i\_hat, amax, amin) integral(@(x\_poly1) abs(x\_poly1-x1\_i\_hat).\*(pol1(1)\*(x\_poly1.^10)+pol1(2)\*(x\_poly1.^9)+pol1(3)\*(x\_poly1.^8)+pol1(4)\*(x\_poly1.^7)+pol1(5)\*(x\_poly1.^6)+pol1(6)\*(x\_poly1.^5)+pol1(7)\*(x\_poly1.^4)+pol1(8)\*(x\_poly1.^3)+pol1(9)\*(x\_poly1.^2)+pol1(10)\*(x\_poly1)+pol1(11)), amax, amin);

quantizer\_level = 0;

%divides the quantization level intervals to small points and

%tries to find the optimal point that gives smallest integral

%(quantization error)

while quantizer\_level < N(nn)

for j = 1:length(x1\_i\_hat)

sub\_interv = a1\_init\_abs(j):0.01:a1\_init\_abs(j+1);

integ = zeros([1,length(sub\_interv)]);

for k = 1:length(sub\_interv)

integ(k)= abs1\_integral(sub\_interv(k), a1\_init\_abs(j), a1\_init\_abs(j+1));

end

%finds the index of the optimal quantization level that

%minimizes the integral and locates that level

[Y,I] = min(integ);

x1\_i\_hat(j) = sub\_interv(I);

end

%ai's are middle point of optimized quantization level xi's

for i = 2:N(nn)

a1\_init\_abs(i) = (x1\_i\_hat(i-1) + x1\_i\_hat(i))\*0.5;

end

quantizer\_level = quantizer\_level + 1;

end

for j = 1:length(y1)

for lev = 1:N(nn)

if y1(j) > a1\_init\_abs(lev) && y1(j) < a1\_init\_abs(lev+1)

y1\_nonqua\_abs(j) = x1\_i\_hat(lev);

end

end

end

D1\_nonqua\_abs = mean(abs(y1 - y1\_nonqua\_abs));

P1\_nonqua\_abs = mean((y1).^2);

sqnr1\_nonuni\_abs\_vec(nn) = P1\_nonqua\_abs/D1\_nonqua\_abs;

distortion1\_nonuni\_abs\_vec(nn) = D1\_nonqua\_abs;

end

%second speech signal

load voice.mat

x = [x;x;x(1:199998)];

x2 = x;

x2\_hist = x2;

nBits = 8;

nchannels = 1;

nID = -1;

Fs = 200000;

Fc = 4000;

time = 5;

% recObj = audiorecorder(Fs,nBits,nchannels,nID);

% recObj.StartFcn = 'disp(''Start speaking.'')';

% recObj.StopFcn = 'disp(''End of recording.'')';

% recordblocking(recObj,time);

% x2 = getaudiodata(recObj);

% y2 = lowpass(x2,Fc,Fs);

figure;

plot(1:length(x2),x2);

title('Speech Audio Data of Second Speech');

xlabel('Data Points');

ylabel('Audio Signal Amplitude');

grid on;

x2 = nonzeros(x2);

figure;

plot(1:length(x2),x2)

title('Optimized Speech Audio Data of Second Speech');

xlabel('Data Points');

ylabel('Audio Signal Amplitude');

grid on;

y2 = lowpass(x2,Fc,Fs);

y2 = normalize(y2,'range',[-1,1]);

figure;

h2 = histogram(x2,'normalization','probability');

title('Normalized Histogram to Estimate PDF of Second Speech');

grid on;

pdf\_x2 = h2.Values;

x2 = linspace(h2.BinLimits(1),h2.BinLimits(2),h2.NumBins);

pol2 = polyfit(x2,pdf\_x2,10);

polx2 = linspace(h2.BinLimits(1),h2.BinLimits(2));

figure;

plot(polx2,polyval(pol2,polx2));

title('Fitted and Normalized Polynomial of Second Speech');

xlabel('x2');

ylabel('f\_x(x2)');

grid on

for i = 1:11

if(pol2(i) < 0)

pol2(i) = 0;

end

end

%Part I Question 2

N = [16 64 128]; %number of quantization levels

sqnr2\_uni\_vec = zeros(1,3);

sqnr2\_nonuni\_vec = zeros(1,3);

sqnr2\_nonuni\_abs\_vec = zeros(1,3);

distortion2\_uni\_vec = zeros(1,3);

distortion2\_nonuni\_vec = zeros(1,3);

distortion2\_nonuni\_abs\_vec = zeros(1,3);

mse2\_nonuni\_vec = zeros(1,3);

for nn = 1:3

%UNIFORM QUANTIZATION

a = zeros(1,N(nn)+1);

a(1) = -1;

a(N(nn)+1) = 1;

log\_vec = zeros(1,length(a));

uniqua\_error\_sum2 = 0;

uniqua\_error\_sum2\_abs = 0;

nonuniqua\_error\_sum2 = 0;

nonuniqua\_error\_sum2\_abs = 0;

MSE\_y2 = 0;

MSE\_y2\_nonuni = 0;

for i = 1:N(nn)+1

a(i) = a(1) + (a(N(nn)+1)-a(1))/N(nn)\*(i-1);

end

level\_dif2 = a(2)-a(1);

for j = 1:N(nn)

x\_quan2(j) = (a(j) + a(j+1))/2;

end

for ally = 1:length(x2)

log\_vec = y2(ally)- a < level\_dif2;

MSE\_y2 = MSE\_y2 + y2(ally).^2;

uniqua\_error\_sum2 = uniqua\_error\_sum2 + (y2(ally) - x\_quan2(first\_one(log\_vec))).^2;

uniqua\_error\_sum2\_abs = uniqua\_error\_sum2\_abs + abs(y2(ally) - x\_quan2(first\_one(log\_vec)));

end

%error calculation for uniform quantizer

D2 = uniqua\_error\_sum2/length(x2);

D2\_abs = uniqua\_error\_sum2\_abs/length(x2);

P2 = MSE\_y2/length(x2);

distortion2\_uni\_vec(nn) = D2;

sqnr2\_uni = P2/D2;

sqnr2\_uni\_abs = P2/D2\_abs;

sqnr2\_uni\_vec(nn) = sqnr2\_uni;

sqnr2\_uni\_vec\_abs(nn) = sqnr2\_uni\_abs;

%NONUNIFORM QUANTIZATION

a2\_nonqua = a; %initializing with uniform quantizer ai's

x2\_nonqua = zeros(N(nn),1);

y2\_nonqua = zeros(length(y2),1);

D2\_nonqua = 0.8; %initial distortion values

%integral of x times probability density function of x

denom\_pdf2= @(x\_poly2) pol2(1)\*(x\_poly2.^10)+pol2(2)\*(x\_poly2.^9)+pol2(3)\*(x\_poly2.^8)+pol2(4)\*(x\_poly2.^7)+pol2(5)\*(x\_poly2.^6)+pol2(6)\*(x\_poly2.^5)+pol2(7)\*(x\_poly2.^4)+pol2(8)\*(x\_poly2.^3)+pol2(9)\*(x\_poly2.^2)+pol2(10)\*(x\_poly2)+pol2(11);

%integral of probability density function of x

numer\_x\_pdf2 = @(x\_poly2) (x\_poly2.\*(pol2(1)\*(x\_poly2.^10)+pol2(2)\*(x\_poly2.^9)+pol2(3)\*(x\_poly2.^8)+pol2(4)\*(x\_poly2.^7)+pol2(5)\*(x\_poly2.^6)+pol2(6)\*(x\_poly2.^5)+pol2(7)\*(x\_poly2.^4)+pol2(8)\*(x\_poly2.^3)+pol2(9)\*(x\_poly2.^2)+pol2(10)\*(x\_poly2)+pol2(11)));

%calculates xi's using the ratio of integrals over two consecutive ai levels

while D2\_nonqua > 0.002 %loops until distortion changes more than 0.002 threshold value. Stops if distortion does not change more than 0.001

%calculates xi's using the ratio of x\*pdf integral and pdf integral

%condiional expectation

for i= 1:N(nn)

x2\_nonqua(i) = integral(numer\_x\_pdf2,a2\_nonqua(i),a2\_nonqua(i+1))/integral(denom\_pdf2,a2\_nonqua(i),a2\_nonqua(i+1));

end

%ai's are midpoints of two consecutive xi's

for j = 2:N(nn)

a2\_nonqua(j)=(x2\_nonqua(j)+x2\_nonqua(j-1))/2;

end

%locates the quantization level

for j = 1:length(y2)

for lev = 1:N(nn)

if y2(j) > a2\_nonqua(lev) && y2(j) < a2\_nonqua(lev+1)

y2\_nonqua(j) = x2\_nonqua(lev);

end

end

end

D2\_nonqua = mean((y2-y2\_nonqua).^2);

P2\_nonqua = mean((y2).^2);

sqnr2\_nonuni\_vec(nn) = P2\_nonqua/D2\_nonqua;

distortion2\_nonuni\_vec(nn) = D2\_nonqua;

%NONUNIFORM QUANTIZATION WITH ABSOLUTE VALUE

a2\_init\_abs = a; %initializing with uniform quantizer ai's

x2\_i\_hat = zeros([1,N(nn)]);

y2\_nonqua\_abs = zeros(length(y2),1);

%the integral for the mean absolute error

abs2\_integral = @(x2\_i\_hat, amax, amin) integral(@(x\_poly2) abs(x\_poly2-x2\_i\_hat).\*(pol2(1)\*(x\_poly2.^10)+pol2(2)\*(x\_poly2.^9)+pol2(3)\*(x\_poly2.^8)+pol2(4)\*(x\_poly2.^7)+pol2(5)\*(x\_poly2.^6)+pol2(6)\*(x\_poly2.^5)+pol2(7)\*(x\_poly2.^4)+pol2(8)\*(x\_poly2.^3)+pol2(9)\*(x\_poly2.^2)+pol2(10)\*(x\_poly2)+pol2(11)), amax, amin);

quantizer\_level = 0;

%divides the quantization level intervals to small points and

%tries to find the optimal point that gives smallest integral

%(quantization error)

while quantizer\_level < N(nn)

for j = 1:length(x2\_i\_hat)

%divides the intervals to small points

sub\_interv = a2\_init\_abs(j):0.01:a2\_init\_abs(j+1);

%calculates the integral (quantization error)

integ = zeros([1,length(sub\_interv)]);

for k = 1:length(sub\_interv)

integ(k)= abs2\_integral(sub\_interv(k), a2\_init\_abs(j), a2\_init\_abs(j+1));

end

%finds the index of the optimal quantization level that

%minimizes the integral and locates that level

[Y,I] = min(integ);

x2\_i\_hat(j) = sub\_interv(I);

end

%ai's are middle point of optimized quantization level xi's

for lev = 2:N(nn)

a2\_init\_abs(lev) = (x2\_i\_hat(lev-1) + x2\_i\_hat(lev))\*0.5;

end

quantizer\_level = quantizer\_level + 1;

end

for j = 1:length(y2)

for a = 1:N(nn)

if y2(j) > a2\_init\_abs(a) && y2(j) < a2\_init\_abs(a+1)

y2\_nonqua\_abs(j) = x2\_i\_hat(a);

end

end

end

D2\_nonqua\_abs = mean(abs(y2 - y2\_nonqua\_abs));

P2\_nonqua\_abs = mean((y2).^2);

sqnr2\_nonuni\_abs\_vec(nn) = P2\_nonqua\_abs/D2\_nonqua\_abs;

distortion2\_nonuni\_abs\_vec(nn) = D2\_nonqua\_abs;

%plots the 2d joinlty pdf of the consecutive source samples

h3 = histogram2(x1\_hist,x2\_hist,"Normalization","Probability")

end

end

function one\_index = first\_one(test\_vec)

for i = 1:length(test\_vec)

if test\_vec(i) == 1;

one\_index = i;

break

end

end

end