HOMEWORK #1 SOLUTIONS

1. (a) Proof:

$$\begin{aligned}
\int x p_x(x)dx &= \int \frac{x}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{x^2}{2\sigma_x^2}} dx \\
&= -\frac{\sigma_x^2}{\sqrt{2\pi\sigma_x^2}} \int \frac{x}{-\sigma_x^2} e^{-\frac{x^2}{2\sigma_x^2}} dx \\
&= -\frac{\sigma_x^2}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{x^2}{2\sigma_x^2}} \qquad \text{since} \qquad \frac{d}{dx} e^{-\frac{x^2}{2\sigma_x^2}} = -\frac{x}{\sigma_x^2} e^{-\frac{x^2}{2\sigma_x^2}} \\
&= -\sigma_x^2 p_x(x)
\end{aligned}$$

□

(b) Proof:

$$\begin{aligned}
\int_a^\infty p_x(x)dx &= \int_a^\infty \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{x^2}{2\sigma_x^2}} dx \\
&= \int_{\frac{a}{\sqrt{2\sigma_x^2}}}^\infty \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-t^2} \sqrt{2\sigma_x^2} dt \qquad \text{using} \qquad t = \frac{x}{\sqrt{2\sigma_x^2}} \\
&= \frac{1}{\sqrt{\pi}} \int_{\frac{a}{\sqrt{2\sigma_x^2}}}^\infty e^{-t^2} dt \\
&= \frac{1}{2} \text{erfc}\left(a/\sqrt{2\sigma_x^2}\right)
\end{aligned}$$

□

(c) The Matlab code I used to generate the Lloyd-Max quantizers is comprised of the following three files:

design.m

```
%%%%%%%%%%%%%%%%%%%%%
% Lloyd-Max Design %
%%%%%%%%%%%%%%%%%%%%%
sig2_x = 1;
L = 32;

% iterative Lloyd-Max design
fname = ['lloyd_max',num2str(L),'.mat'];
if exist(fname)==2,
  load(fname);
else
  y_hat = zeros(1,L);
  x_hat = zeros(1,L+1);
  err = 1;                                  % initial error
  y_hat_1_old = -4*sqrt(sig2_x)-(err/log2(L))^1.1; % ad hoc initialization
  x_hat(1) = -inf;                          % default
  x_hat(L+1) = inf;                         % default
  fzero_options = optimset('Display','off');  % for more info try 'iter'
  while abs(err)/sqrt(sig2_x) > 1e-3;
    y_hat(1) = y_hat_1_old + (err/log2(L))^1.1;       % ad hoc update rule
    y_hat_1_old = y_hat(1);
    for k=1:L-1,
      if k>1 x_hat_init = 2*y_hat(k)-x_hat(k); else x_hat_init = y_hat(k); end;
      x_hat(k+1) = fzero( inline('y-centroid(a,b,sig2)','b','y','a','sig2'),...
          x_hat_init,fzero_options,y_hat(k),x_hat(k),sig2_x);
                      % start search for x_hat(k+1) at y_hat(k)
      y_hat(k+1) = 2*x_hat(k+1)-y_hat(k);
    end;
    y_L = centroid(x_hat(L),x_hat(L+1),sig2_x);
```

```
    err = y_L-y_hat(L)
  end;
  save(fname,'L','sig2_x','x_hat','y_hat');
end;

figure(1);
xx = linspace(-4*sqrt(sig2_x),4*sqrt(sig2_x),1000);
plot(xx,exp(-xx.^2/2/sig2_x)/sqrt(2*pi*sig2_x),...
     y_hat,exp(-y_hat.^2/2/sig2_x)/sqrt(2*pi*sig2_x),'o',...
     x_hat,exp(-x_hat.^2/2/sig2_x)/sqrt(2*pi*sig2_x),'+');
title('input distribution, reconstruction values(o), and thresholds(+)')
grid on;
```

centroid.m

```
function y = centroid(a,b,sig2_x)

  y = -sqrt(sig2_x/2/pi)*( exp(-b^2/2/sig2_x)-exp(-a^2/2/sig2_x) )/...
        ( Qfnc(a,sig2_x) - Qfnc(b,sig2_x));
```

Qfnc.m

```
function y = Qfnx(x,sig2_x)

  y = erfc(x/sqrt(2*sig2_x))/2;
```

(d) Fig. 1 shows the Lloyd-Max quantizers for $L = 5$ and $L = 32$.

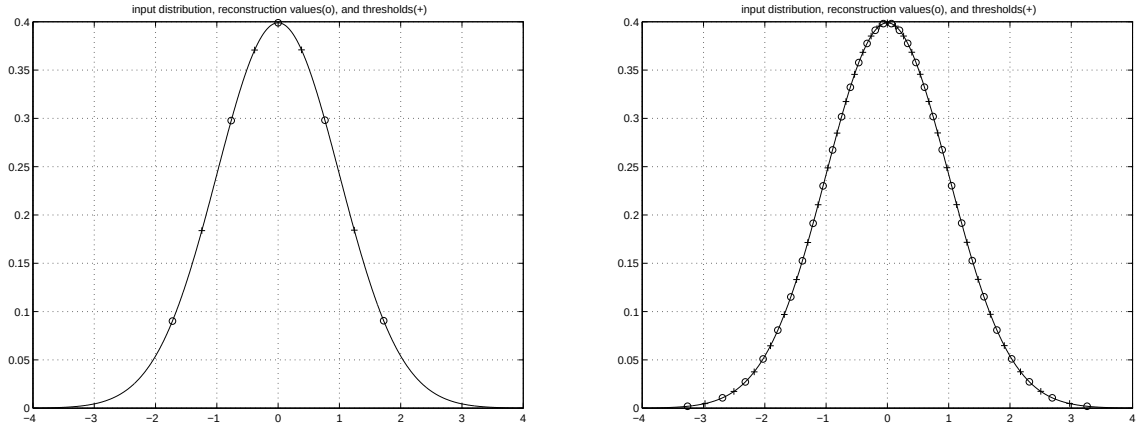

Figure 1: Lloyd-Max quantizers for $L = 5$ (left) and $L = 32$ (right).

2. (a) Experimentally, the LLoyd-Max mean-squared quantization error (MSQE) was found to be $\boxed{0.0024}$ using the code below (after `design.m`), though the number is a bit different for every experiment.

```
%%%%%%%%%%%%%%%%%%%%%%%%
% Lloyd-Max Analysis %
%%%%%%%%%%%%%%%%%%%%%%%%
x = randn(10000,1);                     % Gaussian random vector
M = length(x);
x = x*sqrt(sig2_x/var(x));

% quantize
y_lm = NaN*ones(M,1);
for k=1:L,                              % quantize
  y_lm( find((x>x_hat(k))&(x<x_hat(k+1))) ) = y_hat(k);
end;
q_lm = x-y_lm;                          % quantization error
var_q_lm= var(q_lm)
var_q_lm_theory = 2*pi*3^(3/2)*sig2_x/12/L^2
```

(b) We are asked to show

$$\left(\int_{-x_{\max}}^{x_{\max}} \sqrt[3]{p_x(x)}dx\right)^3 = \left(\int_{-\infty}^{\infty} \frac{1}{\sqrt[6]{2\pi\sigma_x^2}} e^{\frac{-x^2}{6\sigma_x^2}} dx\right)^3$$
$$= \left(\frac{\sqrt{2\pi \cdot 3\sigma_x^2}}{\sqrt[6]{2\pi\sigma_x^2}} \underbrace{\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi \cdot 3\sigma_x^2}} e^{\frac{-x^2}{2\cdot 3\sigma_x^2}} dx}_{=1} \right)^3$$
$$= 3^{3/2} \cdot 2\pi\sigma_x^2,$$

where we use the fact that $x_{\max} = \infty$ for Gaussian r.v. x. Then, using the equation

$$\sigma_q^2 = \frac{1}{12L^2}\left(\int_{-x_{\max}}^{x_{\max}} \sqrt[3]{p_x(x)}dx\right)^3 = \frac{3^{3/2} \cdot 2\pi\sigma_x^2}{12L^2},$$

we find that the theoretical LLoyd-Max MSQE is $\boxed{0.0027}$.

(c) The uniform quantizers designed under 20 *assumptions* of $x_{\max} \in (2\sigma_x, 5\sigma_x)$ generated the experimental MSQE shown in Fig. 2. The following code was used:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Uniform Design and Analysis %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% design and analyze uniform quantizers
x_max = linspace(2*sqrt(sig2_x),5*sqrt(sig2_x),20);
H_u = zeros(size(x_max));
var_q_u = zeros(size(x_max));
for i=1:length(x_max),
  x_hat_u = linspace(-x_max(i),x_max(i),L+1);  % middle bins
  y_hat_u = x_hat_u(1:L)+(x_hat_u(2)-x_hat_u(1))/2;
  x_hat_u(1) = -inf; x_hat_u(L+1) = inf;       % end points
  Delta = x_hat_u(3)-x_hat_u(2);
  y_u = NaN*ones(M,1);
  for k=1:L,                                   % quantize
    y_u( find((x>x_hat_u(k))&(x<x_hat_u(k+1))) ) = y_hat_u(k);
  end;
  q_u = x-y_u;
  var_q_u(i)= var(q_u);                        % experimental MSQE
  var_q_u_theory(i) = Delta^2/12;              % theoretical MSQE
  H_u(i) = calc_entropy(x_hat_u,sig2_x);       % entropy
end;
```

(d) The uniform quantizers designed under 20 *assumptions* of $x_{\max} \in (2\sigma_x, 5\sigma_x)$ generated the theoretical MSQE shown in Fig. 2 based on the following equation.

$$\sigma_q^2 = \frac{\Delta^2}{12} \quad \text{where} \quad \Delta = \frac{2x_{\max}}{L}.$$

(e) In Fig. 2, we compare experimental and theoretical MSQE for the Lloyd-Max and uniform quantizers. The important points are listed below.

- For the Lloyd-max quantizer, experimental and theoretical value of MSQE are close.
- Experimentally, we find that the Lloyd-max quantizer has lower MSQE than any of the uniform quantizers. This is expected since the LLoyd-max quantizer generates the minimum MSQE among all memoryless quantizers.
- When $x_{\max} > 3.5\sigma_x$, the theoretical and experimental versions of uniform quantizer MSQE agree. This is because the theory assumes that $x \in (-x_{\max}, x_{\max})$, which is reasonable when $x_{\max} > 3.5\sigma_x$ for Gaussian x. Since this assumption is not well satisfied for smaller values of $x_{\max}$, the theoretical and experimental values do not agree.
- MSQE increases when the assumption on $x_{\max}$ is small. This is a result of the fact that input values $|x| > x_{\max}$ are clipped to the endpoints y_1 and y_L. As $x_{\max}$ is made smaller, this quantizer "overloading" happens more often and results in larger error.

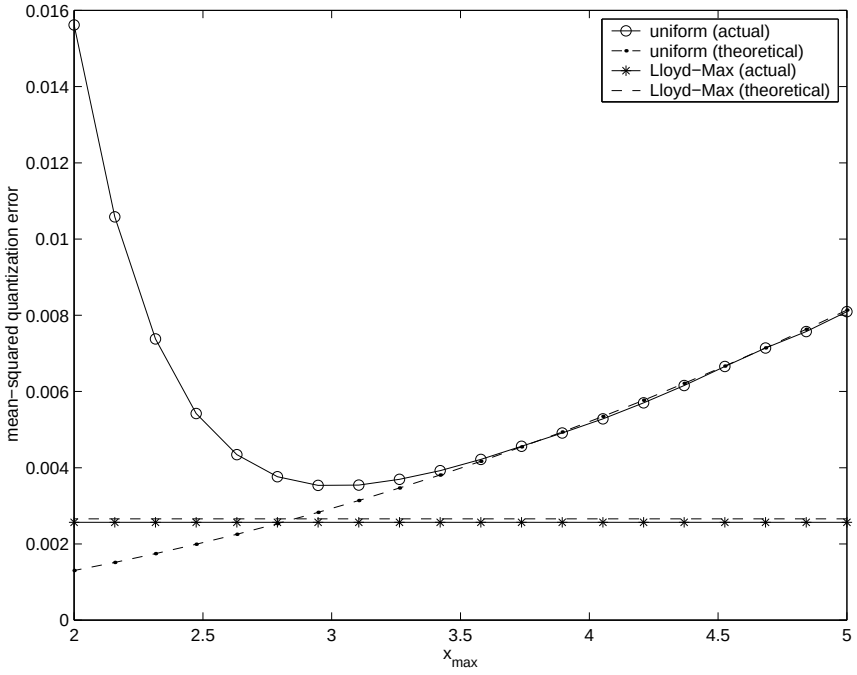

Figure 2: Theoretical and experimental mean-squared quantization error.

- The figure suggests that, for Gaussian input x, a large-L uniform quantizer should be designed under the assumption that $x_{\max} \approx 3\sigma_x$ for best MSQE performance.

3. (a) Output entropy is defined

$$H_y \;=\; -\sum_k P_k \log_2 P_k \qquad \text{for} \qquad P_k \;=\; \Pr\{-x_k \le x < x_{k+1}\}.$$

Assuming Gaussian x, P_k can be calculated exactly for a given set of thresholds $\{x_k\}$:

$$\begin{aligned} P_k &= \int_{x_k}^{x_{k+1}} p_x(x)dx \\ &= \int_{x_k}^{\infty} p_x(x)dx - \int_{x_{k+1}}^{\infty} p_x(x)dx \\ &= \frac{1}{2}\text{erfc}\Big(x_k/\sqrt{2\sigma_x^2}\Big) - \frac{1}{2}\text{erfc}\Big(x_{k+1}/\sqrt{2\sigma_x^2}\Big) \end{aligned}$$

For $L = 32$ Lloyd-Max thresholds, this yields $H_y = \boxed{4.73}$ bits. If we think of y as a member of an i.i.d. random process then the entropy rate of the random process would be 4.73 bits/sample.

(b) Repeating for the twenty different $L = 32$ uniform thresholds designed under different assumptions of $x_{\max}$ yields the H_y in Fig. 3.

(c) For large-L uniform quantizers we have

$$H_y \;=\; h_x - \frac{1}{2}\log_2(12\sigma_q^2)$$

under the assumption that the quantizer input obeys $x \in (-x_{\max}, x_{\max})$. With $h_x = \frac{1}{2}\log_2(2\pi e\,\sigma_x^2)$ and $\text{SNR} = 10\log_{10}(\sigma_x^2/\sigma_q^2)$,

$$\begin{aligned} H_y &= \frac{1}{2}\log_2(2\pi e\,\sigma_x^2) - \frac{1}{2}\log_2(12\sigma_q^2) \\ &= \frac{1}{2}\log_2(\pi e/6) + \frac{1}{2}\log_2(\sigma_x^2/\sigma_q^2) \\ &= \frac{1}{2}\log_2(\pi e/6) + \frac{1}{2}\log_2(10^{\text{SNR}/10}) \\ &= \boxed{\frac{1}{2}\log_2(\pi e/6) + \frac{\text{SNR}}{20}\log_2(10)} \\ &\approx 0.255 + 0.166\,\text{SNR}. \end{aligned}$$

For the optimal non-memoryless quantizer, the notes claim

$$H_y = h_x - \frac{1}{2}\log_2(2\pi e\,\sigma_q^2).$$

Using $h_x = \frac{1}{2}\log_2(2\pi e\,\sigma_x^2)$ and $\text{SNR} = 10\log_{10}(\sigma_x^2/\sigma_q^2)$ again,

$$\begin{aligned} H_y &= \frac{1}{2}\log_2(2\pi e\,\sigma_x^2) - \frac{1}{2}\log_2(2\pi e\,\sigma_q^2) \\ &= \frac{1}{2}\log_2(\sigma_x^2/\sigma_q^2) \\ &= \frac{1}{2}\log_2(10^{\text{SNR}/10}) \\ &= \boxed{\frac{\text{SNR}}{20}\log_2(10)} \\ &\approx 0.166\,\text{SNR}. \end{aligned}$$

(d) In Fig. 3, entropies of uniform and Lloyd-Max quantizer outputs are plotted versus theoretical and experimental values of SNR. Also plotted is the theoretical output entropy of a large-L (input constrained) uniform quantizer, and the minimum output entropy over all (possibly non-memoryless) quantizers. The following points should be noted:

- The experimental uniform trace suggests that, for Gaussian input x, a large-L uniform quantizer should be designed under the assumption that $x_{\max} > 3.5\sigma_x$ to reach the optimal tradeoff between entropy and SNR. (Though the figure seems to show no penalty for arbitrarily large assumptions of $x_{\max}$, it is expected that $x_{\max} \gg L\sigma_x$ would effectively use only the center quantization bin(s) and yield poor performance.)
- The theoretical uniform trace is invalid for SNR $>$ 24 dB due to the assumption $x \in (-x_{\max}, x_{\max})$. For SNR $<$ 24 dB, both experimental and theoretical uniform traces approach the memoryless bound.
- The experimental traces demonstrate that the entropy-per-SNR of a uniform quantizer designed under good assumption of $x_{\max}$ is lower than that of the Lloyd-Max quantizer, as expected since the uniform quantizer is optimal w.r.t. entropy-per-SNR.
- The optimal (non-memoryless) quantizer performance is 0.255 bits or 1.6 dB better than the optimal memoryless quantizer.

(a)-(b) The code used for encoding and decoding of block AQF and AQB appears below.
`adapt_quant_blk.m`

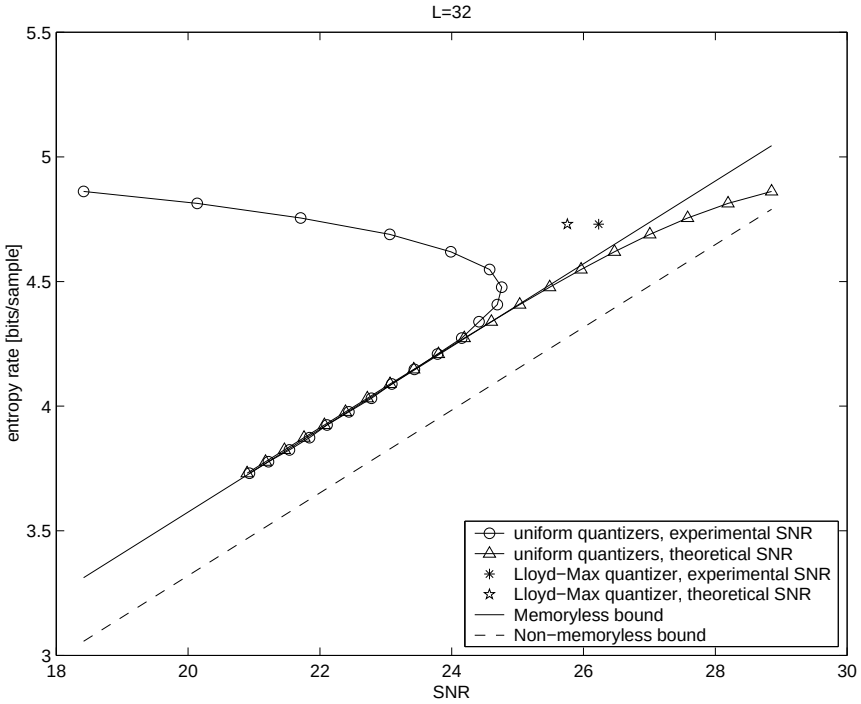

Figure 3: Entropy rates and bounds.

```
% this file simulates the global and local SNR for AQF and AQB

% parameters
inputname = 'sco_29.wav';                 % .wav input file
B = 6; % number of bits
N = 256;                                  % window length
phi = 4;                                  % quantizer factor

D = 16;                                   % sideinfo decimation factor
M_max = inf;                              % max number of input samples
stdv_x_max = 1;                           % maximum allowed stdv

store_on = 0;
  if store_on&findstr(inputname,'29'),
    error('wont store outputs from such a short file!');
  end;
  outputname_clean = 'sco_noaq.wav';
  outputname_aqf = ['sco_aqf',num2str(B),'_',num2str(N),'.wav'];
  outputname_aqb = ['sco_aqb',num2str(B),'_',num2str(N),'.wav'];


%---------------

% load file
[x,Fs,Nbits,Opts] = wavread(inputname);
M = min(length(x),M_max);
x = x(1:M);
var_x = var(x,1);

Lf = 2^(B-2);                             % number of positive levels
Lb = 2^(B-1);                             % number of positive levels

%---------------

%%%%%%%%%%%%%%%
% encode AQF %
%%%%%%%%%%%%%%%
stdv_f = zeros(ceil(M/D),1);
x_f = [x(1:N);x];                         % pad for variance estimation
stdv_f_min = stdv_x_max/2^D;              % smallest tolerated value
ptr = 1+N;
for i=1:ceil(M/D),                        % calculate variances
  stdv_f(i) = max(stdv_f_min, sqrt(sum(x_f(ptr-N+1:ptr).^2)/N) );
  ptr = ptr+D;
end;
clear x_f;
if max(stdv_f)>stdv_x_max, error('maximum stdv exceeded!'); end;
stdv_f = round((stdv_f/stdv_x_max)*2^D);

scale_f = ((stdv_f/2^D)*stdv_x_max)*phi/Lf;
y_f = zeros(M,1);
cnt = 0;
for i=1:M,                                % quantize
  cnt_cur = floor(100*((i-N)/2/M));
```

```
  if cnt_cur>cnt, cnt=cnt_cur; fprintf('%3d%%\b\b\b\b',cnt); end;
  y_f(i) = max(-Lf+1,min(Lf, round(x(i)/scale_f(floor((i-1)/D)+1)) ));
end;

%%%%%%%%%%%%%%%
% decode AQF %
%%%%%%%%%%%%%%%
scale_f = ((stdv_f/2^D)*stdv_x_max)*phi/Lf;
xt_f = zeros(M,1);
for i=1:M,
  xt_f(i) = y_f(i).*scale_f( floor((i-1)/D)+1 );
end

%%%%%%%%%%%%%%%
% encode AQB %
%%%%%%%%%%%%%%%
stdv_b = zeros(N+M,1);
stdv_b_min = eps;                           % smallest tolerated value
stdv_b(1:N+1) = stdv_x_max/Lb;              % initial condition
y_b = [ones(N,1)/phi*Lb;zeros(M,1)];        % AQB output for any constant signal
for i=N+[1:M],
  cnt_cur = 50+floor(100*((i-N)/2/M));
  if cnt_cur>cnt, cnt=cnt_cur; fprintf('%3d%%\b\b\b\b',cnt); end;
  y_b(i) = max(-Lb+1,min(Lb, round(x(i-N)/(stdv_b(i)*phi/Lb)) ));  % quantize
  stdv_b(i+1) = max(stdv_b_min,...
    sqrt(sum((y_b(i-N+1:i).*stdv_b(i-N+1:i)).^2)/N)*phi/Lb );
                                            % more exact way
end;
y_b = y_b(N+[1:M]);                         % unpad

%%%%%%%%%%%%%%%
% decode AQB %
%%%%%%%%%%%%%%%
y_b = [ones(N,1)/phi*Lb; y_b];              % pad
stdv_b = zeros(N+M,1);
stdv_b_min = eps;                           % smallest tolerated value
stdv_b(1:N+1) = stdv_x_max/Lb;              % initial condition
for i=N+[1:M],
   stdv_b(i+1) = max(stdv_b_min,...
    sqrt(sum((y_b(i-N+1:i).*stdv_b(i-N+1:i)).^2)/N)*phi/Lb );
end;
xt_b = y_b(N+[1:M]).*stdv_b(N+[1:M])*phi/Lb;
y_b = y_b(N+[1:M]);                         % unpad

%----------------

%compute error
N2 = 64;
e2_f = abs(x-xt_f);
SNR_aqf = -10*log10(mean(e2_f.^2)/mean(x.^2))

e2_b = abs(x-xt_b);
SNR_aqb = -10*log10(mean(e2_b.^2)/mean(x.^2))

%----------------

if store_on,
  scale = max(abs([xt_f(:);xt_b(:);x(:)]))*1.01;
  wavwrite(x/scale,Fs,Nbits,...
     ['/home/backhoe/schniter/audio_class/matlab/adapt_quant/sounds/',outputname_clean])
  wavwrite(xt_f/scale,Fs,Nbits,...
     ['/home/backhoe/schniter/audio_class/matlab/adapt_quant/sounds/',outputname_aqf])
  wavwrite(xt_b/scale,Fs,Nbits,...
     ['/home/backhoe/schniter/audio_class/matlab/adapt_quant/sounds/',outputname_aqb])
end;

%----------------
nvec = [1:M];
figure(1);
subplot(411);
  plot(nvec,x(nvec));
  title('input');
subplot(412);
  plot(D*floor((nvec-1)/D)+1,...
      stdv_f(floor((nvec-1)/D)+1)/2^D*stdv_x_max);
  title('AQF std-dev estimates');
subplot(413);
  plot(nvec,y_f(nvec));
  hold on;
  plot([min(nvec),max(nvec)],[1,1]*Lf,'b--',...
       [min(nvec),max(nvec)],-[1,1]*Lf,'b--');
  hold off;
  title('quantized AQF output');
subplot(414);
  plot(nvec,e2_f(nvec));
  title('absolute AQF error');
orient tall;

figure(2);
subplot(411);
  plot(nvec,x(nvec));
  title('input');
subplot(412);
  plot(nvec,stdv_b(N+nvec))
  title('AQB std-dev estimates');
subplot(413);
  plot(nvec,y_b(nvec));
  hold on;
  plot([min(nvec),max(nvec)],[1,1]*Lb,'r--',...
       [min(nvec),max(nvec)],-[1,1]*Lb,'r--');
  hold off;
  title('quantized AQB output');
subplot(414);
  plot(nvec,e2_b(nvec));
  title('absolute AQB error');
orient tall;
```

4. The processing of `sco_29.wav` gives the plots in Figures 4–7 and the SNRs below.

	AQF-256	AQF-1024	AQB-256	AQB-1024
SNR [dB]	21.6	19.3	26.0	21.0

   Note the following.

   - The standard-deviation estimates for AQF and AQB look very similar when the block size is the same.
   - The quantized ouputs are limited to ± 16 in AQF and ± 32 in AQB due to the bit rate limitations specified in the assignment.
   - The reconstuction error magnitude has a brief peak when the signal changes from soft to loud. This is because the standard-deviation estimates do not change quickly enough for the quantizer stepsize to increase from small to large, and the quantizer briefly overloads as a result.

5. Listening to the output of the various codecs, I thought that AQB-1024 gave the best sounding output. Though AQB-256 gives the lowest SNR, the quantization noise was "bursty" and more annoying than for AQB-1024.

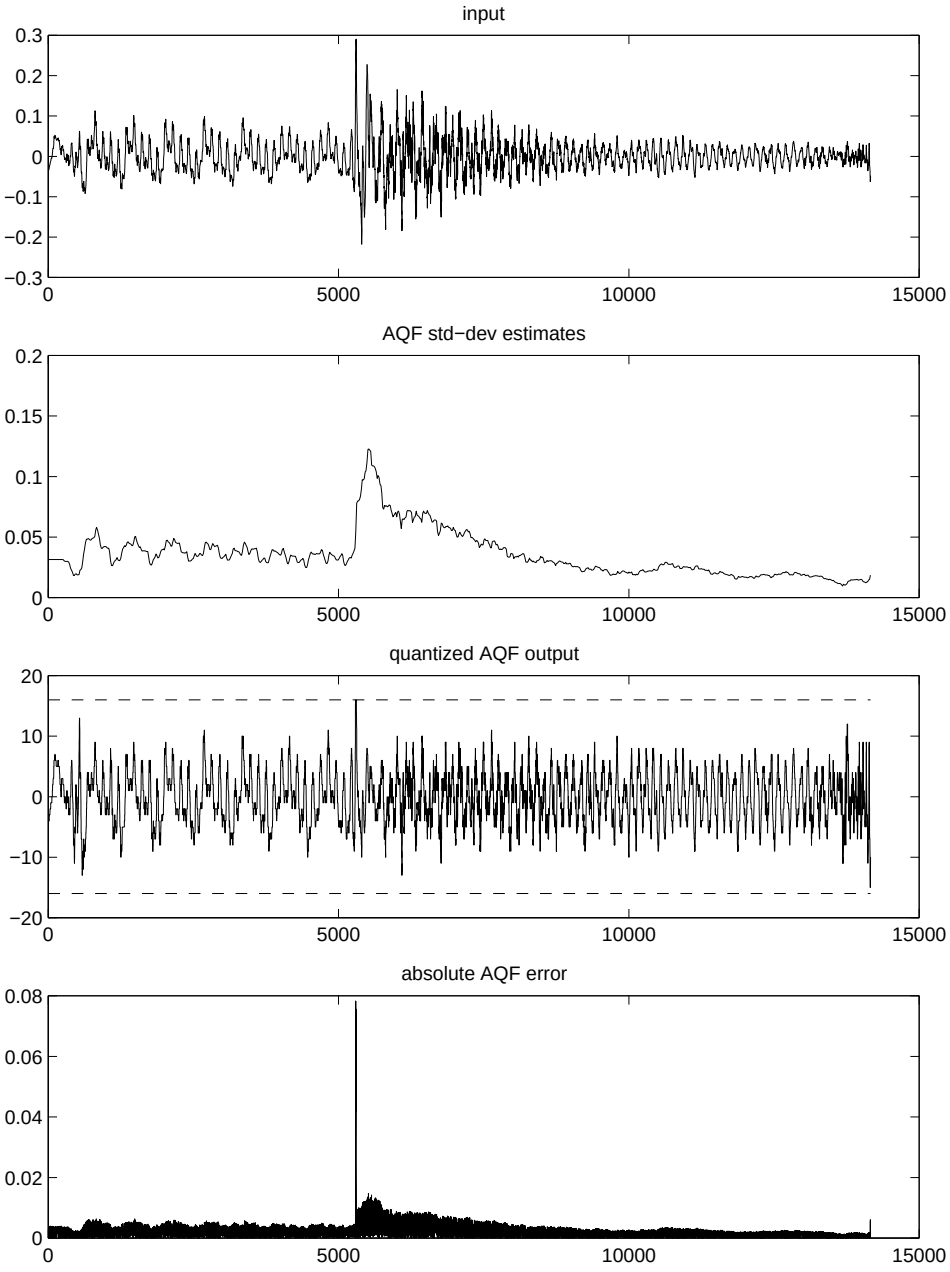

Figure 4: Block AQF, $N = 256$.

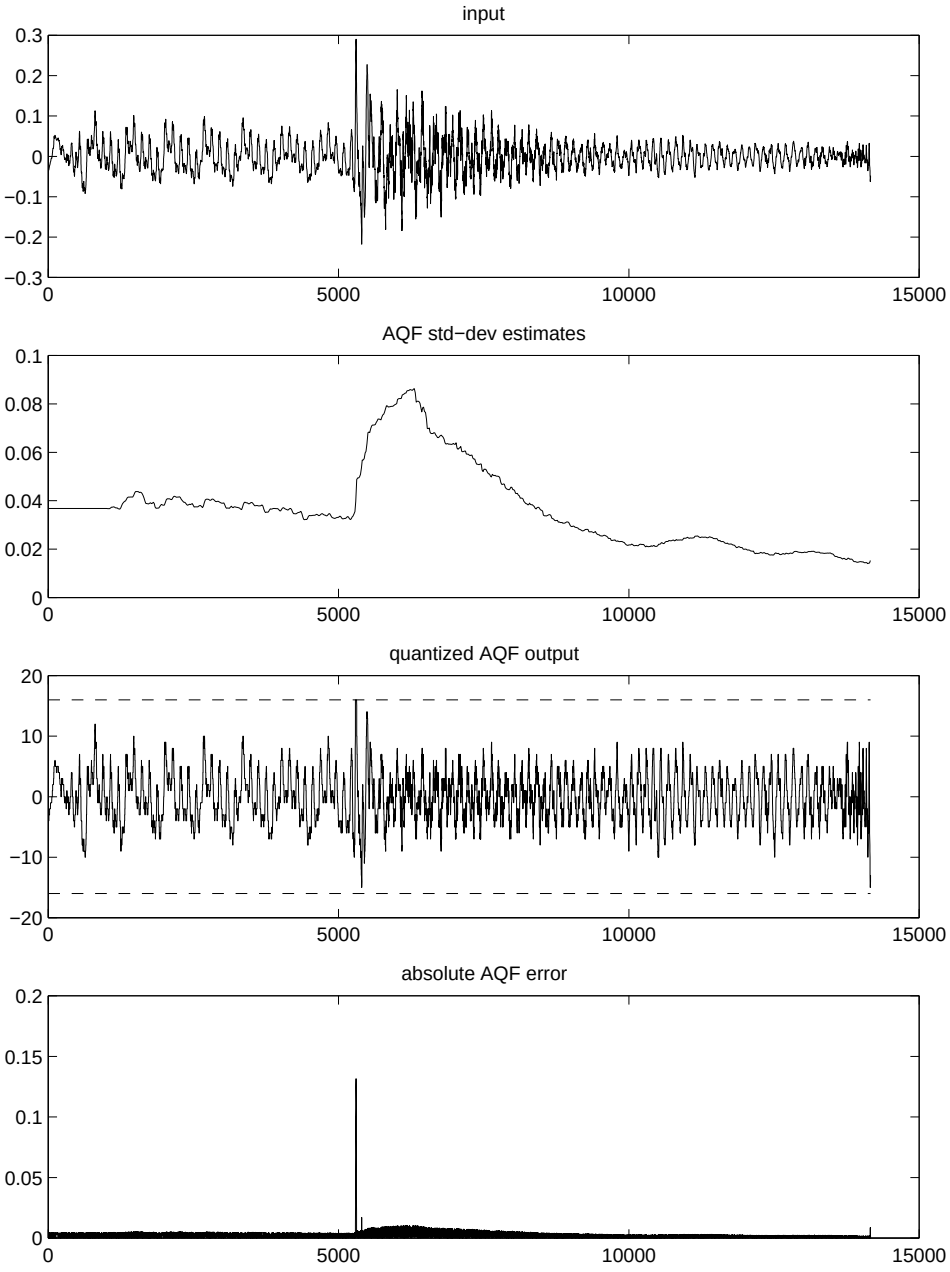

Figure 5: Block AQF, $N = 1024$.

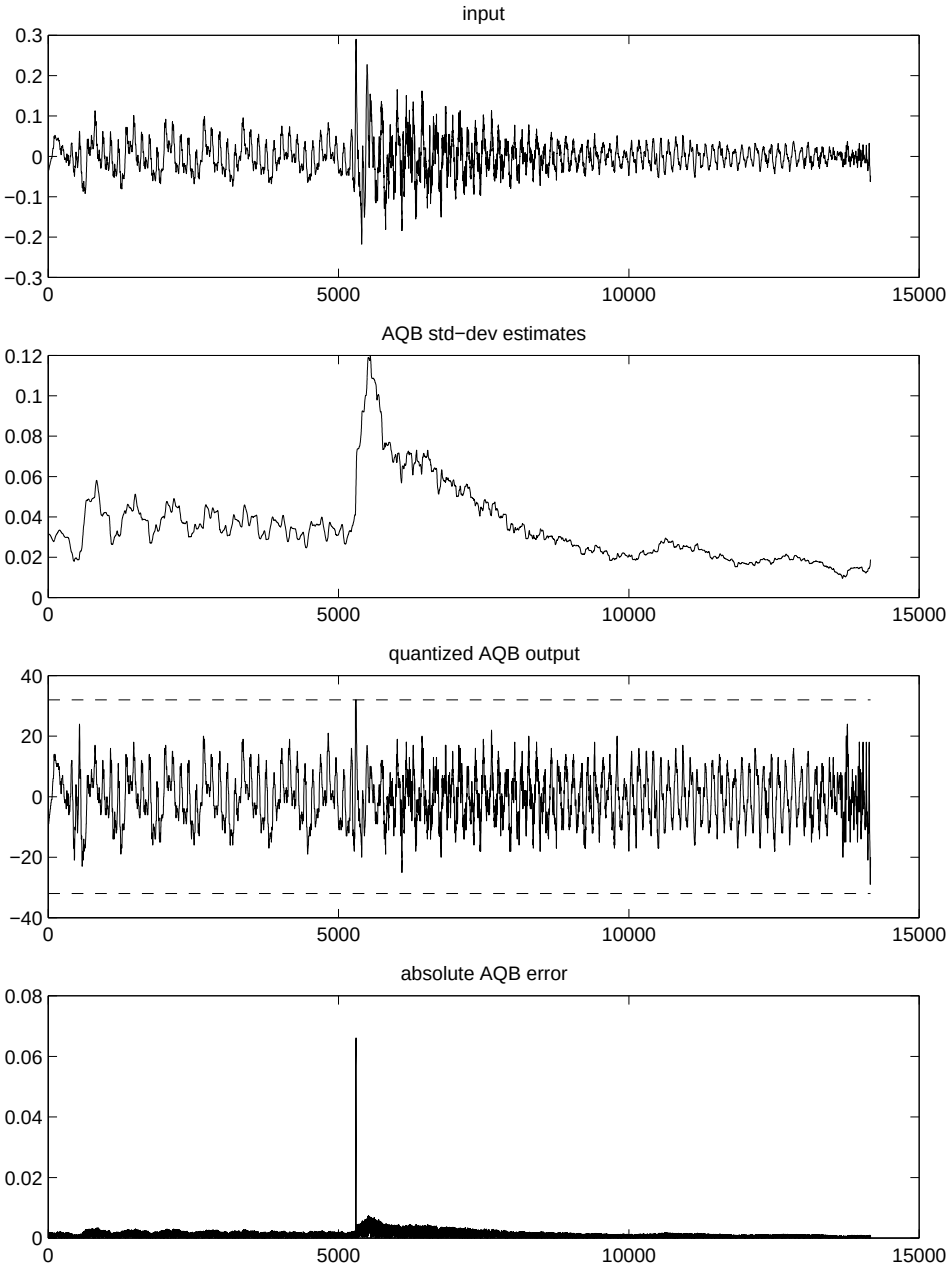

Figure 6: Block AQB, $N = 256$.

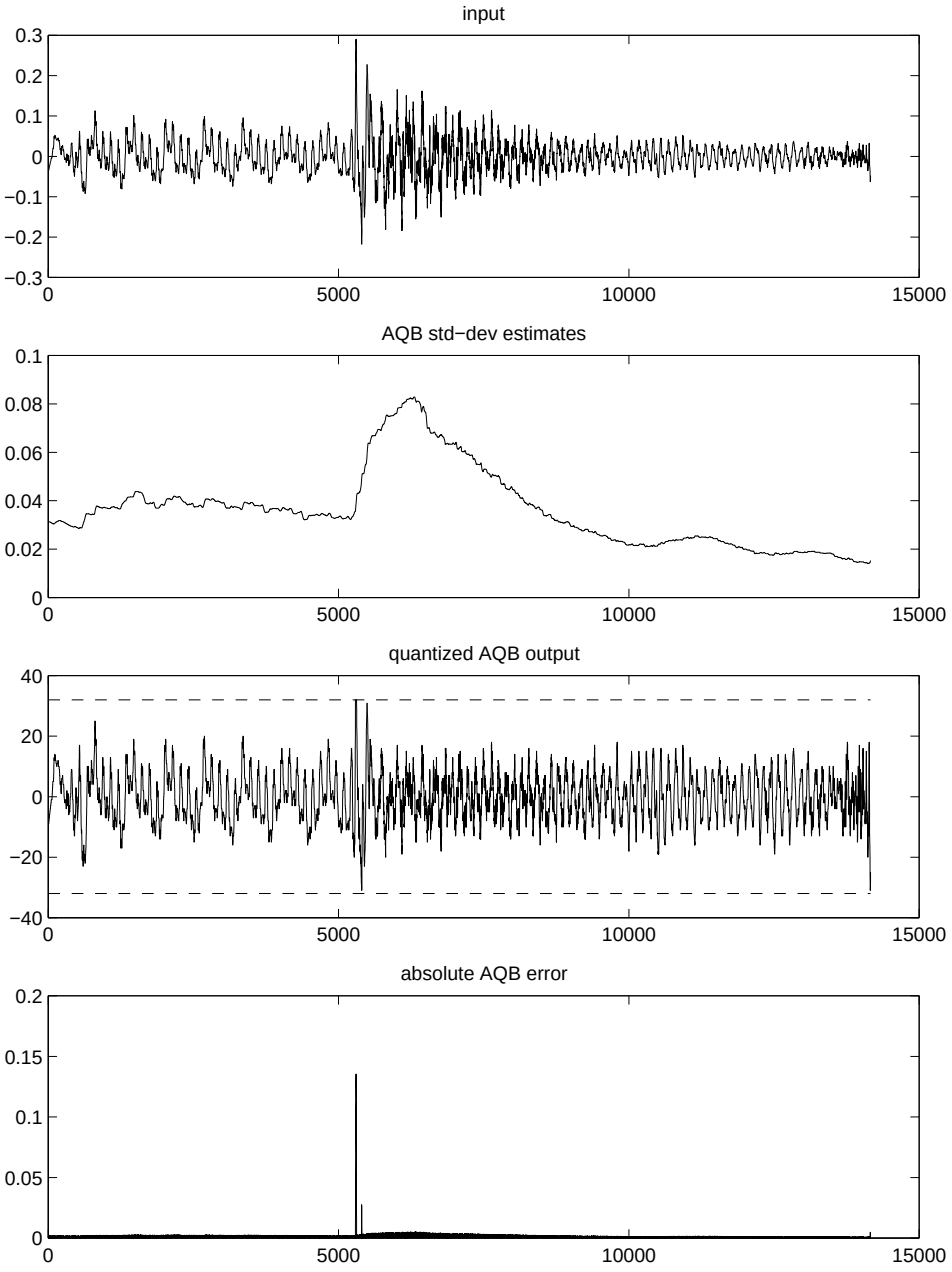

Figure 7: Block AQB, $N = 1024$.