POC HW#4 - Martyn Staalsen

Table of Contents

14.2

Exercise 14.2. Consider transmitting a signal with values chosen from the six level alphabet ±1,±3,±5.

```
% a. Suppose that all six symbols are equally likely. Identify N, x_i,
 and p(x_i), and
% calculate the information I(x_i) associated with each i.

% In this case, because there are 6 potential levels, N = 6.
% x_i represents the possible levels which could be sampled, which are
% ±1,±3,±5, and p(x_i) is 1/6 for all x_i since they are all equally
% likely.

% b. Suppose instead that the symbols ±1 occur with probability 1/4
 each, ±3
% occur with probability 1/8 each, and 5 occurs with probability 1/4.
 What
% percentage of the time is ?5 transmitted? What is the information
 conveyed
% by each of the symbols?

% the probabilities must add up to 1, so p(-5) = 1-
(1/4+1/4+1/8+1/8+1/4) =
% 1-1 = 0. This means that the symbol -5 will never be transmitted
 (and
% would hold an infinite amount of information.) The other symbols
% represent different amounts of information: ±1 and 5 represent
% log_2(4) = 2 bits of information, while ±3 represents log_2(8) = 3
 bits
% of information.
```

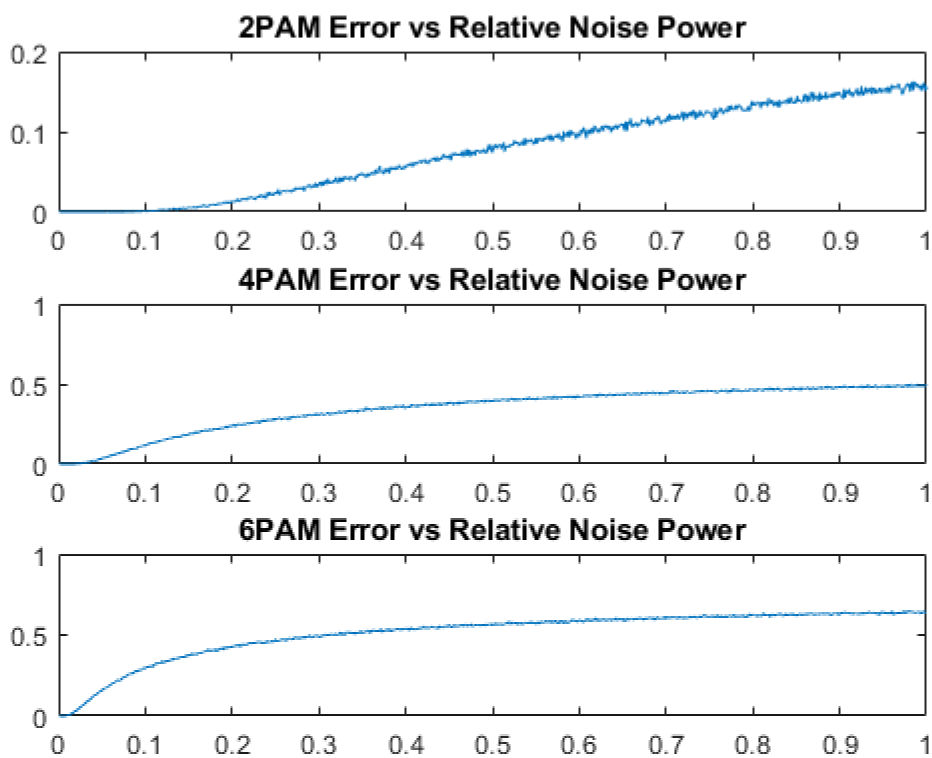
14.13

Use noisychan.mto compare the noise performance of two-level, four-level, and six-level transmissions.

```
% a. Modify the program to generate two- and six-level signals.

% b. Make a plot of the noise power versus the percentage of errors
 for
% two, four,and six levels.

pam2_err = [];
```

```
pam4_err = [];
pam6_err = [];
q=0;
numReps = 10;
noisePows = [];

for test_p = 0.001:0.001:1
q=q+1;
res2 = 0;
res4 = 0;
res6 = 0;
for reps = 1:numReps
% noisychan.m generate 2-level data and add noise
m=1000;                          % length of data sequence
p=test_p; s=1.0;                   % power of noise and signal
x=pam(m,2,s);                    % 4-PAM input with power 1...
L=sqrt(1);                     % ...with amp levels L
n=sqrt(p)*randn(1,m);            % noise with power p
y=x+n;                           % output adds noise to data
qy=quantalph(y,[-L,L]); % quantize
err=sum(abs(sign(qy'-x)))/m;      % percent transmission errors
res2 = res2+err;

% noisychan.m generate 4-level data and add noise
m=1000;                          % length of data sequence
p=test_p; s=1.0;                   % power of noise and signal
x=pam(m,4,s);                    % 4-PAM input with power 1...
L=sqrt(1/5);                     % ...with amp levels L
n=sqrt(p)*randn(1,m);            % noise with power p
y=x+n;                           % output adds noise to data
qy=quantalph(y,[-3*L,-L,L,3*L]); % quantize to [-3*L,-L,L,3*L]
err=sum(abs(sign(qy'-x)))/m    ;  % percent transmission errors
res4 = res4+err;



% noisychan.m generate 6-level data and add noise
m=1000;                          % length of data sequence
p=test_p; s=1.0;                   % power of noise and signal
x=pam(m,6,s);                    % 4-PAM input with power 1...
L=sqrt(3/35);                     % ...with amp levels L
n=sqrt(p)*randn(1,m);            % noise with power p
y=x+n;                           % output adds noise to data
qy=quantalph(y,[-5*L,-3*L,-L,L,3*L,5*L]); % quantize
err=sum(abs(sign(qy'-x)))/m  ;    % percent transmission errors
res6=res6+err;

end
res2 = res2/reps;
res4 = res4/reps;
res6=res6/reps;

pam2_err(q) = res2;
pam4_err(q)=res4;
```

```
pam6_err(q)=res6;
noisePows(q) = test_p;

end

figure()
subplot(3,1,1); plot(noisePows,pam2_err); title("2PAM Error vs
 Relative Noise Power")
subplot(3,1,2); plot(noisePows,pam4_err); title("4PAM Error vs
 Relative Noise Power")
subplot(3,1,3); plot(noisePows,pam6_err); title("6PAM Error vs
 Relative Noise Power")

% these plots show that a system with more levels is more strongly
 affected
% by noise. This makes intuitive sense especially for a PAM system,
 since
% the distance between these levels is closer together when more
 levels are
% used, such that the same noise variation will cause more errors when
 the
% levels are harder to distinguish between. However, this also
 supports
% Shannon's channel capacity equation, since an increased noise should
% reduce the capacity of the channel, as a smaller alphabet has to be
% used to combat a noisier channel.
```

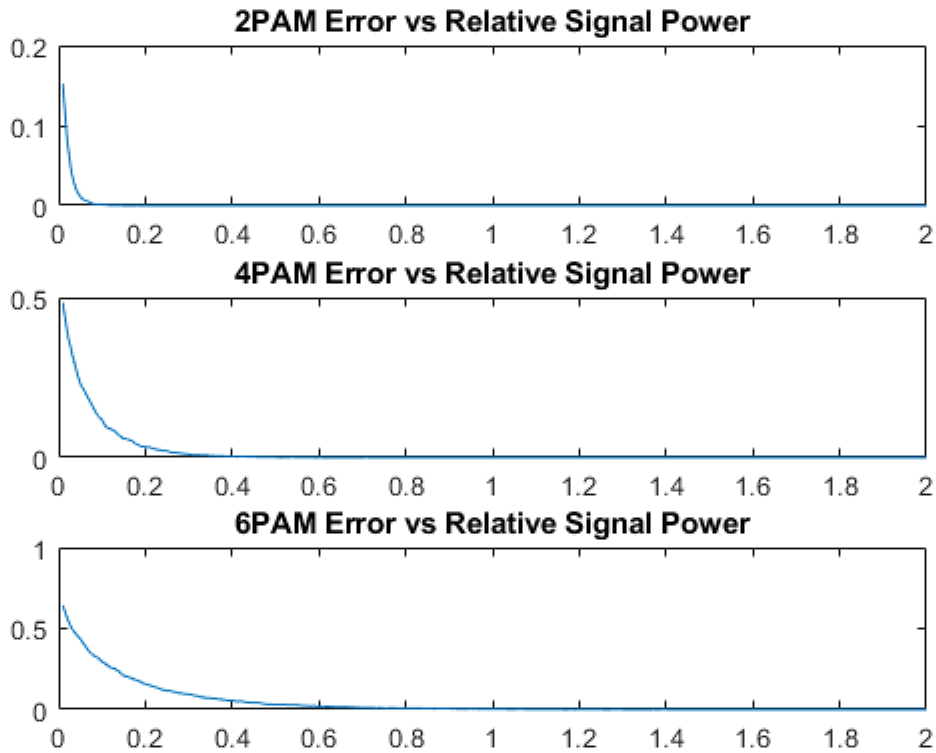


14.14

Usenoisychan.mto compare the power requirements for two-level, four-level, and six-level transmissions. Fix the noise power at p=0.01, and find the error probability for four-level transmission. Experimentally find the power S that is required to make the two-leveland six-level transmissions have the same probability of error. Can you think of a way to calculate this?

```
pam2_err = [];
pam4_err = [];
pam6_err = [];
q=0;
numReps = 10;
noisePows = [];

for test_p_sig = 0.01:0.01:2
q=q+1;
res2 = 0;
res4 = 0;
res6 = 0;
for reps = 1:numReps %take average of multiple iterations because
 randomness
% noisychan.m generate 2-level data and add noise
m=1000;                          % length of data sequence
p=0.01; s=test_p_sig;                   % power of noise and signal
x=pam(m,2,s);                    % 4-PAM input with power 1...
L=sqrt(s*1);                     % ...with amp levels L
n=sqrt(p)*randn(1,m);            % noise with power p
y=x+n;                           % output adds noise to data
qy=quantalph(y,[-L,L]); % quantize
err=sum(abs(sign(round(qy',5)-round(x,5))))/m;      % percent
 transmission errors

res2 = res2+err;

% noisychan.m generate 4-level data and add noise
m=1000;                          % length of data sequence
p=0.01; s=test_p_sig;                   % power of noise and signal
x=pam(m,4,s);                    % 4-PAM input with power 1...
L=sqrt(s*1/5);                     % ...with amp levels L
n=sqrt(p)*randn(1,m);            % noise with power p
y=x+n;                           % output adds noise to data
qy=quantalph(y,[-3*L,-L,L,3*L]); % quantize to [-3*L,-L,L,3*L]
err=sum(abs(sign(round(qy',5)-round(x,5))))/m    ;  % percent
 transmission errors
res4 = res4+err;



% noisychan.m generate 6-level data and add noise
m=1000;                          % length of data sequence
p=0.01; s=test_p_sig;                   % power of noise and signal
x=pam(m,6,s);                    % 4-PAM input with power 1...
L=sqrt(s*3/35);                     % ...with amp levels L
```

```
n=sqrt(p)*randn(1,m);            % noise with power p
y=x+n;                           % output adds noise to data
qy=quantalph(y,[-5*L,-3*L,-L,L,3*L,5*L]); % quantize
err=sum(abs(sign(round(qy',5)-round(x,5))))/m  ;    % percent
 transmission errors
res6=res6+err;

end
res2 = res2/reps;
res4 = res4/reps;
res6=res6/reps;

pam2_err(q) = res2;
pam4_err(q)=res4;
pam6_err(q)=res6;
noisePows(q) = test_p_sig;

end

figure()
subplot(3,1,1); plot(noisePows,pam2_err); title("2PAM Error vs
 Relative Signal Power")
subplot(3,1,2); plot(noisePows,pam4_err); title("4PAM Error vs
 Relative Signal Power")
subplot(3,1,3); plot(noisePows,pam6_err); title("6PAM Error vs
 Relative Signal Power")

% For a signal power of 1, the 4PAM system acheives 0 error. The 2PAM
% system can match this error rate with a power around 0.14, while the
 6PAM
% system does not settle to 0 error until a signal power level of
 about
% 1.83. To find calculate this mathematically, I would suggest trying
 to
% find a logarithmic curve which matches the drop of of the 2PAM,
 4PAM, and
% 6PAM systems and solving for the point where the 2PAM and 6PAM
 systems
% have the same error rate as the 4PAM system does with a signal power
 of
% 1.
```



14.16

Consider the source with N = 5 symbols with probabilities p(x1) = 1/16, p(x2) = 1/8, p(x3) = 1/4, p(x4) = 1/16, and p(x5) = 1/2.

```
% a. What is the entropy of this source?

% b. Build the Huffman chart.

% c. Show that the Huffman code is

% x1 ? 0001, x2 ? 001, x3 ? 01, x4 ? 0000, and x5 ? 1.

% d. What is the efficiency of this code?

% e. If this source were encoded naively, how many bits per symbol
 would be
% needed? What is the efficiency?

% Scan of Handwritten Solution Attached
```

Published with MATLAB® R2017b

POC HW#4

14.16

$N = 5, \quad p(x_1) = 1/16, \; p(x_2) = 1/8, \; p(x_3) = 1/4$
$p(x_4) = 1/16, \; p(x_5) = 1/2$

a) $H(X) = \sum_{i=1}^{N} p(x_i) I(x_i) = -\sum_{i=1}^{n} p(x_i) \log_2(p(x_i))$

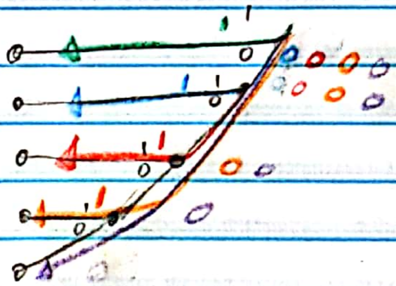
$$H(X) = -\tfrac{1}{16}\log_2\left(\tfrac{1}{2^4}\right) - \tfrac{1}{8}\log_2\left(\tfrac{1}{2^3}\right) - \tfrac{1}{4}\log_2\left(\tfrac{1}{2^2}\right) - \tfrac{1}{16}\log_2\left(\tfrac{1}{2^4}\right) - \tfrac{1}{2}\cdot\log_2\left(\tfrac{1}{2^1}\right)$$

$$= \frac{4}{16} + \frac{3}{8} + \frac{2}{4} + \frac{4}{16} + \frac{1}{2}$$

$$\boxed{H(x) = 1.875}$$

b)

x_i	$P(x_i)$
x_5	0.5
x_3	0.25
x_2	0.125
x_1	0.0625
x_4	0.0625



c)
x_1 : 0001
x_2 : 001
x_3 : 01
x_4 : 0000
x_5 : 1

$$\frac{4+3+2+4+1}{5} = \frac{14}{5} = 2.8$$

d) $\text{efficiency} = \dfrac{\text{entropy}}{\text{average \#bits per symbol}} = \dfrac{1.875}{2.8} = \boxed{0.670}$

e) Naive encoding: 000, 001, 010, 011, 100, 3 bits per symbol
efficiency $= \dfrac{1.875}{3} = \boxed{0.625}$, worse efficiency than the variable-rate code.

Scanned with CamScanner