Project 1 Baseband Channel

**Objective**: To simulate 4-ary communication and fading channels. In each case, observe the sent and received symbol values and measure bit errors. (Suggested package is MATLAB.)

## Part 1: 4-ary Channel

Generate an equally probable random sequence of 1’s and 0’s of length 200,000 or more. In other words, the a priori probabilities, i.e., the probability of sending a 1 or a 0 are: Pr[0s] = Pr[1s] = 0.5. Now obtain the QPSK symbols, i.e., form the 2-bit groups as follows: {A = ‘00’, B = ‘01’, C = ‘11’, D = ‘10’}.

% Generate an array of random bits using a binomial distribution generator

n = 1; p = 0.5; l = 200000;

msgBits = binornd(n, p, 1, l);

% Create QPSK Symbol list

slist = string(zeros(1, l/2));sbinlist = string(zeros(1, l/2));

binmap = containers.Map; smap = containers.Map;

binmap('00') = 'A'; binmap('01') = 'B'; binmap('11') = 'C'; binmap('10') = 'D';

smap('A') = '00'; smap('B') = '01'; smap('C') = '11'; smap('D') = '10';

for i = 1:2:200000

s = string(msgBits(i)) + string(msgBits(i+1));

sbinlist((i+1)/2) = s;

slist((i+1)/2) = binmap(s);

end

Consider a symmetric 4-ary communications channel with Pr[XR|XS] = 0.99 where XR and XS are the received and sent symbol values {A, B, C, D}. Input the QPSK symbols to the 4-ary channel and record the output symbols, de-map them to bits.

rlist = string(zeros(1, l/2));

rbinlist = string(zeros(1, l/2));

for k = 1:l/2

x = slist(k);

[rlist(k), rbinlist(k)] = fouraryChannel(x, smap, binmap);

end

%Calculate the bit error

badbits = 0;

for i = 1:l/2

r = rbinlist(i); s = sbinlist(i);

if r ~= s

badbits = badbits + 1;

end

end

biterror = double(badbits/(l/2));

disp('Bit Error: '+string(biterror\*100) + '%')

Bit Error: 1.991%

## Part 2: Fading Channel

%create the constellation mapping from binary sequence rbinlist

constellationMap = containers.Map;

constellationMap('A') = 1; constellationMap('B') = 1i; constellationMap('C') = -1; constellationMap('D') = -1i;

[cmplxlist, ltrlist] = binToSymbols(sbinlist, binmap, constellationMap);

sigma = .10;

a\_mag = raylrnd(sigma, [1, l/2]); %create the complex coefficient 'a' using a rayleigh distribution

theta = (2\*pi).\*rand([1, l/2]); %create theta based on uniform distribution

expTheta = exp(-1j.\*theta);

a\_vec = a\_mag .\* expTheta; % enumerate the a values in a list

y = fadeChannel(cmplxlist, a\_vec);

# Plotting Results

% tiledlayout(2, 1)

% nexttile

% scatter(real(cmplxlist), imag(cmplxlist), "blue", '\*')

% title('QPSK Signals Sent');

% ylabel('Quadrature')

% xlabel('In-Phase')

%

% nexttile

% scatter(real(y), imag(y), "red", '+')

% title('QPSK Signals Received');

% ylabel('Quadrature')

% xlabel('In-Phase')

% hold off;

# Calculating Bit Error

rec\_symbols = [real(y); imag(y)];

qpsks = [1, 0;0, 1; -1, 0;0, -1];

req\_qpsk = knnsearch(qpsks, rec\_symbols');

rcvMsgBits = backToBits(req\_qpsk);

%Calculate the bit error

badbits = 0;

for i = 1:l

r = rcvMsgBits(i); s = msgBits(i);

if r ~= s

badbits = badbits + 1;

end

end

biterror = double(badbits/l);

disp('Fade Channel Bit Error: '+string(biterror\*100) + '%')

Fade Channel Bit Error: 0%

function rcvMsgBits = backToBits(rec\_symbols)

l = length(rec\_symbols);

rcvMsgBits = double(zeros(1, l\*2));

for i = 1:2:l\*2

n = rec\_symbols((i+1)/2);

switch n

case 1

rcvMsgBits(i) = 0;

rcvMsgBits(i+1) = 0;

case 2

rcvMsgBits(i) = 0;

rcvMsgBits(i+1) = 1;

case 3

rcvMsgBits(i) = 1;

rcvMsgBits(i+1) = 1;

otherwise

rcvMsgBits(i) = 1;

rcvMsgBits(i+1) = 0;

end

end

end

function [y, ybin] = fouraryChannel(x, smap, binmap) % x is a symbol: [A, B, C, D]

binVal = smap(x);

i = randi([0,100]);

if i <= 98

y = x;

ybin = binVal;

return

end

j = mod(i, 2) + 1 ; % determine which bit gets flipped; assumes equiprob bit err

if binVal(j) == '1'

binVal(j) = '0';

else

binVal(j) = '1';

end

y = binmap(binVal);

ybin = binVal;

return

end

function [cmplxlist, ltrlist] = binToSymbols(xarr, binmap, cmap)

ylen = length(xarr);

cmplxlist = zeros(1, ylen);

ltrlist = string(zeros(1, ylen));

for n = 1:1:ylen

s = xarr(n);

ltrlist(n) = binmap(s);

cmplxlist(n) = cmap(binmap(s));

end

end

function y = fadeChannel(xarr, as)

xlen = length(xarr);

y = zeros([1, xlen]);

y(1) = xarr(1);

for n = 2:1:xlen

y(n) = xarr(n) + as(n).\*xarr(n-1);

end

end