Project 2

8PSK Modulation and Demodulation

CES 544 – Wireless Communication
Sonoma State University
Spring 2007
David Bozarth

Introduction

The signal r at the receiver of a communication system is modeled as function of time:

$$r(t) = \hat{s}(t) \cdot c(t) \cdot h(t) + n(t)$$

where $\hat{s}$ is the complex baseband modulation signal, c is the carrier, h is the channel response, and n is channel noise. When baseband modulation is performed on an information bit stream, the number of bits encoded per symbol is constant $\log_2 M$, with M representing the size of the symbol set. When the phase of the modulated signal is used to encode information into symbols, the method is known as M-ary phase shift keying (MPSK).

The complex baseband MPSK signal is

$$\hat{s}(t) = (E_S)^{½} \cdot \exp[-j \cdot (2\pi / M) \cdot m(t) + \theta_0]$$

where E_S is the energy per symbol, θ_0 is a constant phase offset, and $m(t) \varepsilon \{0, 1, 2, \dots , M-1\}$. We can see that for each of the M values of m, a unique symbol (complex signal value) results.

In a physical wireless communication system, these symbols are modulated onto a high-frequency carrier, and the resulting bandpass signal is transmitted into the atmosphere. The transmitted signal is subject to channel fading h(t) and noise n(t). If at the receiver, fading compensation can be performed on the detected baseband component, then all that remains is to recover the original information from the noise-affected symbol set.

In this simulation, we focus on 8PSK baseband modulation and demodulation in the presence of Rayleigh flat fading and additive white Gaussian noise (AWGN). Carrier modulation and demodulation are removed from the process without effect. Using Matlab, a bit stream is translated into 8PSK symbols. The symbol set is multiplied by a fading function and contaminated by noise. The fading is then removed by applying numerical division, and from the resulting symbol set is derived a best estimate of the original bit stream by minimum Euclidean distance approximation. Appendix I (Project 2 Procedure) gives details of the theory and procedure.

Core implementation

The project was implemented as an m-file:

```
% project2.m
% Simulate a wireless channel using 8PSK.
% Find BER for various Eb/No
% D.Bozarth, CES 544, Sonoma State University
% 03-31-07
%
graphBiasFlg = 1;   % 0 => Allow (#bit errors) == 0.
                    % 1 => Ensure min (#bit errors) == 1.
%
bsz = 486;      % length of binary sequence (one frame)
Es  = 1;        % energy per symbol
Rs  = 24e3;     % symbol rate, baud
fD  = 200;      % max Doppler frequency, Hz
ni  = 200;      % number of iterations used to determine BER
%
EbN = [0  5 10 15 20 25]; % bit energy per noise, dB
ben = [0  0  0  0  0  0]; % counted Bit Errors
ber = [0  0  0  0  0  0]; % calculated Bit Error Rates

format short e;

t1 = cputime;
for i = 1:size(ber, 2)
    be = 0 ; % number of bit errors detected

    for j = 1:ni
        % Generate a random binary sequence.
        [Bt] = zeros(1, bsz);
        for k = 1:bsz ; Bt(k) = (rand(1, 1) > 0.5); end

        % Transmit & receive the frame.
        [Br] = frameLink_8PSK( Bt, Es, Rs, fD, EbN(i) );

        % Compare with original sequence and find BER.
        dif = abs(Bt - Br);
        be = be + sum(dif);
    end

    if (graphBiasFlg == 1)
        if be == 0; be = 1; end
    end
    ben(i) = be;
    ber(i) = be ./ (bsz * ni);
end
t2 = cputime;
elapsed = t2 - t1

ben
ber
semilogy(EbN, ber);

% End of program.
```

The building block modules:

```
function [Br] = frameLink_8PSK(Bt, Es, Rs, fD, EbN)
% Simulate one frame 8PSK link.
% Br: 1 x n vector: Received bit stream after compensation & demodulation.
% Bt: 1 x n vector: Bit stream to be transmitted.
%   Precondition: Length of Bt corresponds to exactly one frame.
% Es: Energy per symbol (no dimension)
% Rs: Desired symbol rate, baud
% fD: Max Doppler frequency, Hz
% EbN: Bit energy per noise, dB
%
% CES 544, Sonoma State University
% D.Bozarth
% Built 03-31-07
% Mod 04-19-07: Corrected na (noise amplitude).
%
bsz = size(Bt, 2);  % length of binary sequence
ssz = bsz / 3;      % length of symbol sequence
Ts = 1 / Rs;         % symbol period, s

% Convert to complex baseband.
C = EightPSK_mod(Bt, Es);

% Transmit with flat Rayleigh fading.
N = ssz;
h = Rayleigh(N, fD, Ts);
Y = C .* h(2, :);

% Add noise and form the received signal.
na = sqrt( Es / (3 * 2 * 10 ^ (EbN / 10)) ); % noise amplitude
n = zeros(1, ssz);
for k = 1:ssz; a = randn(1, 1); b = randn(1, 1); n(k) = na * (a + i*b); end
r = Y + n;

% Receiver fading compensation.
rc = r ./ h(2, :);

% Demodulation -> binary sequence at receiver.
Br = EightPSK_demod(rc, Es);

% End of program.
```

```
function [X] = EightPSK_mod(B, e)
% Returns a vector of complex numbers representing the complex baseband
% symbols derived from a bit stream.
% Arguments:
%   B is a vector of binary digits.
%   e is the energy per symbol.
%
table = [3*pi/4 3*pi/2 pi/2 7*pi/4 pi 5*pi/4 pi/4 0];
table = sqrt(e) * exp(i .* table);
n_B = size(B);  % B is a (1 x n) vector.
n_B = n_B(2);   % Second element is the #columns in B.
n_sym = floor(n_B ./ 3);
X = zeros(1, n_sym);
ptr = 0;

for k = 0:(n_sym - 1)
    ptr = k * 3 + 1;
    a3 = B(ptr:(ptr + 2));

    ndx = a3(1)*4 + a3(2)*2 + a3(3)*1 + 1;
    X(k + 1) = table(ndx);
end

% end of program
```

```
function [B] = EightPSK_demod(X, e)
% Returns a bit stream derived from 8PSK demodulation.
% Arguments:
%   B is a vector of complex numbers representing complex baseband symbols.
%   e is the energy per symbol.
%
% D.Bozarth, CES 544, Sonoma State University
% 03-31-2007
%
s_tbl = [3*pi/4 3*pi/2 pi/2 7*pi/4 pi 5*pi/4 pi/4 0];
s_tbl = exp(i .* s_tbl);
b_tbl = [0 0 0 ; 0 0 1 ; 0 1 0 ; 0 1 1 ; 1 0 0 ; 1 0 1 ; 1 1 0 ; 1 1 1];

n_X = size(X);  % X is a (1 x n) vector.
n_X = n_X(2);   % Second element is the #columns in B.
n_B = n_X .* 3;
B = zeros(1, n_B);
ptr = 0;

for k = 0:(n_X - 1)
    ptr = k * 3 + 1;
    x = X(k + 1) ./ sqrt(e);

    % Euclidean distance weight function - Neural Net Toolbox
    D = dist(s_tbl', x);
    [m, ndx] = min(D);

    seq = b_tbl(ndx, :);

    B(ptr + 0) = seq(1);
    B(ptr + 1) = seq(2);
    B(ptr + 2) = seq(3);
end

% end of program
```

Calculations

Maximum Doppler shift. For f = 1800 MHz and v = 120 km/hr, f_D = 200 Hz.

Variance of the noise. Since the real and imaginary noise components each have variance 1, the variance of the noise envelope is:

$$\text{var}(n) = (1^2 + 1^2) \cdot (na) = 2 \cdot (na)\text{\^{}}2$$

where (na) is the noise amplitude.

Noise amplitude. For $E_S = 1$, M = 8, and $E_b/N_0 = 1$,

$$E_S = 3 \cdot \text{var}(n) = 3 \cdot 2 \cdot (na)\text{\^{}}2 = 1$$

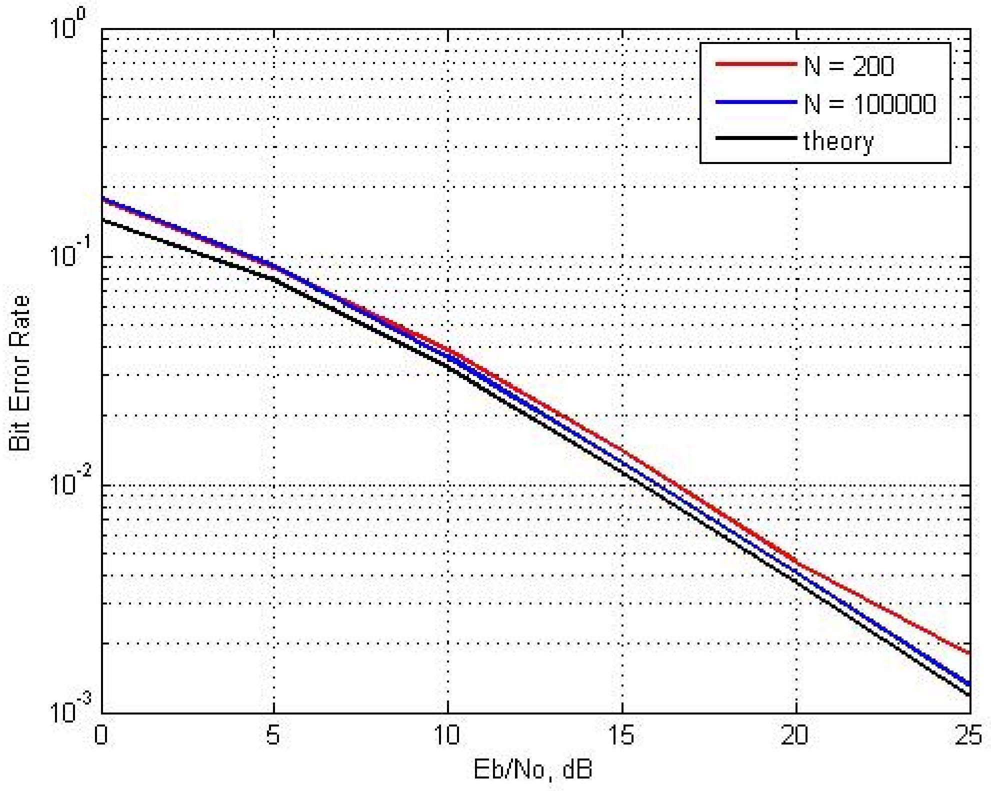
$$(na) = \text{sqrt}(1/6) = 0.40825$$

Noise amplitude is calculated by the function frameLink_8PSK(), using the above assumption for noise variance.

$$(na) = \text{sqrt}(E_S / [6 \cdot 10\text{\^{}}((E_b/N_0) / 10)])$$

Results

The simulation was run using 200 and 100000 iterations. An expression for symbol error rate in terms of M and SNR[1] was used to plot theoretical Bit Error Rate vs. E_b/N_0.



Regarding the theoretical plot, we know that SNR = $(E_b/N_0) \cdot (R_S/B) \cdot \log_2 M$, where R_S is symbol rate and B is bandwidth. Assuming ($B = R_S$) yields an expression for SNR in terms of (E_b/N_0). This result was used to produce the above graph.

Under a different assumption[2] ($B = 2 \cdot R_S$), the theoretical plot varies widely from the simulation results, and is not shown here.

The Matlab code used to form the theoretical function is shown below.

[1] J.Wu, C.Xiao, N.Beaulieu. "Optimal Diversity Combining based on Noisy Channel Estimation," in *IEEE Communications Society 0-7803-8533-0*, 2004, p.216
[2] Haykin, Simon, *Communication Systems (4th Edition)*, John Wiley & Sons, 2001, p. 368

```
function [X] = ber_MPSK(M, EbN)
% Return the theoretical Bit Error Rate corresponding to a given Eb/No.
% Assume M-ary phase shift keying.
% M: Number of bits per symbol
% EbN: vector: Bit energy per noise, dB
%
% CES 544, Sonoma State University
% D.Bozarth
% Built 04-19-07
%
g = 10.^(EbN/10) * log2(M) / 1; % Signal-to-Noise ratio

gs = g * sin(pi/M)^2;
gss = sqrt(gs ./ (1 + gs));

ser = (M-1)/M - gss .* (0.5 + (1/pi) .* atan( gss .* cot(pi/M) )); % Symbol error rate
X = ser ./ log2(M);
% End of program
```