Πολυτεχνειο Κρητης, Σχολη Η.Μ.Μ.Υ. Παράδοση: 20/3/2018
Project 1 - Viterbi Algorithm Προθεσμία: 17/4/2018
Κωνσταντινος Σκυβαλακης - 2013030034
ΤΗΛ513 - Δορυφορικες Επικοινωνιες

# Ερώτημα Α

Στο ερώτημα Α υπολογίζουμε το BER της διαμόρφωσης MSK. Αρχικά, προσομοιώσαμε την εξίσωση (11), $y_n = ATz_n + \sqrt{\beta T}n_n$, για SNR = 5 dB, $N = 10^5$ MSK σύμβολα και κατά συνέπεια για $0.5 \times 10^5$ OQPSK σύμβολα καθώς αν ομαδοποιήσουμε τα ζευγάρια $(x_{2n-1}, x_{2n})$ διαδοχικών MSK συμβόλων σε QPSK σύμβολα $x_{I,n} + jx_{Q,n}$ λαμβάνουμε τον μιγαδικό φάκελο της OQPSK. Τέλος, εκτιμήσαμε το BER το οποίο και ήταν ίσο με 7.22%. Παρακάτω παρατίθεται ο κώδικας που υλοποιεί το πρώτο αυτό ερώτημα.

Ερώτημα Α - Κώδικας MATLAB

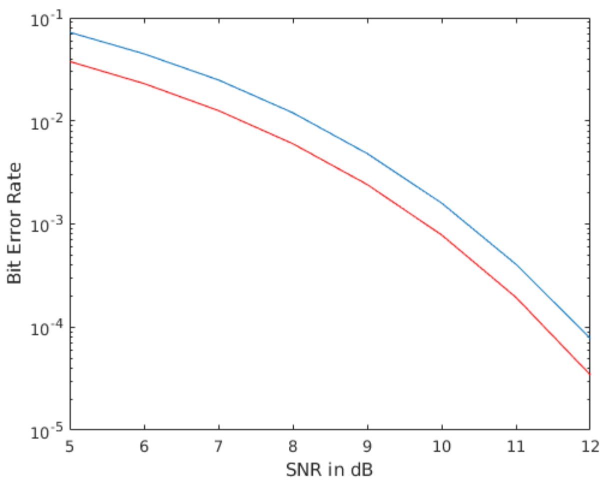
```matlab
P = 100;
N = 10^5;
A = 1;
T = 0.1;
beta = 0.02;
SNR_dB = 5;
SNR = 10.^(SNR_dB/10);
BER = 0;

for p=1:P
    x_est = zeros(1,N);
    x = 2*round(rand(1,N))-1;

    x_i = zeros(1,N/2);
    x_q = zeros(1,N/2);
    x_i(1) = -(-1)*1;         %x_i(0)=-x_q(n-1)*x(2n-1)=-x_q(-1)*x(-1)=-(-1)*1
    x_q(1) = -x_i(1)*x(1);    %x_q(0)=-x_i(0)*x(0)=-1*x(0)

    for n=2:N/2
        x_i(n) = -x_q(n-1)*x(2*(n-1));
        x_q(n) = -x_i(n)*x(2*n-1);
    end

    z = x_i + 1j*x_q;

    n = randn(1,N/2) + 1j*randn(1,N/2);
    y = A*T*z + sqrt(T^2*A^2/SNR)*n;

    y_est_r = sign(real(y));
    y_est_i = sign(imag(y));

    x_est(1) = -y_est_i(1)*y_est_r(1);
    for k=2:N/2
        x_est(2*(k-1)) = -y_est_i(k-1)*y_est_r(k);
        x_est(2*k-1) = -y_est_i(k)*y_est_r(k);
    end
    BER = BER + sum(x_est ~= x);

end

BER = BER/(N*P);
```

# Ερώτημα Β

Στο ερώτημα Β χρησιμοποιούμε το κομμάτι από τον κώδικα του ερωτήματος Α που δημιουργεί τα OQPSK σύμβολα και απλώς επεκτείνουμε τον υπολογισμό του BER και για περισσότερες τιμές του SNR = 6:12 dB. Παρακάτω παρατίθεται ο σχετικός κώδικας και το αντίστοιχο γράφημα που προκύπτει.

Ερώτημα Β - Κώδικας MATLAB

```matlab
P = 100;
N = 10^5;
A = 1;
T = 0.1;
SNR_dB = 5:12;
SNR = 10.^(SNR_dB/10);
BER = zeros(1,length(SNR));

for j=1:length(SNR)
    for p=1:P
        x_est = zeros(1,N);
        x = 2*round(rand(1,N))-1;

        x_i = zeros(1,N/2);
        x_q = zeros(1,N/2);
        x_i(1) = -(-1)*1;          %x_i(0)=-x_q(n-1)*x(2n-1)=-x_q(-1)*x(-1)=-(-1)*1
        x_q(1) = -x_i(1)*x(1);     %x_q(0)=-x_i(0)*x(0)=-1*x(0)

        for n=2:N/2
            x_i(n) = -x_q(n-1)*x(2*(n-1));
            x_q(n) = -x_i(n)*x(2*n-1);
        end

        z = x_i + 1j*x_q;

        n = randn(1,N/2) + 1j*randn(1,N/2);
        y = A*T*z + sqrt(T^2*A^2/SNR(j))*n;

        y_est_r = sign(real(y));
        y_est_i = sign(imag(y));

        x_est(1) = -y_est_i(1)*y_est_r(1);
        for k=2:N/2
            x_est(2*(k-1)) = -y_est_i(k-1)*y_est_r(k);
            x_est(2*k-1) = -y_est_i(k)*y_est_r(k);
        end
        BER(j) = BER(j) + sum(x_est ~= x);
    end
    BER(j) = BER(j)/(N*P);
end

figure, semilogy(SNR_dB,BER), xlabel('SNR in dB'), ylabel('Bit Error Rate'), hold on, semilogy(
    SNR_dB,qfunc(sqrt(SNR)),'r')
```




Σχήμα 1: Διάγραμμα BER ερωτήματος Β.

## Ερώτημα Γ

Ο λόγος που παρατηρούμε αυτή τη διαφορά μεταξύ του BER της προσομοίωσης μας και της συνάρτησης $Q(\sqrt{SNR})$ είναι επειδή το σύστημα μας έχει μνήμη καθώς το QPSK σύμβολο $z_n$ εξαρτάται από το προηγούμενο QPSK σύμβολο $z_{n-1}$ και από το ζεύγος MSK συμβόλων $(x_{2n-1}, x_{2n})$. Ως αποτέλεσμα των προαναφερθέντων η πιθανότητα σφάλματος της προσομοίωσης μας είναι κάτω φραγμένη από την συνάρτηση $Q(\sqrt{SNR})$.

## Ερώτημα Δ

Στο ερώτημα Δ προσομοιώσαμε την εξίσωση (17) της εκφώνησης, για SNR = 5 dB, $N = 10^5$ MSK σύμβολα και εκτιμήσαμε το BER το οποίο και ήταν ίσο με 7.26%. Παρακάτω παρατίθεται ο κώδικας του ερωτήματος Δ.

Ερώτημα Δ - Κώδικας MATLAB

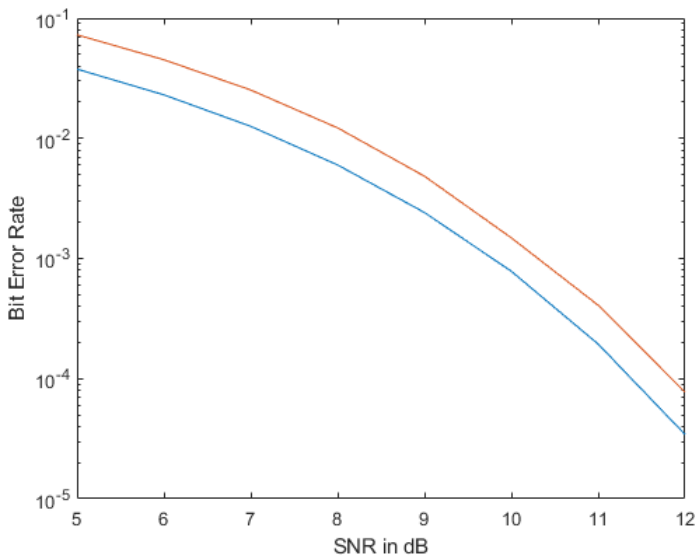
```
SNR_dB = 5;
SNR = 10.^(SNR_dB/10);
N = 10^5;
P = 1000;
A = 1;
T = 0.1;
s1 = [A*sqrt(T); 0];
s2 = [-2*A*sqrt(T)*1j/pi; A*sqrt(T)*sqrt(pi^2 -4)/pi];
x = 2*round(rand(1,N))-1;
phi = zeros(1,N);
r = zeros(2,N);
BER_VA = 0;

for j=1:P
    phi = zeros(1,N);
    n1 = sqrt(A^2*T/SNR)*(randn(1,N) + 1j*randn(1,N));
    n2 = sqrt(A^2*T/SNR)*(randn(1,N) + 1j*randn(1,N));

    for n=1:N
        phi(n+1) = phi(n) + x(n)*pi/2;

        if(x(n)==1)
            r(:,n) =  s1.*exp(1j*phi(n)) + [n1(n); n2(n)];
        else
            r(:,n) =  s2.*exp(1j*phi(n)) + [n1(n); n2(n)];
        end
    end

    x_est = ViterbiAlgorithm(N,s1,s2,r);
    BER_VA = BER_VA + sum(x~=x_est);
end

BER_VA = BER_VA/(N*P);
```

# Ερώτημα Ε

Στο ερώτημα Ε χρειάστηκε να υλοποιήσουμε τον αλγόριθμο του Viterbi ο κώδικας του οποίου παρατίθεται στο τέλος αυτής της εργασίας. Η υλοποίηση βασίστηκε στο διάγραμμα Trellis για τη φάση των MSK συμβόλων και στις επιτρεπτές μεταβάσεις σε κάθε στάδιο μετάδοσης.

Ερώτημα Ε - Κώδικας MATLAB

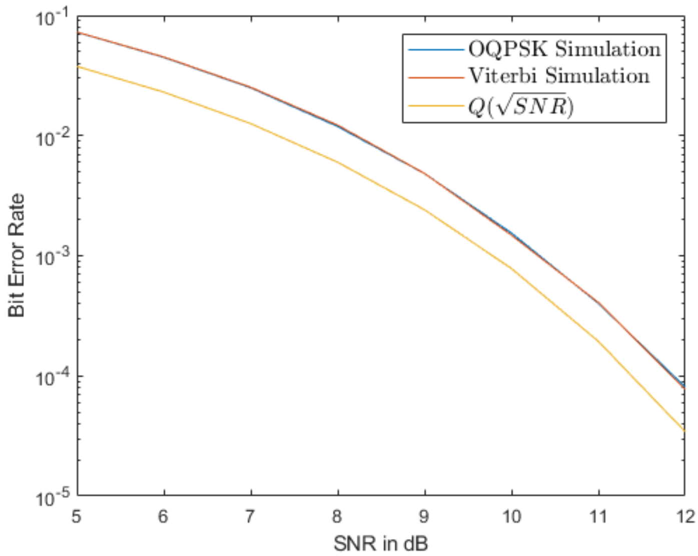
```
P = 1000;
N = 10^5;
A = 1;
T = 0.1;
SNR_dB = 5:12;
SNR = 10.^(SNR_dB/10);
BER_VA = zeros(1,length(SNR));
r = zeros(2,N);
s1 = [A*sqrt(T); 0];
s2 = [-2*A*sqrt(T)*1j/pi; A*sqrt(T)*sqrt(pi^2 -4)/pi];

for i=1:length(SNR)
    for j=1:P
        phi = zeros(1,N);
        n1 = sqrt(A^2*T/SNR(i))*(randn(1,N) + 1j*randn(1,N));
        n2 = sqrt(A^2*T/SNR(i))*(randn(1,N) + 1j*randn(1,N));

        for n=1:N
            phi(n+1) = phi(n) + x(n)*pi/2;
            if(x(n)==1)
                r(:,n) =  s1.*exp(1j*phi(n)) + [n1(n); n2(n)];
            else
                r(:,n) =  s2.*exp(1j*phi(n)) + [n1(n); n2(n)];
            end
        end

        x_est = ViterbiAlgorithm(N,s1,s2,r);
        BER_VA(i) = BER_VA(i) + sum(x~=x_est);
    end
    BER_VA(i) = BER_VA(i)/(N*P);
end

figure, semilogy(SNR_dB,BER_VA), xlabel('SNR in dB'), ylabel('Bit Error Rate')
```




Σχήμα 2: Διάγραμμα BER ερωτήματος Ε.

## Ερώτημα Ζ

Για το ερώτημα Ζ απλώς παραθέτουμε ένα κοινό plot των παραπάνω BER διαγραμμάτων σε συνδυασμό με το διάγραμμα της συνάρτησης $Q(\sqrt{SNR})$.




Σχήμα 3: Κοινό διάγραμμα BER ερωτήματος Ζ.

## Ερώτημα Η

Η εργασία αυτή θα μπορούσε να ονομαστεί SURVIVOR για τον απλούστατο λόγο ότι ο αλγόριθμος του Viterbi που χρησιμοποιούμε στο τελευταίο στάδιο του επιλέγει ένα από τα δύο μονοπάτια το οποίο προσέρει τη μέγιστη πιθανοφάνεια και είναι λοιπόν το survivor path.

**ΣΗΜΕΙΩΣΗ:** Το λάθος που είχα κάνει στα διαγράμματα οφείλεται σε έναν πολλαπλασιασμό με τον αριθμό 2 κατά την εφαρμογή του θορύβου στο χρήσιμο σήμα με αποτέλεσμα η μέθος του Viterbi να δινει χειρότερο BER από την πρώτη μέθοδο ενώ κανονικά θα έπρεπε να δίνει το ίδιο μιας και οι δύο μέθοδοι είναι ML για την MSK.

```
function [x_est] = ViterbiAlgorithm(N,s1,s2,r)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INPUT :
% N : Number of Symbols
% s1 : Constant vector for x_n = 1
% s2 : Constant vector for x_n = -1
% r : Baseband equivalent signal
%
%----------------------------------------------------------------
% OUTPUT :
% x_est : estimation of vector x
%
%----------------------------------------------------------------
% Variable Naming Convention :
%
% 1 --> 3pi/2
% 2 --> pi
% 3 --> pi/2
% 4 --> 0
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initializations
w12 = zeros(1,N); w32 = w12; w14 = w12; w34 = w12; w21 = w12; w41 = w12; w43 = w12;
w1 = w12; w2 = w12; w3 = w12; w4 = w12; i1 = w12; i2 = w12; i3 = w12; i4 = w12; x_est = w12;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Forward-Pass

i2(1) = -1;     % Set to -1 since on the first step there is no transition from 0 -> 0 or from 0 -> pi
i4(1) = -1;
w1(1) = real(r(:,1)'*s2);   % w2 from the trellis diagram
w3(1) = real(r(:,1)'*s1);   % w1 from the trellis diagram

for n=2:N
    if(mod(n,2)==0)
        w12(n) = real(r(:,n)'*s2*exp(1j*3*pi/2));
        w32(n) = real(r(:,n)'*s1*exp(1j*pi/2));
        w14(n) = real(r(:,n)'*s1*exp(1j*3*pi/2));
        w34(n) = real(r(:,n)'*s2*exp(1j*pi/2));

       x_est = zeros(1,N); t1 = w12(n)+w1(n-1);
        t2 = w32(n)+w3(n-1);
        [w2(n),i2(n)] = max([t1 0 t2 0]);

        t1 = w14(n)+w1(n-1);
        t2 = w34(n)+w3(n-1);
        [w4(n),i4(n)] = max([t1 0 t2 0]);
    else
        w21(n) = real(r(:,n)'*s1*exp(1j*pi));
        w41(n) = real(r(:,n)'*s2);
        w23(n) = real(r(:,n)'*s2*exp(1j*pi));
        w43(n) = real(r(:,n)'*s1);

        t1 = w21(n)+w2(n-1);
        t2 = w41(n)+w4(n-1);
        [w1(n),i1(n)] = max([0 t1 0 t2]);

        t1 = w23(n)+w2(n-1);
        t2 = w43(n)+w4(n-1);
        [w3(n),i3(n)] = max([0 t1 0 t2]);
    end
end
```

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Backward-Pass

% Keeping only one of the paths that gives the highest weight sum
if(mod(n,2)~=0)
    [~,path(N+1)] = max([w1(n) 0 w3(n) 0]);
else
    [~,path(N+1)] = max([0 w2(n) 0 w4(n)]);
end

path(1) = 4;
for n=N:-1:1
    if(mod(n,2)~=0 && n~=1)
        [~,i] = max([w1(n) 0 w3(n) 0]);
        in = [i1(n) 0 i3(n) 0];
        path(n) = in(i);
    elseif(mod(n,2)==0 && n~=1)
        [~,i] = max([0 w2(n) 0 w4(n)]);
        in = [0 i2(n) 0 i4(n)];
        path(n) = in(i);
    end

    if(path(n)-path(n+1)==-1)
        x_est(n) = -1;
    elseif(path(n)-path(n+1)==1)
        x_est(n) = 1;
    elseif(path(n)-path(n+1)==-3)
        x_est(n) = 1;
    elseif(path(n)-path(n+1)==3)
        x_est(n) = -1;
    end
end

end
```