ECE710 Space Time Coding For Wireless Communication HW3

Zhirong Li
Electrical & Computer Engineering Department
University of Waterloo, Waterloo, ON, Canada
z32li@engmail.uwaterloo.ca

Abstract

This homework report intends to provide an explanation of main steps in simulation of a baseband 4-state 4-PSK space-time trellis code commnucation system.

Index Terms

Wireless, Rayleigh Fading Channel, transmitter diversity, Monte Carlo simulation, BER, FER.

I. Simulation Requirements

SIMULATION of 4-state 4-PSK space-time trellis code- In this problem, you will create a computer program to simulate the error rate performance of Tarokhs 4-state 4-PSK spacetime trellis code (See Fig.1)

a) Assume that the channnel coefficients are spatially independent, i.e. spatial correlation between transmit antennas is zero,$\rho = 0$. Plot bit error rate (BER) and frame error rate (FER) vs. SNR for symbol-by-symbol interleaved channel (i.e. assume infinite ideal interleaving).
b) Assume now there exists spatial dependency, i.e. $\rho \neq 0$. Plot BER and FER plots for the values of $\rho = 0.4$, $\rho = 0.8$, $\rho = 0.99$, $\rho = 1$.
c) Repeat a) and b) for the quasi-static channel. Assume frame length is 130 symbols.
d) Based on your observations above, discuss the effect of spatial correlation on the diversity order over interleaved and quasi-static channels.

Hint: Since the question assumes frequency and time-nonselective channels, you do n-o-t need to use your simulator developed in Homework 1 for time -selective channels. For this homework, simply generate complex Gaussian random variables with appropriate mean and variance for your fading coefficients, whose amplitude needs to follow Rayleigh distribution.

II. Methodology

A. Space Time Trellis Coding

In the original paper, Tarokh et.al. proposed the design rules to construct full-rank codeword different matrices for 2-TX antenna case.

a) Transitions departing from the same state differ in the sencond symbol.
b) Transtions arriving at the same state differ in the first symbol.

Making B and therefore A have full rank 2. The handcrafted Tarokh 4-state 4-PSK space-time trellis code can be generated by following equivalent representations. (Course Handout Part V pp. 20)

```
function [Trans1, Trans2]=STTC_ENC(raw_bits)
global Es;
jj=sqrt(-1);
if ((size(raw_bits,2)/2)~=0)
    raw_bits=[raw_bits 0];
end
raw_bits=[raw_bits 0 0];% Force Convolution encoder to reach zero state by adding [0 0 ]
Trans1=0;
Trans2=2*raw_bits(1)+raw_bits(2);
for SymsLoop=2:bitshift(size(raw_bits,2),-1)
    Trans1(SymsLoop)=2*raw_bits(2*SymsLoop-3)+raw_bits(2*SymsLoop-2);
    Trans2(SymsLoop)=2*raw_bits(2*SymsLoop-1)+raw_bits(2*SymsLoop);
```

```
end
Trans1=sqrt(Es)*exp(jj*2*pi/4*Trans1);
Trans2=sqrt(Es)*exp(jj*2*pi/4*Trans2);
```

The incoming bits are firstly padded to length of multiple of 2. Using delay elements representation, we avoid calculating the corresponding output by using switch-case sentences which is a kind of state machine to determine the output. The modulated signals are passed to time and frequency non-selective rayleigh fading channels. In the receiver end, we use soft decision viterbi decoding algorithms to recover the original raw bits.

B. Viterbi Algorithms

The Viterbi algorithm calculates the maximum likelihood of a path being correct by its hamming distance from the real message. It can be realized as following steps:

a) First, select the state having the smallest accumulated error metric and save the state number of that state.
b) Iteratively perform the following step until the beginning of the trellis is reached: Working backward through the state history table, for the selected state, select a new state which is listed in the state history table as being the predecessor to that state. Save the state number of each selected state. This step is called traceback.
c) Now work forward through the list of selected states saved in the previous steps. Look up what input bit corresponds to a transition from each predecessor state to its successor state. That is the bit that must have been encoded by the convolutional encoder.

In our homework's case, the constraint length $K = 1$, because one incoming influences only one output symbol pair. When searching a trellis backward to check for hamming metrics, the depth should generally be no more than $5 \rightarrow 7 \times K$, here we select the traceback length to be $7 \times K + 1$. Here including 1symbols for flushing bits. The decoder builds up a trellis of depth of 8 and then traces back to the beginning of the trellis and output two bits. The decoder then shifts the trellis left one time instant, keep the accumulated error metrics, then computes the error metrics for the next time instant, trace back again and output 2bits. This continues in this way until it reaches the flushing bits. In our simulation, we force the trellis go back to zero state by adding two bits 00 in the end of the incoming bit streams.

C. The main control algorithm

In order to determine the FER and BER vs SNR in dB, as introduced in review part 2, the number of trials to garantee 95% sure that my estimate is within 10% of the true values. ($N = \frac{1}{P_e}\left(\frac{Q^{-1}(\alpha/2)}{\beta}\right)^2$ Let $\alpha = 0.05$ and $\beta = 0.1$). If we are supposed to observe the FER around 10^{-4} we need to run at least $10^4 \times 38.4$ frames whose length is supposed to be 130 in our simulations. The source code is listed as follows:

```
for SNR=SNR_start:SNR_end
    NwPwr=SigPwr/(10^(SNR/10));
    if SNR<10
        BER_Theory(SNR-SNR_start+1)=1e-2;
    elseif SNR<16
        BER_Theory(SNR-SNR_start+1)=1e-3;
    else
        BER_Theory(SNR-SNR_start+1)=1e-4;
    end
    Rayleigh_Bitlen=ceil(sqrt((40/BER_Theory(SNR-SNR_start+1) )));

    Rayleigh_Trialnum=Rayleigh_Bitlen;
    Es=SigPwr/Transnum;
    BERacc_Rayleigh=0;
    for Trialoop=1:Trailnum
        BECacc_Rayleigh=0;
        for Nbitloop=1:Rayleigh_Trialnum
            raw_bits=randint(1,Rayleigh_Bitlen);
            [Trans1_Syms,Trans2_Syms]=STTC_ENC(raw_bits);

            Rayleigh_noise_Syms=sqrt(NwPwr/2)*(randn(1,ceil(Rayleigh_Bitlen/2)+1)+jj
*randn(1,ceil(Rayleigh_Bitlen/2)+1));
            Rayleigh_alpha1_Syms=sqrt(1/2)*(randn(1,ceil(Rayleigh_Bitlen/2)+1)+jj*
randn(1,ceil(Rayleigh_Bitlen/2)+1));
            Rayleigh_alpha2_Syms=sqrt(1/2)*(randn(1,ceil(Rayleigh_Bitlen/2)+1)+jj*
```

```
randn(1,ceil(Rayleigh_Bitlen/2)+1));


            tmp1=1/sqrt(1+beta^2)*Trans1_Syms+beta/sqrt(1+beta^2)*Trans2_Syms;
            tmp2=beta/sqrt(1+beta^2)*Trans1_Syms+1/sqrt(1+beta^2)*Trans2_Syms;
            Trans1_Syms=tmp1;
            Trans2_Syms=tmp2;

         Rayleigh_rx_Syms=Trans1_Syms.*Rayleigh_alpha1_Syms+Trans2_Syms.*
Rayleigh_alpha2_Syms+Rayleigh_noise_Syms;
            [Rayleigh_dec,Rayleigh_dec_NoViterbi]=STTC_DECNew(Rayleigh_rx_Syms);
            Rayleigh_BEC=sum(abs(Rayleigh_dec(1:Rayleigh_Bitlen)-raw_bits));
            Rayleigh_BEC2=sum(abs(Rayleigh_dec_NoViterbi(1:Rayleigh_Bitlen)-
raw_bits));
BECacc_Rayleigh=BECacc_Rayleigh+Rayleigh_BEC;
        end
        BER_Rayleigh=BECacc_Rayleigh/(Rayleigh_Trialnum*Rayleigh_Bitlen);
        BERacc_Rayleigh=BERacc_Rayleigh+BER_Rayleigh;
    end
    BER_SymsInterleave_Rayleigh(SNR-SNR_start+1)=BERacc_Rayleigh/Trailnum;
end
```

In our simulation, correlated transmitting signals are defines as: $\begin{bmatrix} \frac{1}{\sqrt{1+\beta^2}} & \frac{\beta}{\sqrt{1+\beta^2}} \\ \frac{\beta}{\sqrt{1+\beta^2}} & \frac{1}{\sqrt{1+\beta^2}} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$, if $\text{cov}(X_1, X_2) =$ identity, then $\text{cov}(Y_1, Y_2) = \begin{bmatrix} 1 & \frac{2\beta}{1+\beta^2} \\ \frac{2\beta}{1+\beta^2} & 1 \end{bmatrix}$, thus correlation coefficient ρ can be defined as $\rho = \frac{2\beta}{1+\beta^2}$. Or we can generate correlation channel model using following matlab code:

```
FAold=[FA1old(jj); FA2old(jj)]; %independent channels
auto_h=[1 cross; conj(cross) 1]; %introduce correlation
[qq,dd]=eig(auto_h);
lt=qq*sqrt(dd);
FAtemp=lt*FAold; %correlated channels
```

III. Results

A. Simulation Results

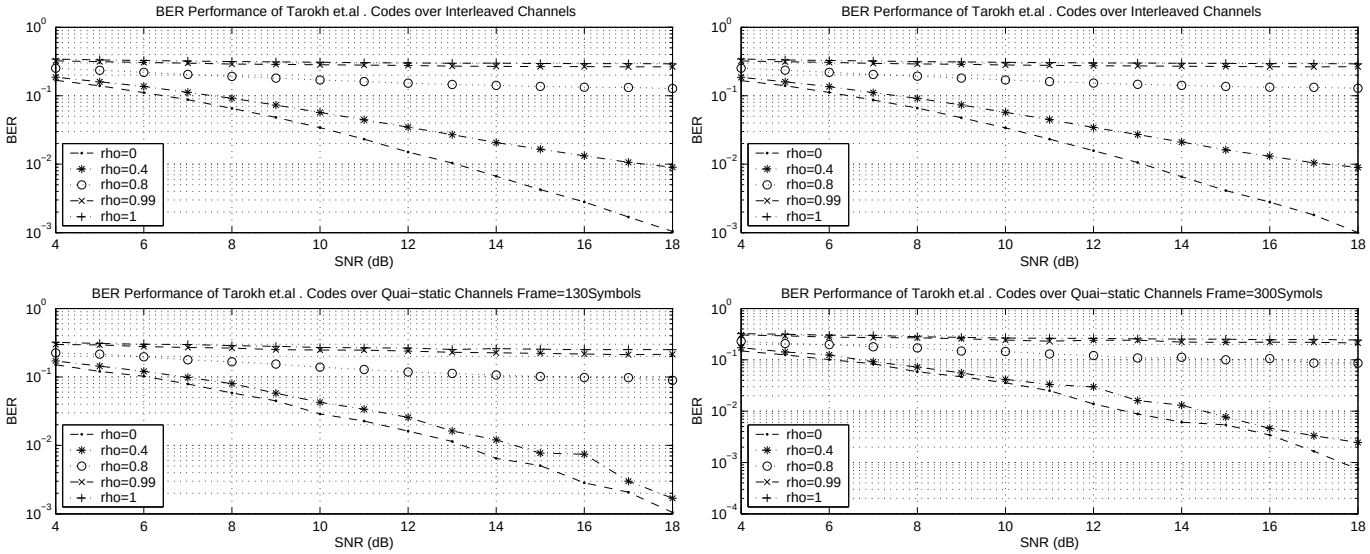

Fig. 1. 4-state 4-PSK space-time trellis code Communication System BER Performance (antenna correlation case)

Above figures show the performance when transmit antenna correlation is taken into consideration. Below shows the performance when channel correlation is taken into consideration only.

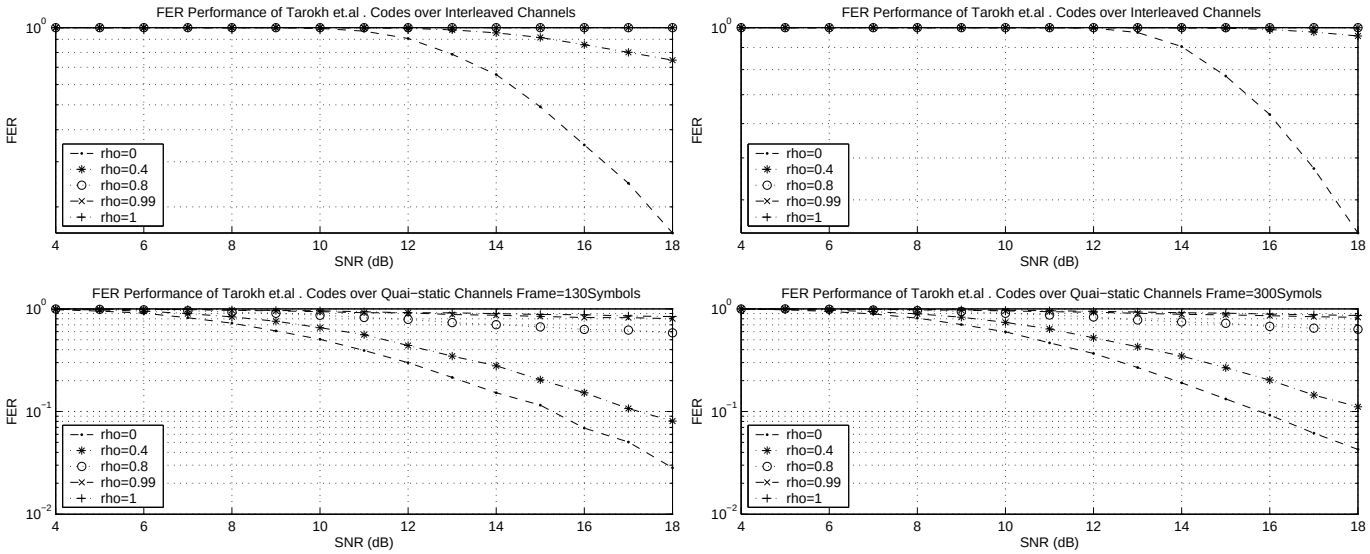

Fig. 2. 4-state 4-PSK space-time trellis code Communication System FER Performance (antenna correlation case)

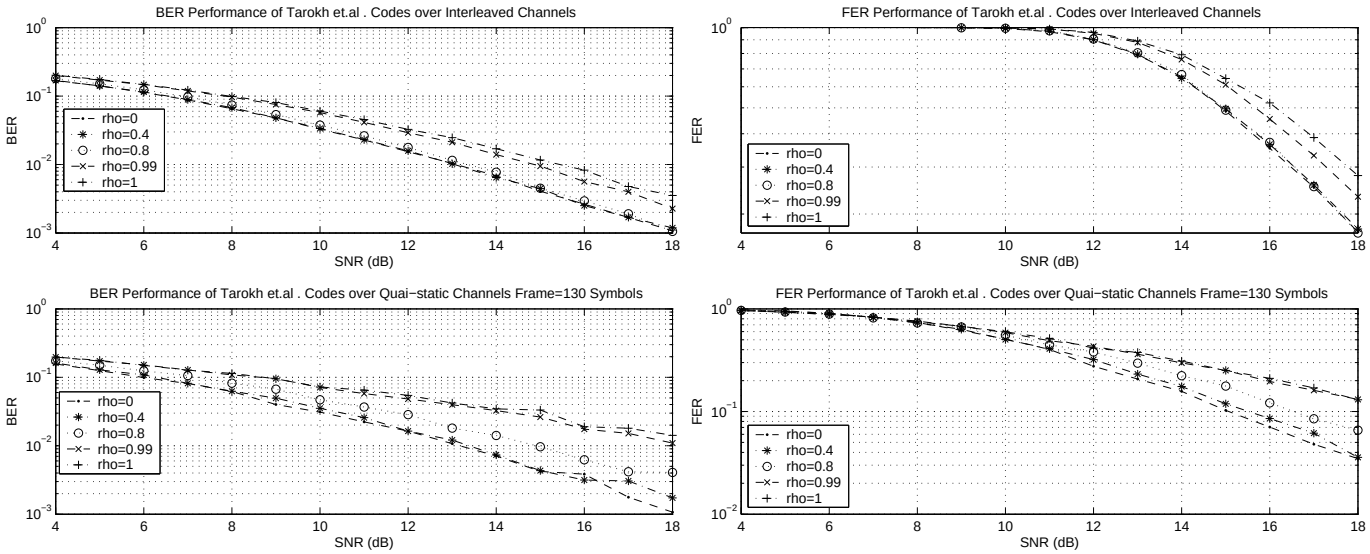

Fig. 3. BER and FER performance when taking channel correlation into account

IV. Conclusions

In my 4-state 4-PSK space-time trellis code Communication System, BER Performance is around 10^{-3}at SNR$=18$ dB which is close to the theoretical value shown in Handout Part V pp. 39. The FER performance for quasi-static channel performs much better than that of symbol-interleaved channel. My quasi-static channel whose frame length 300 perform 4% FER at SNR$=18$ dB which is a little higher than 3% shown in Handout Part V pp. 16. While our simulation for frame length 130 symbols percforms a perfect match to the results shown in the handouts as 3%. It is obvious when taking transmit antenna spatial correlation between transmit antennas into consideration, the performance curves degraded considerably especially when the coefficient near 1 which means both antennas transmit the same signals simultaneously, thus it is well known there will be no transmit diversity at all. However, when we take channel correlation into consideration, actually when correlation coefficients come to be 1, the channel acts as 1 channel in essence, and our viterbi soft decision algorithms still can handle this situation and the performance curves taking form as diversity order of 1 instead of 2.

V. Appendix Viterbi Source Code

The viterbi source code is listed here for reference.

```
global Es;
global Rayleigh_alpha1_Syms;
```

```
global Rayleigh_alpha2_Syms;
jj=sqrt(-1);
SymsSet=exp(jj*2*pi/4*(0:3));
SymsLen=size(Rayleigh_rx_Syms,2);
SymsDemod=zeros(2,SymsLen);
Demod_bits=zeros(1,2*SymsLen);
HardDecision=0;
%%%%%%%%%Start Symbol Domodulation$$$$$$$$$$
for SymsLoop=1:SymsLen
    OrErr = Inf * ones(4,4);
    for MinLoopTx1=1:4
        for MinLoopTx2=1:4
            OrErr(MinLoopTx1,MinLoopTx2)=abs(Rayleigh_rx_Syms(SymsLoop)-sqrt(Es)*
(Rayleigh_alpha1_Syms(SymsLoop)*SymsSet(MinLoopTx1)+Rayleigh_alpha2_Syms
(SymsLoop)*SymsSet(MinLoopTx2)));
        end
    end
    [ColMin, RowIndx] = min(OrErr);
    [ErrMin, ColIndx]=min(ColMin);
    SymsDemod(1,SymsLoop)=RowIndx(ColIndx)-1;
    SymsDemod(2,SymsLoop)=ColIndx-1;
    %%%%%%%%%% Demodulated Symbol mapping into Bits
    Demod_bits(2*SymsLoop-1:2*SymsLoop)=[bitshift(ColIndx-1,-1),ColIndx-1-2
*bitshift(ColIndx-1,-1)];
end
Rayleigh_dec_NoViterbi=Demod_bits(1:2*SymsLen-2);
%%%%%%%%%%%%%%Start Viterbi Decoding Hard-Decision%%%%%%%%%%%%%%%%%%
MemSize=2; %2 bit register 4 state Convolutional Encoder
num_spb=2; % Number of symbols per branch. 4-state-4-PSK we say 2
symbols output per branch
num_states=2^MemSize; %Number of states at one instant in the Trellis
num_trans=4; % Number of transitions 2 bits incoming, cause 4 branches
leaving current state
DecLen=size(Demod_bits,2);
ConstraintLen=1;
TraceBackLen=7*ConstraintLen+1;
%%%Define 6 talbe used in decoding
state_trans_table=[1,2,3,4 ; 1,2,3,4 ;1,2,3,4 ;1,2,3,4];
ENC_output_table=[0,1,2,3 ; 10,11,12,13 ; 20,21,22,23 ; 30,31,32,33];
curr_next_input_table=[0,1,2,3 ; 0,1,2,3 ;0,1,2,3 ;0,1,2,3];
pred_history_table=zeros(num_states,SymsLen); %store the corresponding
previous state of the survivor path
acc_error_metric_table=zeros(num_states,2);% store the accumulated
error metric
traceback_table=zeros(1,TraceBackLen);% Store the lists of states during
traceback procedure
for SymsLoop=1:SymsLen
    if SymsLoop==1
        for InputLoop=1:num_trans
            aux_metric=0;
            tmp=ENC_output_table(1,state_trans_table(1,InputLoop));
            if HardDecision
                BETran1=(bitshift(bitxor(floor(tmp/10),SymsDemod(1,SymsLoop)),-1)
==1)+bitand(bitxor(floor(tmp/10),SymsDemod(1,SymsLoop)),1);
                BETran2=(bitshift(bitxor(tmp-floor(tmp/10)*10,SymsDemod(2,SymsLoop))
,-1)==1)+bitand(bitxor(tmp-floor(tmp/10)*10,SymsDemod(2,SymsLoop)),1);
                aux_metric=BETran1+BETran2+acc_error_metric_table(1,2);
            else
```

```
                MinLoopTx1=floor(tmp/10);
                MinLoopTx2=tmp-floor(tmp/10)*10;
                BE=abs(Rayleigh_rx_Syms(SymsLoop)-sqrt(Es)*(Rayleigh_alpha1_Syms
(SymsLoop)*SymsSet(MinLoopTx1+1)+Rayleigh_alpha2_Syms(SymsLoop)*SymsSet
(MinLoopTx2+1)));
                aux_metric=BE^2+acc_error_metric_table(1,2);
            end


            acc_error_metric_table(state_trans_table(1,InputLoop),1)=aux_metric;
            pred_history_table(state_trans_table(1,InputLoop),SymsLoop)=1;

        end %end of input loop
        acc_error_metric_table(:,2)=acc_error_metric_table(:,1) ;


    elseif SymsLoop<=(TraceBackLen)

        marker=ones(1,num_states); % Represent the times that arrive at
the current state, max value should equal to 4
        for StateLoop=1:num_states
            for InputLoop=1:num_trans

                aux_metric=0;
                tmp=ENC_output_table(StateLoop,state_trans_table(StateLoop,
InputLoop));
%Here state_trans_table(StateLoop,InputLoop)=InputLoop
                if HardDecision
                    BETran1=(bitshift(bitxor(floor(tmp/10),SymsDemod(1,SymsLoop)),-1)
==1)+bitand(bitxor(floor(tmp/10),SymsDemod(1,SymsLoop)),1);
                    BETran2=(bitshift(bitxor(tmp-floor(tmp/10)*10,SymsDemod(2,SymsLoop))
,-1)==1)+bitand(bitxor(tmp-floor(tmp/10)*10,SymsDemod(2,SymsLoop)),1);
                    aux_metric=BETran1+BETran2+acc_error_metric_table(StateLoop,2);
                else
                    MinLoopTx1=floor(tmp/10);
                    MinLoopTx2=tmp-floor(tmp/10)*10;
                    BE=abs(Rayleigh_rx_Syms(SymsLoop)-sqrt(Es)*(Rayleigh_alpha1_Syms
(SymsLoop)*SymsSet(MinLoopTx1+1)+Rayleigh_alpha2_Syms(SymsLoop)*
SymsSet(MinLoopTx2+1)));
                    aux_metric=BE^2+acc_error_metric_table(StateLoop,2);
                end

                if marker(state_trans_table(StateLoop,InputLoop))==1
%The first time arrive at this state, just store the metric as smallest
one
                    acc_error_metric_table(state_trans_table(StateLoop,InputLoop),1)
=aux_metric;
                    pred_history_table(state_trans_table(StateLoop,InputLoop),SymsLoop)
=StateLoop;
                    marker(state_trans_table(StateLoop,InputLoop))=0;
                elseif aux_metric<acc_error_metric_table(state_trans_table(StateLoop,
InputLoop),1)
% means exists another path having smaller error metrix, update
                    acc_error_metric_table(state_trans_table(StateLoop,InputLoop),1)
=aux_metric;
                    pred_history_table(state_trans_table(StateLoop,InputLoop),SymsLoop)
=StateLoop;
                elseif aux_metric==acc_error_metric_table(state_trans_table(StateLoop,
```

```
InputLoop),1)
% random select one if current minimum metric have 2 paths
                    if rand(1)<0.5
                        acc_error_metric_table(state_trans_table(StateLoop,InputLoop),1)
=aux_metric;
                        pred_history_table(state_trans_table(StateLoop,InputLoop),SymsLoop)
=StateLoop;
                    end
                end

            end %end of input loop
        end  %end of stateloop
        acc_error_metric_table(:,2)=acc_error_metric_table(:,1);
        % It is very important when change to next time snap, we keep
the table always have two copies of minimum metric,
acc_error_metric_table(:,2) is used for next symbol
        %loop as reference, acc_error_metric_table(:,1)is used to store
the updated error metric, when update is finished, make
acc_error_metric_table(:,2)=acc_error_metric_table(:,1)

    elseif (SymsLoop<(SymsLen)) & (SymsLoop>(TraceBackLen))

        [Minvalue,MinIndx]=min(acc_error_metric_table(:,1));
        traceback_table(TraceBackLen)=MinIndx; % Means zero state
        for TraceLoop=TraceBackLen-1:-1:1
            traceback_table(TraceLoop)=pred_history_table(traceback_table(TraceLoop+
1),SymsLoop-(TraceBackLen-TraceLoop));
        end

        if SymsLoop==(TraceBackLen+1)
            tmp=curr_next_input_table(1,traceback_table(1));
            Rayleigh_dec(2*(SymsLoop-TraceBackLen)-1)=bitshift(tmp,-1);
            Rayleigh_dec(2*(SymsLoop-TraceBackLen))=tmp-2*bitshift(tmp,-1);
        else
            tmp=curr_next_input_table(pred_state,traceback_table(1));
            Rayleigh_dec(2*(SymsLoop-TraceBackLen)-1)=bitshift(tmp,-1);
            Rayleigh_dec(2*(SymsLoop-TraceBackLen))=tmp-2*bitshift(tmp,-1);
        end
        pred_state=traceback_table(1);
        marker=ones(1,num_states); % Represent the times that arrive at
the current state, max value should equal to 4
        for StateLoop=1:num_states
            for InputLoop=1:num_trans
                %Compute the error metric between received symbols bits
and coded branch
                aux_metric=0;
                tmp=ENC_output_table(StateLoop,state_trans_table(StateLoop,
InputLoop));
%Here state_trans_table(StateLoop,InputLoop)=InputLoop
                if HardDecision
                    BETran1=(bitshift(bitxor(floor(tmp/10),SymsDemod(1,SymsLoop)),-1)
==1)+bitand(bitxor(floor(tmp/10),SymsDemod(1,SymsLoop)),1);
                    BETran2=(bitshift(bitxor(tmp-floor(tmp/10)*10,SymsDemod(2,
SymsLoop)),-1)==1)+bitand(bitxor(tmp-floor(tmp/10)*10,
SymsDemod(2,SymsLoop)),1);
                    aux_metric=BETran1+BETran2+acc_error_metric_table(StateLoop,2);
                else
                    MinLoopTx1=floor(tmp/10);
```

```
                    MinLoopTx2=tmp-floor(tmp/10)*10;
                    BE=abs(Rayleigh_rx_Syms(SymsLoop)-sqrt(Es)*(Rayleigh_alpha1_Syms
(SymsLoop)*SymsSet(MinLoopTx1+1)+Rayleigh_alpha2_Syms(SymsLoop)*
SymsSet(MinLoopTx2+1)));
                    aux_metric=BE^2+acc_error_metric_table(StateLoop,2);
                end

                if marker(state_trans_table(StateLoop,InputLoop))==1
%The first time arrive at this state, just store the metric as smallest
one
                    acc_error_metric_table(state_trans_table(StateLoop,InputLoop),1)
=aux_metric;
                    pred_history_table(state_trans_table(StateLoop,InputLoop),SymsLoop)
=StateLoop;
                    marker(state_trans_table(StateLoop,InputLoop))=0;
                elseif aux_metric<acc_error_metric_table(state_trans_table(StateLoop,
InputLoop),1)
% means exists another path having smaller error metrix, update
                    acc_error_metric_table(state_trans_table(StateLoop,InputLoop),1)
=aux_metric;
                    pred_history_table(state_trans_table(StateLoop,InputLoop),
SymsLoop)=StateLoop;
                elseif aux_metric==acc_error_metric_table(state_trans_table(StateLoop,
InputLoop),1)
% random select one if current minimum metric have 2 paths
                    if rand(1)<0.5
                        acc_error_metric_table(state_trans_table(StateLoop,InputLoop),1)
=aux_metric;
                        pred_history_table(state_trans_table(StateLoop,InputLoop),
SymsLoop)=StateLoop;
                    end
                end

            end %end of input loop
        end  %end of stateloop
        acc_error_metric_table(:,2)=acc_error_metric_table(:,1);



    else% The Final symbol
        [Minvalue,MinIndx]=min(acc_error_metric_table(:,1));
        traceback_table(TraceBackLen)=MinIndx; % Means zero state
        for TraceLoop=TraceBackLen-1:-1:1
            traceback_table(TraceLoop)=pred_history_table(traceback_table(TraceLoop
+1),SymsLoop-(TraceBackLen-TraceLoop));
        end

        if SymsLoop==(TraceBackLen+1)
            tmp=curr_next_input_table(1,traceback_table(1));
            Rayleigh_dec(2*(SymsLoop-TraceBackLen)-1)=bitshift(tmp,-1);
            Rayleigh_dec(2*(SymsLoop-TraceBackLen))=tmp-2*bitshift(tmp,-1);
        else
            tmp=curr_next_input_table(pred_state,traceback_table(1));
            Rayleigh_dec(2*(SymsLoop-TraceBackLen)-1)=bitshift(tmp,-1);
            Rayleigh_dec(2*(SymsLoop-TraceBackLen))=tmp-2*bitshift(tmp,-1);
        end
        pred_state=traceback_table(1);
        marker=ones(1,num_states); % Represent the times that arrive at
```

```
the current state, max value should equal to 4
        for StateLoop=1:num_states
            for InputLoop=1:num_trans
                %Compute the error metric between received symbols bits
and coded branch
                aux_metric=0;
                tmp=ENC_output_table(StateLoop,state_trans_table(StateLoop,
InputLoop));
%Here state_trans_table(StateLoop,InputLoop)=InputLoop
                if HardDecision
                    BETran1=(bitshift(bitxor(floor(tmp/10),SymsDemod(1,SymsLoop)),-1)==1)+
bitand(bitxor(floor(tmp/10),SymsDemod(1,SymsLoop)),1);
                    BETran2=(bitshift(bitxor(tmp-floor(tmp/10)*10,SymsDemod(2,SymsLoop))
,-1)==1)+bitand(bitxor(tmp-floor(tmp/10)*10,SymsDemod(2,SymsLoop)),1);
                    aux_metric=BETran1+BETran2+acc_error_metric_table(StateLoop,2);
                else
                    MinLoopTx1=floor(tmp/10);
                    MinLoopTx2=tmp-floor(tmp/10)*10;
                    BE=abs(Rayleigh_rx_Syms(SymsLoop)-sqrt(Es)*(Rayleigh_alpha1_Syms
(SymsLoop)*SymsSet(MinLoopTx1+1)+Rayleigh_alpha2_Syms(SymsLoop)*
SymsSet(MinLoopTx2+1)));
                    aux_metric=BE^2+acc_error_metric_table(StateLoop,2);
                end

                if marker(state_trans_table(StateLoop,InputLoop))==1
%The first time arrive at this state, just store the metric as smallest
one
                    acc_error_metric_table(state_trans_table(StateLoop,InputLoop),1)
=aux_metric;
                    pred_history_table(state_trans_table(StateLoop,InputLoop),SymsLoop)
=StateLoop;
                    marker(state_trans_table(StateLoop,InputLoop))=0;
                elseif aux_metric<acc_error_metric_table(state_trans_table(StateLoop,
InputLoop),1)
% means exists another path having smaller error metrix, update
                    acc_error_metric_table(state_trans_table(StateLoop,InputLoop),1)]
=aux_metric;
                    pred_history_table(state_trans_table(StateLoop,InputLoop),
SymsLoop)=StateLoop;
                elseif aux_metric==acc_error_metric_table(state_trans_table(StateLoop,
InputLoop),1)
% random select one if current minimum metric have 2 paths
                    if rand(1)<0.5
                        acc_error_metric_table(state_trans_table(StateLoop,InputLoop),1)
=aux_metric;
                        pred_history_table(state_trans_table(StateLoop,InputLoop),
SymsLoop)=StateLoop;
                    end
                end

            end %end of input loop
        end  %end of stateloop
        acc_error_metric_table(:,2)=acc_error_metric_table(:,1);

        [Minvalue,MinIndx]=min(acc_error_metric_table(:,1));
        traceback_table(TraceBackLen)=1; % Means zero state
        for TraceLoop=TraceBackLen-1:-1:1
            traceback_table(TraceLoop)=pred_history_table(traceback_table(TraceLoop
```

```
+1),SymsLoop+1-(TraceBackLen-TraceLoop));
        end
        tmp=curr_next_input_table(pred_state,traceback_table(1));
        Rayleigh_dec(2*(SymsLoop-TraceBackLen+1)-1)=bitshift(tmp,-1);
        Rayleigh_dec(2*(SymsLoop-TraceBackLen+1))=tmp-2*bitshift(tmp,-1);
        for DecLoop=2:TraceBackLen
            tmp=curr_next_input_table(traceback_table(DecLoop-1),
traceback_table(DecLoop));
            Rayleigh_dec(2*(SymsLoop-TraceBackLen+DecLoop)-1)=bitshift(tmp,-1);
            Rayleigh_dec(2*(SymsLoop-TraceBackLen+DecLoop))=tmp-2*bitshift(tmp,-1);
        end

    end %end of if
end %end of whole loop
```

REFERENCES

[1] Prof. Murat Uysal, “ECE710 Space-time Coding for Wireless Communication handouts”, Spring, 2004