V-BLAST MIMO Decoders

Yara Mohsen Mahmoud

ITI intake41 | wireless communication | 22 Fed 2021

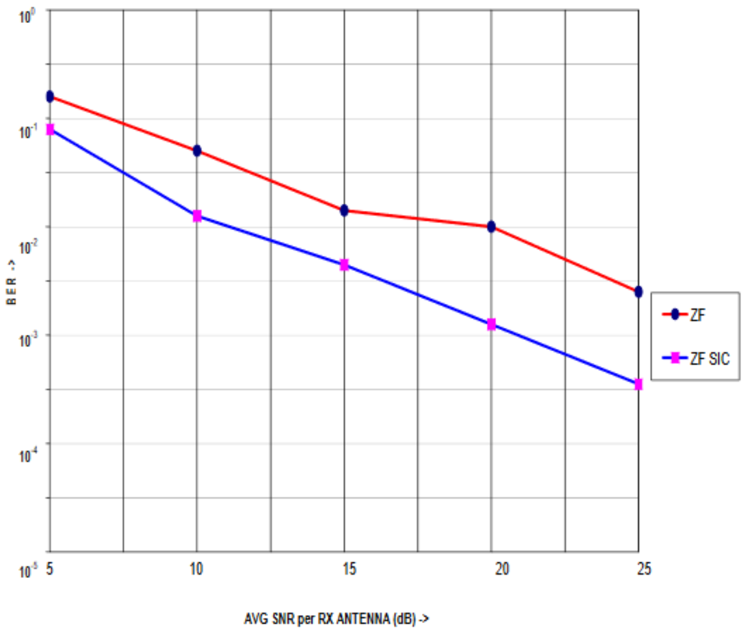
4 × 4 MIMO system on BPSK

simulate and compare the performance of different VBLAST (spatial multiplexing)

- Zero-forcing (ZF) Decoder
- MMSE Decoder
- ZF-SIC Decoder
- MMSE-SIC Decoder
- ML decoding using exhaustive search

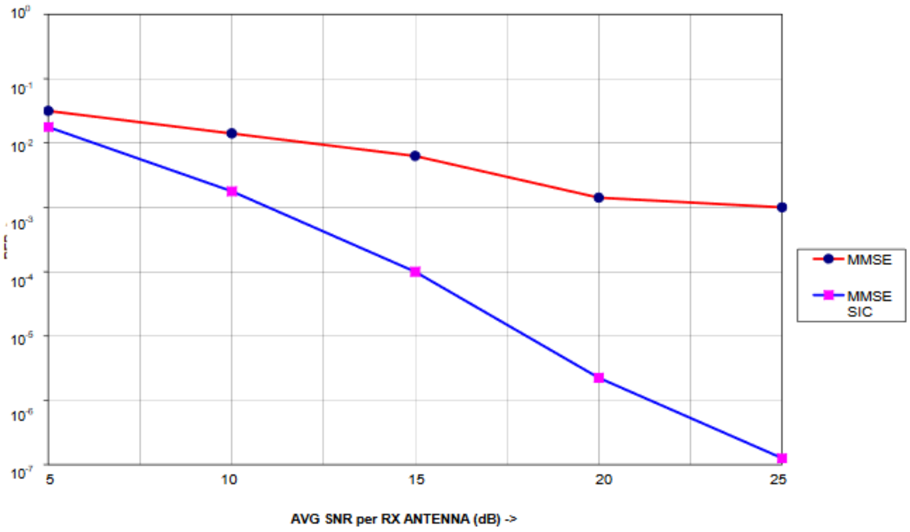
This is an expected results from a paper, compared between the behavior of " ZF , ZF SIC " and "MMSE , MMSE SIC" – See References

ZF and ZF_SIC



BER analysis 4x4 MIMO using BPSK for ZF and ZF SIC Receiver.

MMSE and MMSE_SIC

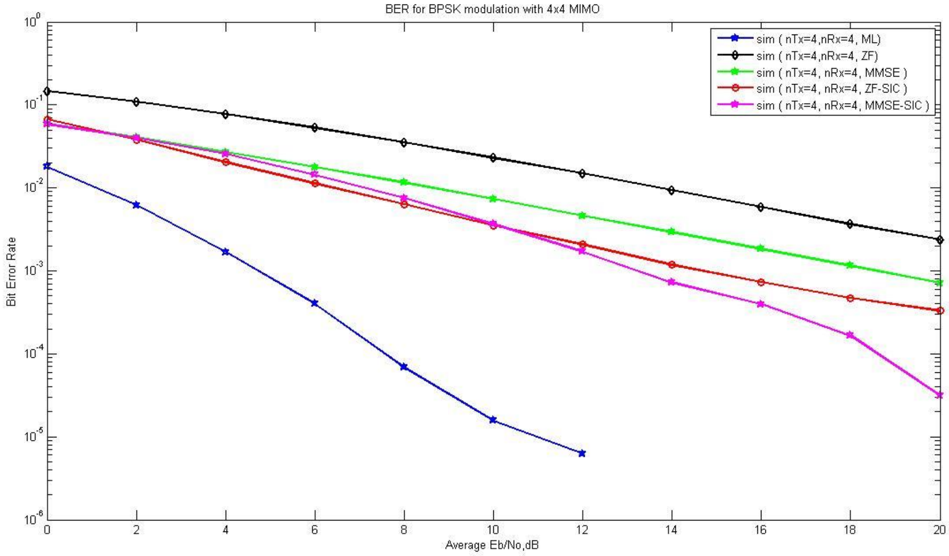


BER analysis of 4x4 MIMO using BPSK for MMSE and MMSE SIC based receiver

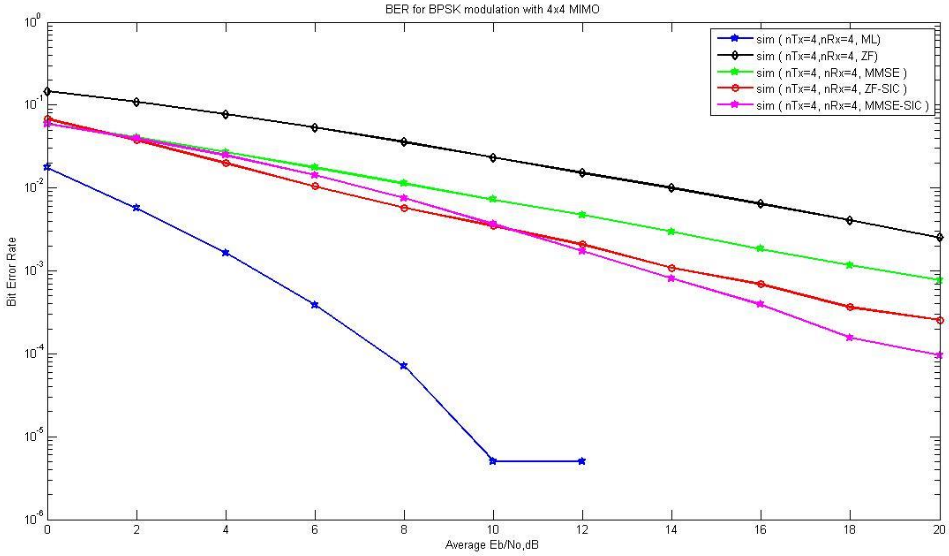
My Code and Graphs

Results:

At 80,000 sample



At 50,000 sample



1. ZF_SIC is better than ZF
2. MMSE_SIC is better than MMSE
3. MMSE_SIC is better than ZF_SIC
4. The performance of MIMO system with MMSE-SIC receiver is not only better than MMSE,ZF and ZF-SIC receiver but also provide better overall system performance with the increasing diversity order .

CONTENTS

4*4 MIMO DETECTOR

```
clear all;
close all;

EB_N0_dB=0:2:20;
EB_N0=1./(10.^(EB_N0_dB/10));
M=4; %numbers of tx antennas

%%bitstream generation and BPSK symbols
bitsNum=1;
Bits1=randi([0,1],1,bitsNum);
Bits2=randi([0,1],1,bitsNum);
Bits3=randi([0,1],1,bitsNum);
Bits4=randi([0,1],1,bitsNum);

Symb1=Bits1*2-1;
Symb2=Bits2*2-1;
Symb3=Bits3*2-1;
Symb4=Bits4*2-1;

%arrange the signals in 4*1*K dim -> k=bitsNum
S(1,1,:)=Symb1;
S(2,1,:)=Symb2;
S(3,1,:)=Symb3;
S(4,1,:)=Symb4;

sample=[[1;1;1;1],[1;1;1;-1],[1;1;-1;1],[1;1;-1;-1],[1;-1;1;1],[1;-1;1;-1],[1;-1;-
1;1],[1;-1;-1;-1],[-1;1;1;1],[-1;1;1;-1],[-1;1;-1;1],[-1;1;-1;-1],[-1;-1;1;1],[-1;-
1;1;-1],[-1;-1;-1;1],[-1;-1;-1;-1]];

%noise generation
N1=randn(4,1,bitsNum)+i*randn(4,1,bitsNum);
H=sqrt(1/2).*(randn(4,4,bitsNum)+i*randn(4,4,bitsNum));
Y=zeros(4,1,bitsNum);
```

```
%starting the detection at reciever
%variables to get the bit error rates
BER_ML=[];
BER_MMSE=[];
BER_MMSE_SIC=[];
BER_ZF=[];
BER_ZF_SIC=[];



for v=EB_N0
    n1=N1.*sqrt(v/2);

    error_ML=0;

    error_ZF=0;

    error_MMSE=0;

    error_ZF_SIC1=0;

    error_ZF_SIC=0;

    error_MMSE_SIC1=0;

    error_MMSE_SIC=0;

    for k=1:bitsNum
       k

        %the output from the rx antennas
        Y(:,:,k)=H(:,:,k)*S(:,:,k)+n1(:,:,k);
```

ML

```
        d1=norm(Y(:,:,k)-H(:,:,k)*sample(:,1),4).^2;

        d2=norm(Y(:,:,k)-H(:,:,k)*sample(:,2),4).^2;

        d3=norm(Y(:,:,k)-H(:,:,k)*sample(:,3),4).^2;

        d4=norm(Y(:,:,k)-H(:,:,k)*sample(:,4),4).^2;



        d5=norm(Y(:,:,k)-H(:,:,k)*sample(:,5),4).^2;

        d6=norm(Y(:,:,k)-H(:,:,k)*sample(:,6),4).^2;

        d7=norm(Y(:,:,k)-H(:,:,k)*sample(:,7),4).^2;

        d8=norm(Y(:,:,k)-H(:,:,k)*sample(:,8),4).^2;
```

```
d9=norm(Y(:,:,k)-H(:,:,k)*sample(:,9),4).^2;
d10=norm(Y(:,:,k)-H(:,:,k)*sample(:,10),4).^2;
d11=norm(Y(:,:,k)-H(:,:,k)*sample(:,11),4).^2;
d12=norm(Y(:,:,k)-H(:,:,k)*sample(:,12),4).^2;

d13=norm(Y(:,:,k)-H(:,:,k)*sample(:,13),4).^2;
d14=norm(Y(:,:,k)-H(:,:,k)*sample(:,14),4).^2;
d15=norm(Y(:,:,k)-H(:,:,k)*sample(:,15),4).^2;
d16=norm(Y(:,:,k)-H(:,:,k)*sample(:,16),4).^2;

D=[d1;d2;d3;d4;d5;d6;d7;d8;d9;d10;d11;d12;d13;d14;d15;d16];
[m,Ind]=min(D);
error_ML=error_ML+sum(S(:,:,k)~=sample(:,Ind));
```

MMSE

```
Gmmse=sqrt(1/M).*H(:,:,k)'*inv(v.*eye(M)+((1/M).*H(:,:,k)*H(:,:,k)'));
S_MMSE=sqrt(1/2)*Gmmse(:,:)*Y(:,:,k);
S_MMSE=real(S_MMSE);
S_MMSE(S_MMSE<0)=-1;
S_MMSE(S_MMSE>0)=1;
error_MMSE=error_MMSE+sum(S(:,:,k)~=S_MMSE);
```

ZF

```
         S2=sqrt(1/2)*inv(H(:,:,k))*Y(:,:,k);

         S2=real(S2);

         S2(S2<0)=-1;

         S2(S2>0)=1;

         error_ZF=error_ZF+sum(S(:,:,k)~=S2);
```

ZF_SIC

```
    %copy of the original signal to make changes on it
     singnal_copy1=S(:,:,k);
     %copy of the original channels to make changes on it
      H_copy_ZF_SIC=H(:,:,k);
     %the output of the 4 rx antennas of ZF_SIC
      Y_copy_ZF_SIC=H_copy_ZF_SIC*singnal_copy1+n1(:,:,k);

     for index=1:4    % for to loop on the 4 signails

         %calculate g and get the min normal
         G_ZF_SIC=(inv(H_copy_ZF_SIC' *H_copy_ZF_SIC))*H_copy_ZF_SIC';
         [~,h1]=min(sum(abs(G_ZF_SIC).^2,2));

         %detect the signal number=ind
         S_ZF_SIC=sqrt(1/2).*G_ZF_SIC(h1,:)*Y_copy_ZF_SIC;

         S_ZF_SIC=real(S_ZF_SIC);
         S_ZF_SIC(S_ZF_SIC<0)=-1;
         S_ZF_SIC(S_ZF_SIC>0)=1;

         %remove the detected signal from y
         Y_copy_ZF_SIC=Y_copy_ZF_SIC-H_copy_ZF_SIC(:,h1)*S_ZF_SIC;

         %count the error
         error_ZF_SIC1=error_ZF_SIC1+sum(singnal_copy1(h1,:)~=S_ZF_SIC);

         %update the signal and remove the detected signal
         %and the detected channel
         singnal_copy1(h1,:)=[];
         H_copy_ZF_SIC(:,h1)=[];

     end
```

MMSE_SIC

```
    singnal_copy2=S(:,:,k);           %copy of the original signal to make changes on
it
    H_copy_MMSE_SIC=H(:,:,k);         %copy of the original channels to make changes on
it
    Y_copy_MMSE_SIC=H_copy_MMSE_SIC*singnal_copy2+n1(:,:,k);    %the output of the 4 rx
antennas of MMSE_SIC

    for ind=1:4

        %calculate g and get the min normal

G_MMSE_SIC=sqrt(1/M).*H_copy_MMSE_SIC'*inv(v.*eye(M)+((1/M).*H_copy_MMSE_SIC*H_copy_MMS
E_SIC'));
        [~,h2]=min(sum(abs(transpose(G_MMSE_SIC)).^2));

        %detect the signal number=ind
        S_MMSE_SIC=sqrt(1/2).*G_MMSE_SIC(h2,:)*Y_copy_MMSE_SIC;

        S_MMSE_SIC=real(S_MMSE_SIC);
        S_MMSE_SIC(S_MMSE_SIC<0)=-1;
        S_MMSE_SIC(S_MMSE_SIC>0)=1;

        Y_copy_MMSE_SIC=Y_copy_MMSE_SIC-H_copy_MMSE_SIC(:,h2)*S_MMSE_SIC;

        %count the error
        error_MMSE_SIC1=error_MMSE_SIC1+sum(singnal_copy2(h2,:)~=S_MMSE_SIC);

        %update the signal and remove the detected signal
        %and the detected channel
        singnal_copy2(h2,:)=[];
        H_copy_MMSE_SIC(:,h2)=[];

    end
    end
```

CALCULATE ERROE

```
    %ML
    error_ML=error_ML/(4*bitsNum);
    BER_ML=[BER_ML error_ML];

    % ZF
     error_ZF=error_ZF/(4*bitsNum);
    BER_ZF=[BER_ZF error_ZF];

    % MMSE
    error_MMSE=error_MMSE/(4*bitsNum);
    BER_MMSE=[BER_MMSE error_MMSE];

    %ZF_SIC
    error_ZF_SIC=error_ZF_SIC1/(4*bitsNum);
    BER_ZF_SIC=[BER_ZF_SIC error_ZF_SIC];

    %MMSE_SIC
    error_MMSE_SIC=error_MMSE_SIC1/(4*bitsNum);
    BER_MMSE_SIC=[BER_MMSE_SIC error_MMSE_SIC];
end


%%drawing
snr=10*log10(1./EB_N0);
grid on;
semilogy(snr,BER_ML,'bp-','LineWidth',2);
hold on;
semilogy(snr,BER_ZF,'kd-','LineWidth',2);
hold on;
semilogy(snr,BER_MMSE,'gp-','LineWidth',2);
hold on;
semilogy(snr,BER_ZF_SIC,'ro-','LineWidth',2);
hold on;
semilogy(snr,BER_MMSE_SIC,'mp-','LineWidth',2);
hold on;

legend('sim ( nTx=4,nRx=4, ML)', 'sim ( nTx=4,nRx=4, ZF)', 'sim ( nTx=4, nRx=4, MMSE
)','sim ( nTx=4, nRx=4, ZF-SIC )','sim ( nTx=4, nRx=4, MMSE-SIC )');
xlabel('Average Eb/No,dB');
ylabel('Bit Error Rate');
title('BER for BPSK modulation with 4x4 MIMO');
```

Where->G - Equalization Matrix , H - Channel Matrix , n - Channel noise ,y- Received signal.

References

1) www.mathworks.com
2) https://ijcset.net/docs/Volumes/volume1issue8/ijcset2011010819.pdf
3) Video of lec5 of our course