中山大學

SUN YAT-SEN UNIVERSITY

数值计算
期中大作业

专业：通信工程

学号：22309080

姓名：梁倍铭

时间：2023.1.18

目录

1 前言

在期中报告中已完成了对《[1] Efficient Algorithm for Detecting Layered Space Time Codes》和《[2] MMSE Extension of V-BLAST based on Sorted QR Decomposition》这两篇论文中 V-BLAST、QRD、SQRD、ZF、MMSE、MMSE-QRD 和 MMSE-SQRD 算法的仿真复现。

在期末报告中，将完成对 MMSE-QRD-PSA、基于信道矩阵 SVD 分解的发送与检测、基于信道矩阵 GMD 分解的发送与检测的仿真实现，并通过计算误码率和误帧率来比较他们的性能差异。

2 系统模型

系统模型为 MIMO 系统，在期中报告中已经给出。

MIMO 全称是 multiple-in multiple-out，多输入多输出。示意图如图 1, 具有 N_T 根发射天线和 N_R 根接收天线，且 $N_R > N_T$，数据在 N_T 个等长的数据子流（称为层）中进行解复用。这些子流被映射成 M-PSK 或 M-QAM 符号, 或者，可以使用前向纠错（FEC）码在映射之前对数据子流进行编码。

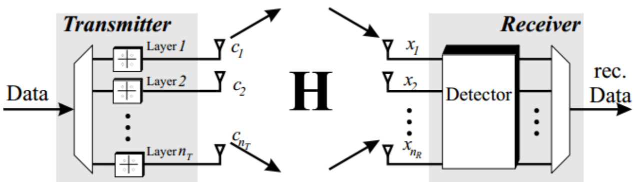

图 1: MIMO 系统示意图

用 $c = \begin{pmatrix} c_1 & c_2 & \dots & c_{nt} \end{pmatrix}^\top$ 来表示发射的信号，用 $x = \begin{pmatrix} x_1 & x_2 & \dots & x_{nr} \end{pmatrix}$ 来表示接收到的信号。信道 H 的大小为 $N_T * N_R$，信道 H 可表示为

$$H = \begin{pmatrix} h_{1,1} & \dots & h_{1,nT} \\ \dots & \dots & \dots \\ h_{nR,1} & \dots & h_{nR,nT} \end{pmatrix}$$

由于两篇论文所使用的符号不尽相同，这里采用第一篇论文的符号进行统一描述，用 v 来表示接收天线中的高斯白噪声，$v = \begin{pmatrix} v_1 & v_2 & \dots & v_{NR} \end{pmatrix}^\top$，因此，整个系统可以用 $x = Hc + v$ 来描述。

其中我们假设假设所有天线每维的方差 N0=2 的不相关高斯白噪声。传

输的符号被归一化，使得每比特的平均接收能量为一。我们假设静态平坦衰落环境，即信道矩阵 H 在帧内保持不变，并且在帧与帧之间独立变化。假定不同的衰落增益是不相关的并且接收机完全了解这些增益。

3 算法实现

3.1 MMSE-SQRD 算法的仿真实现

MMSE 算法通过最小化实际传输符号与线性检测器输出之间的均方误差 (MSE)，并得出滤波器矩阵：

$$G_{MMSE} = (H^H H + \sigma^2 I_{nT})^{-1} H^H$$

进而得到检测信号：

$$\hat{c}_{MMSE} = H^{+} x$$

结合 QR 算法，为了获得最优的检测顺序，可以通过在每个正交化步骤之前对信道矩阵的列进行重新排序，来提高检测性能。伪代码如图 2.

$\underline{\mathbf{R}} = \mathbf{0}$, $\underline{\mathbf{Q}} = \underline{\mathbf{H}}$, $\mathbf{p} = (1, \ldots, n_T)$
for $i = 1, \ldots, n_T$
    $\mathbf{norm}_i = \|\underline{\mathbf{q}}_i\|^2$
end
for $i = 1, \ldots, n_T$
    $k_i = \arg\min_{\ell = i, \ldots, n_T} \mathbf{norm}_\ell$
    exchange columns i and k_i in $\underline{\mathbf{R}}$, $\mathbf{p}$, $\mathbf{norm}$ and in the first $n_R + i - 1$ rows of $\underline{\mathbf{Q}}$
    $r_{i,i} = \sqrt{\mathbf{norm}_i}$
    $\underline{\mathbf{q}}_i := \underline{\mathbf{q}}_i / \underline{r}_{i,i}$
    for $k = i + 1, \ldots, n_T$
        $r_{i,k} = \underline{\mathbf{q}}_i^H \cdot \underline{\mathbf{q}}_k$
        $\underline{\mathbf{q}}_k := \underline{\mathbf{q}}_k - \underline{r}_{i,k} \cdot \underline{\mathbf{q}}_i$
        $\mathbf{norm}_k := \mathbf{norm}_k - \underline{r}_{i,k}^2$
    end
end

图 2: MMSE-SQRD 伪代码

matlab 代码如图 3.

```
function C = mmse_sqrd_fun(H,x,n)
%MMSE_SQRD_FUN 该函数实现MMSE-QRD算法
%   输入参数H: 信道矩阵
%   输入参数x: 接收到的信号
%   输入参数n: 噪声的标准差
%   输出参数C: 解调出来的信号矩阵
[~,col]=size(H);
I=n*eye(col);
H=[H;I];%对信道矩阵进行扩展
x=[x;zeros(col,1)];%对接收的信号进行处理
[row,col]=size(H);
c = zeros(col, 1);
C = zeros(col, 1);
q = H;
r = zeros(col, col);
s = 1:col; % 用于记录排序的索引

% 进行QR 分解
for i2 = 1:col
    min_norm = norm(q(:, i2));  % 初始化最小范数
    k = i2;
    % 寻找最小范数列
    for j2 = i2:col
        if min_norm > norm(q(:, j2))
            min_norm = norm(q(:, j2));
            k = j2;
        end
    end
    % 交换列, 更新 QR 分解矩阵
    q(:, [i2, k]) = q(:, [k, i2]);
    r(:, [i2, k]) = r(:, [k, i2]);
    s([i2, k]) = s([k, i2]);

    r(i2, i2) = norm(q(:, i2));
    q(:, i2) = q(:, i2) / r(i2, i2);
    r(i2, i2) = norm(q(:, i2));
    q(:, i2) = q(:, i2) / r(i2, i2);

    % 更新 R 矩阵的剩余部分
    for l = i2+1:col
        r(i2, l) = q(:, i2)' * q(:, l);
        q(:, l) = q(:, l) - r(i2, l) * q(:, i2);
    end
end
y = q' * x;  % 计算中间变量 y
d = zeros(row, 1);
z = zeros(row, 1);
% 反向解调
for i = col:-1:1
    for j = i+1:col
        d(i) = d(i) + r(i, j) * c(j);  % 计算干扰项
    end

    z(i) = y(i) - d(i);  % 消除干扰项
    k = z(i) / r(i, i);%得到解调出来的原始信号
    if k < 0.5%对原始信号进行判决
        c(i) = 0;
    else
        c(i) = 1;
    end
end
% 根据排序结果将解调后的结果排序并返回
for i4 = 1:length(s)
    C(s(i4)) = c(i4);
end
end
```

图 3: MMSE-SQRD 伪代码

3.2 MMSE-SQRD-PSA 算法的仿真实现

由于协方差误差矩阵 Φ 可以写成 $\Phi = Q_2 Q_2^H$，由于 Q_2 是上三角的，Φ 的第 k 个对角元素与 Q_2 第 k 行的范数成正比, 在最优排序算法中，Q_2 的最后一行必须具有所有行的最小范数。

若满足该条件，则 Q_2 左上 $n_T - 1 \times n_T - 1$ 子矩阵的最后一行必须具有该子矩阵所有行的最小范数，如果排序正确，则此条件由所有左上子矩阵完成。

若矩阵 Q_2 不满足该条件。那么范数最小的行和最后一行需要交换，通过将 Q_2 的置换版本与适当的酉 $n_T \times n_T$Householder 反射矩阵 (Φ) 相乘得到分块三角矩阵。最后，Q_1 更新为 $Q_1\Phi$，在 PSA 结束时反转 Q_2。

然后针对修改后的矩阵 Q_2 的左上 $n_T - 1 \times n_T - 1$ 子矩阵和新矩阵 Q_1 的前 $n_T - 1$ 列迭代这些排序和反射步骤，从而产生 QR 分解最优排序的信道矩阵 H.

伪代码如图 4, 在 MMSE-SQRD 的基础上增加的 matlab 代码如图 5

```
k_min = n_T
for i = n_T, ..., 2
    for ℓ = 1, ..., i
        error_ℓ = ||Q_2(ℓ, 1 : i)||^2
    end
    k_i = arg min_{ℓ=1,...,i} error_ℓ
    k_min = min(k_min, k_i)
    if k_i < i
        exchange rows i and k_i in Q_2 and col. i and k_i in p
    end
    if k_min < i
        calculate Householder reflector Θ such that elements
        of Q_2(i, k_min : i − 1) become zero
        Q_2(1 : i, k_min : i) := Q_2(1 : i, k_min : i)Θ
        Q_1(:, k_min : i) := Q_1(:, k_min : i)Θ
    end
end
R = 1/σ_n Q_2^{-1}
```

图 4: MMSE-SQRD-PSA 伪代码

```matlab
%psa部分
kmin=col;
p=1:col;
for i=col:-1:2
    error=zeros(i,1);
    for l=1:i
        error(l)=vecnorm(Q2(l,1:i),2,2)^2;
    end
    [~,ki]=min(error);
    kmin=min(kmin,ki);
    if ki<i
        Q2([i ki],:)=Q2([ki i],:);
        p([i ki])=p([ki i]);
    end
    if kmin<i
        a=Q2(i,kmin:i);
        e=[zeros(1,length(a)-1),1];
        u=(a-norm(a)*e)/norm(a-norm(a)*e);
        w=(u*a')/(a*u');
        re=eye(length(a))-(1+w)*(u')*u;
        Q2(1:i,kmin:i)=Q2(1:i,kmin:i)*re;
        Q1(:,kmin:i)=Q1(:,kmin:i)*re;
    end
end
R=n*inv(Q2);
y = q' * x;  % 计算中间变量 y
d = zeros(row, 1);
z = zeros(row, 1);
```

图 5: MMSE-SQRD-PSAmatlab 代码

3.3 基于信道矩阵 SVD 分解的发送与检测的仿真实现

奇异值分解 (Singular Value Decomposition，以下简称 SVD) 是一种广泛使用的算法，SVD 并不要求要分解的矩阵为方阵，他将矩阵 (A) 分解为正交矩阵 (U), 对角矩阵 (Σ) 和正交矩阵 (V)。即 $A = U\Sigma V$.

在 MIMO 系统中，可以对信道矩阵进行 SVD 分解

$$H = U\Sigma V$$

系统函数变成

$$x = U\Sigma Vc - v$$

令

$$y = U'x$$

$$s = Vc$$

则

$$y = \Sigma s + U'v$$

所以在接收端和发射端分别乘一个矩阵，目的是为了得到一个干净的传输矩阵，也就是对角线矩阵。

在进行仿真时，我们只需对前面的代码稍作修改，便可实现基于信道矩阵 SVD 分解的发送与检测。假设将信号矩阵 H 分解成 $U\Sigma V$，即 $H = U\Sigma V$，在发送 c 时，左乘 V，如图 6，在接收端，信道矩阵 H 变成了 Σ，将接收到的 x 左乘 U 的逆，如图 7

```
bit_stream_tx=randi([0,1],N_T,1);%发送的比特流
c=bit_stream_tx;
```

```
[~,~,V_]=svd(H);
bit_stream_tx=randi([0,1],N_T,1);%发送的比特流
c=V_*bit_stream_tx;%左乘V矩阵
```

图 6: SVD 发射端对比 matlab 代码

```
[U,S,~]=svd(H);%SVD分解
x=inv(U)*x;%左乘U的逆
H=S;%信号矩阵变为对角阵
```

图 7: SVD 接收端增加的 matlab 代码

3.4 基于信道矩阵 GMD 分解的发送与检测

GMD(几何均值分解) 的思想和 SVD 类似，都是在接收和发射端同时对信号进行处理。处理过程如下：

将信道矩阵进行 GMD 分解：

$$H = QRP'$$

对发送信号进行处理：

$$x = Ps$$

系统模型变为：

$$y = QRs + z$$

对其进行解调即可得到发射信号 s，发射端代码如图 8，接收端代码如图 9.

```
bit_stream_tx=randi([0,1],N_T,1);%发送的比特流
c=bit_stream_tx;
```

```
[U_,S_,V_]=svd(H);
[~,~,P]=gmd(U_,S_,V_);
bit_stream_tx=randi([0,1],N_T,1);%发送的比特流
c=inv(P_)*bit_stream_tx;%左乘V矩阵
```

图 8: GMD 发射端对比 matlab 代码

```
function c = qr_gmd_fun(H,x)
%QR_FUN 此函数是对基于信道矩阵GMD分解的接收端的仿真实现
%   输入参数H: 信道矩阵
%   输入参数x: 接收到的信号
%   输出参数c: 解调出来的信号矩阵
[U,S,V]=svd(H);
[Q,R,~]=gmd(U,S,V);
[~,col]=size(H);%求信道矩阵H的列数
c=zeros(col,1);
y=Q'*x;%对接收到的信号进行处理
d=zeros(col,1);
z=zeros(col,1);
for i=col:-1:1%从最后一层开始检测
    for j=i+1:col%通过迭代获得每一层的干扰项
        d(i,1) = d(i,1) + R(i, j) * c(j,1);
    end
    z(i,1) = y(i,1) - d(i,1);%减去干扰项
    tmp = z(i,1) / R(i, i);%获得原始解调出来的信号
        if tmp < 0.5%对信号进行判决
            c(i,1) = 0;
        else
            c(i,1) = 1;
        end
end
end
```

图 9: GMD 接收端的 matlab 代码

4 对各项算法的测试

4.1 MMSE-SQRD 和 MMSE-SQRD-PSA 与期中各个算法对比

假设接收天线为 8，发射天线为 6，测试 100000 次，并统计各个算法的无码率，测试结果如图 10.

假设接收天线为 12，发射天线为 8，测试 100000 次，并统计各个算法的误码率，测试结果如图 11.

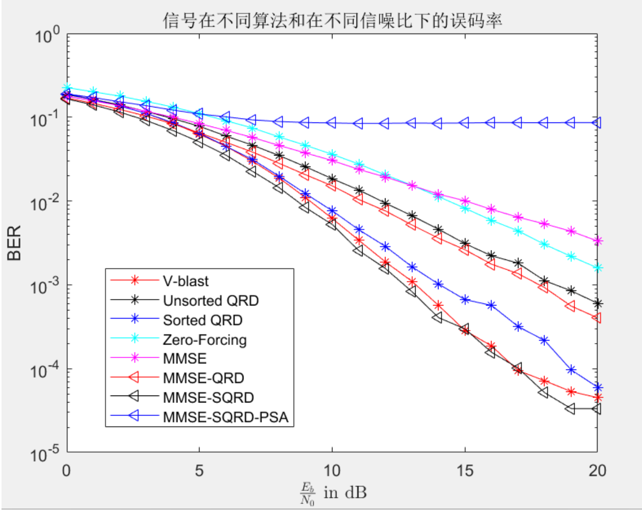

图 10: 测试结果 1

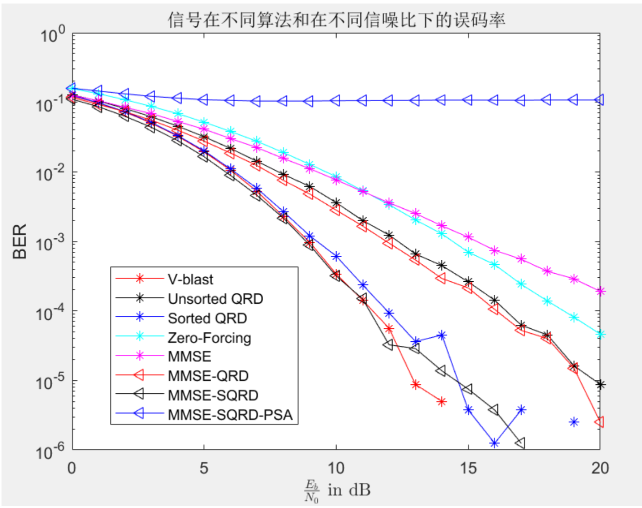

图 11: 测试结果 2

从测试结果可以看出，MMSE-SQRD 算法和 V-BLAST 算法的误码率最低，然后到 SQRD 算法，紧接着是 MMSE-QRD 和 QRD 算法, 最后是 MMSE 和 ZF 算法，结合算法复杂度，不考虑 MMSE-SQRD-PSA 的情况下，MMSE-SQRD 的综合性能最好。图中观察可知 MMSE-SQRD-PSA 的算法误码率明显偏高，单独测试 MMSE-SQRD-PSA 算法，测试结果如图 12.

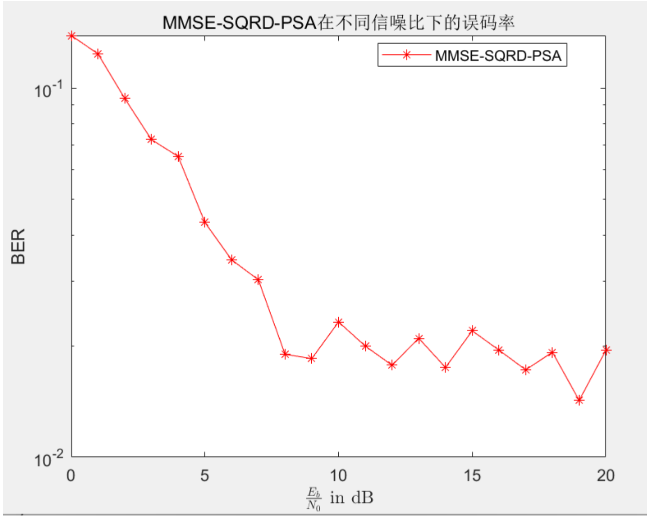

图 12: 测试结果 3

从测试结果来看，MMSE-SQRD-PSA 算法误码率在 0.01 和 0.1 之间，说明还是存在问题，但是 debug 了好久，仍然没有发现出错的地方，因此此处没有完全完成。

4.2 SVD、GMD 与其他算法对比

4.2.1 对比 SVD 对 V-blast 和 ZF 算法的影响

假设接收天线为 8，发射天线为 6，测试 100000 次，并统计各个算法的无码率，测试结果如图 13.

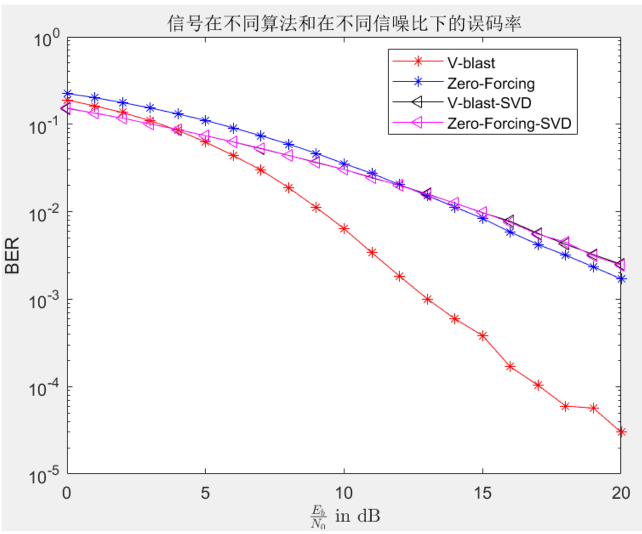

图 13: 测试结果 4

从图中可以看出，进行 SVD 分解后，可能会导致误码率偏高，但是由于信号矩阵变为了对角阵，大大简化了运算

4.2.2 对比 GMD 对 QRD 算法的影响

假设接收天线为 8，发射天线为 6，测试 1000000 次，并统计各个算法的无码率，测试结果如图 14.

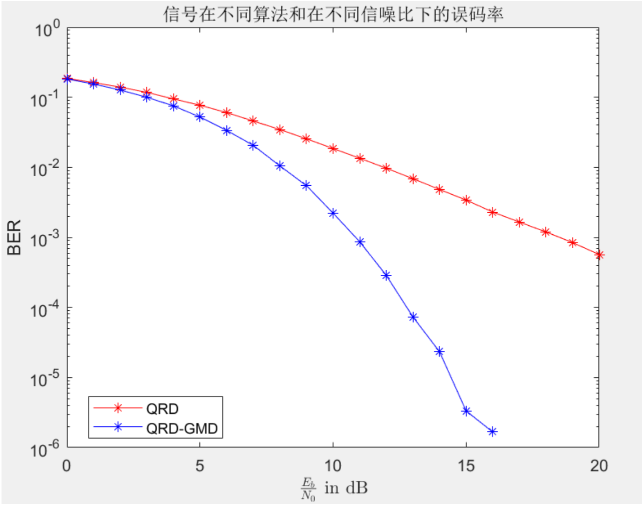

图 14: 测试结果 5

从图中可以看出，进行 GMD 分解后，大大降低了 QR 分解的误码率，提高了系统性能。

5 源代码附件说明

5.1 测试说明

直接运行 main.m 文件即可，为了测试方便，已经将测试次数改小。

5.2 源码说明

1. bertest* 文件，用来生成随机 H 和 x 矩阵。
2. compare* 文件，用来调用测试文件和算法文件来进行测试并绘图
3. 以算法命名的文件，每个文件代表一个算法。
4. main.m 文件，主测试文件，直接运行即可测试。