通信原理 实验报告

学号：2206113602
班级：信息 005
姓名：王靳朝

3 接收端的同步处理

一 实验内容（10 分）

1.1 接收端的同步处理

包括三方面内容：帧组同步、频偏补偿、相偏补偿。

二 实验原理（40 分）

2.1 接收端为什么需要同步？

同步传输是以同定的时钟节拍来发送数据信号的，因此在一个串行的数据流中，各信号码元之间的相对位置都是固定的，接收方为了从收到的数据流中正确地区分出一个个信号码元，首先必须建立准确的时钟信号。如果没有进行同步，将不能将单个符号从符号流中正确的分离出来。

2.2 频偏估计和补偿的原理

IQ 调制后的发送信号为

$$S(t)=I(t)\cos(\omega_c t)+Q(t)\sin(\omega_c t) \xrightarrow{\text{信道}} y(t)$$

预期接收：

$$I_r=y(t)\cos(\omega_c t)=[I_t\cos(\omega_c t)-Q_t\sin(\omega_c t)]\cos(\omega_c t) \xrightarrow{LPF} I_r=\frac{1}{2}I_t$$

$$Q_r=y(t)(-\sin(\omega_c t))=[I_t\cos(\omega_c t)-Q_t\sin(\omega_c t)][-\sin(\omega_c t)] \xrightarrow{LPF} Q_r=\frac{1}{2}Q_t$$

但由于存在 $\omega_t+\omega_c$ 及 $\Delta\varphi$ 的现象，实际接收信号为

$$\begin{cases} I_r=y(t)\cos\omega_r t = \frac{1}{2}I_t\cos(\Delta\omega t+\Delta\varphi)-\frac{1}{2}Q_t\sin(\Delta\omega t+\Delta\varphi) \\ Q_r=y(t)(-\sin\omega_r t)=\frac{1}{2}I_t\sin(\Delta\omega t+\Delta\varphi)+\frac{1}{2}Q_t\cos(\Delta\omega t+\Delta\varphi) \end{cases}$$

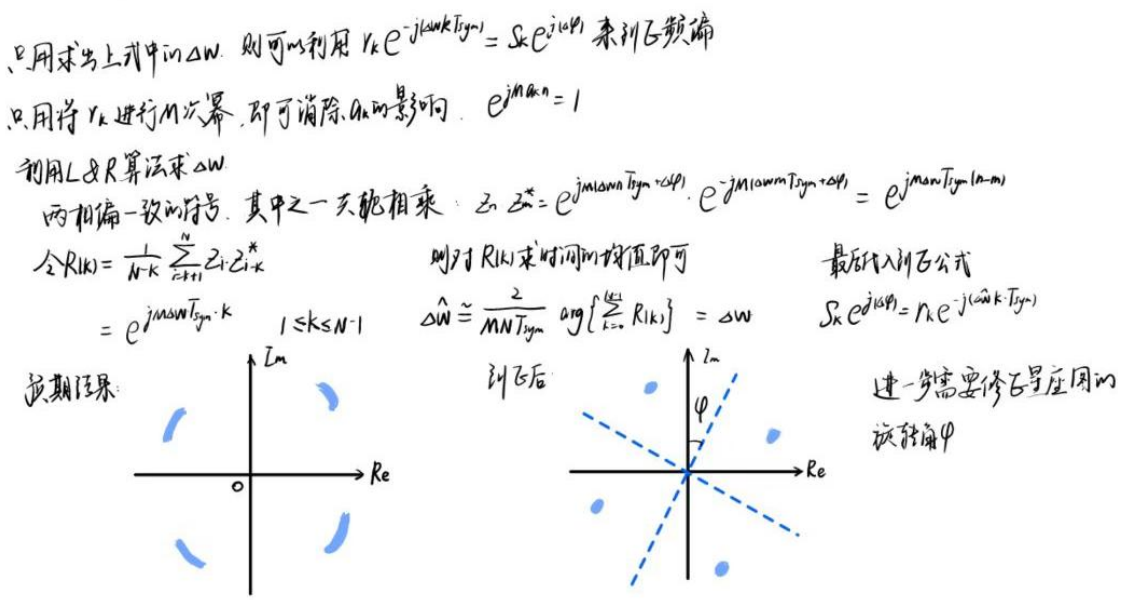
设符号相角集为 $\{a_k\}$，a_k 由调制的调制方式决定，例如：4PSK 时 $\{a_k\}=\{\pm\frac{\pi}{4},\pm\frac{3\pi}{4}\}$

定义 M 为符号集中符号的数目，例如：4PSK：$M=4$　BPSK：$M=2$，定义 T_{sym} 为系统时钟

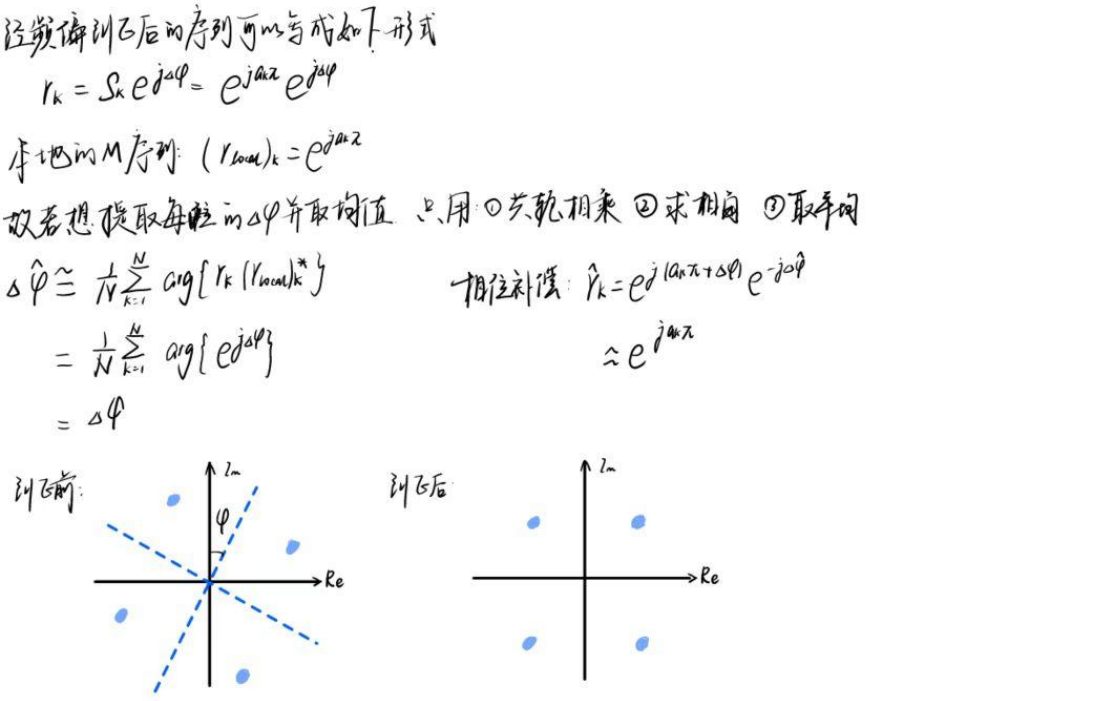
由 $t=kT_{sym}$ 代入，接收信号可以写为

$$r_k=I_r+jQ_r=e^{ja_k}e^{j(\Delta\omega kT_{sym}+\Delta\varphi)} \quad 1\le k\le N$$

N 为训练序列长度



2.3 相偏估计和补偿的原理



2.4 帧组同步的原理

帧组同步的目的是找到一帧的开头，所以此步骤需要在频偏补偿和相偏补偿之前进行。本实验中使用的是 127 位 M 序列作为训练序列，由于 M 序列有高度自相关和较低互相关

特性，可以用接收端本地的已知 M 序列在接收到的序列上滑动并计算相关值，相关值高于设定的阈值即可认为找到了 M 序列。

由于序列较长，求多次互相关可能会拖慢运算速度，故可以将整个帧组同步过程分为两步：粗同步和精同步。粗同步是起始点逐个取，但起始点之后的相关值数据点每隔一定距离取一个点（本实验中该距离取的是 4），进行滑动和相关值计算，在找到高于阈值的起始点后，再将其与其之后的一些点（本实验中该窗口长度为 8）逐一求完整相关值，选出最高的那个，即可精确定位到 M 序列。

三 具体实现（15 分）

3.1 帧组同步的具体实现

帧组同步函数 rx_package_search 有以下重要参量：

```
function [out_signal,cor_abs,col,index_s] = rx_package_search(rxdata,local_sync,len_frame,ratio)

    down_sig=reshape(rxdata,ratio,[]);% 假设为4倍过采样
    [m,n]=size(down_sig); %取数组大小，m=4,n=6258
    cor_abs=zeros(m,n);   %构建相同大小的数组，存储相关值
    len_window=8;         %精通步窗长度
    threshold=0.3;        %门限阈值0.3
    l=0;
    flag=false;           %找到的标志初始化为false，高于阈值置为true表示成功同步
```

该函数的实现以 flag 为核心，先将粗同步第一列的数与 M 序列依次求相关，当找到相关值大于门限 threshold 的起始点后，将 flag 置为 true，并记录次数，超过 8 次（窗口长度）跳出循环，找到了 M 序列。

3.2 频偏估计和补偿的具体实现

核心代码为：

```
z_k=training_seq.^2;
r=zeros(1,N-1);
for k=1:N-1
    r(k)=mean(z_k(1+k:N).*conj(z_k(1:N-k)));
end

freq_offset=angle(sum(r))/(2*pi*N*T_sym);
out_signal=rxdata.*exp(-1i*2*pi*freq_offset*(1:len)*T_sym);

end
```

频偏估计和补偿的代码实现和原理中的公式基本一致，即先将训练序列求 M 次幂（在本实验中使用 BPSK 调制方式，故 M=2，也可以为 2 的整数倍），得到 z_k，然后两两分别共轭相乘并求平均值，得到 r（即原理中的 R），提取相角并归一化即可的到频偏 freq_offset

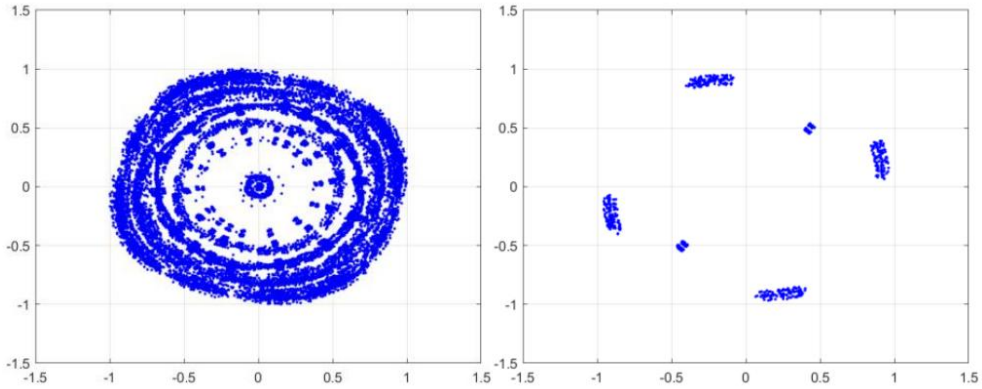
3.3 相偏估计和补偿的具体实现

函数输入：1*639 out_signal3，函数输出：1*639 out_signal4，偏移角 ang
算法描述：取 out_signal3 的前 127 位训练序列，和本地 local_sync 的共轭值相乘，得到 exp(j*theta) 求取 127 个角度值，取平均，作为偏移角 ang 对接收信号（1*639）的每一位进行相位补偿，即乘一个 exp(-j*theta)，具体代码如下：

```
function [out_signal1,ang] = rx_phase_sync(signal1_freq_sync,local_seq)
    len=length(local_seq);
    L=len;
    for i=1:L-1
        cor(i)=signal1_freq_sync(i).*conj(local_seq(L-i));
    end
    ang=angle(mean(cor))-pi;
    out_signal1=signal1_freq_sync.*exp(-1i*ang);
```

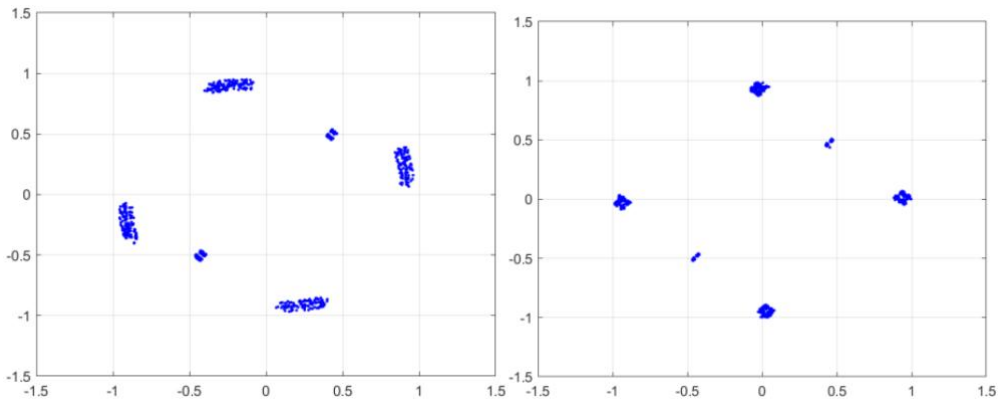
四 实验结果图示及分析（15 分）

4.1 帧组同步前后星座图分析



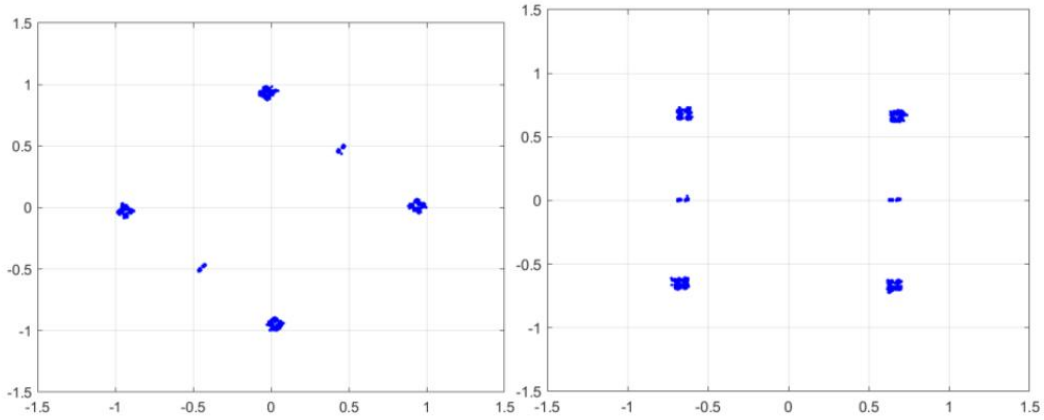
左侧是帧组同步前的星座图，右侧是帧组同步后，可以看到，系统提取出来了一帧的数据，相对原来来说基本收敛到一定区域内。

4.2 频偏估计前后星座图分析



左侧为频偏校正前，右侧为频偏校正后，可以看到拖尾基本消除，星座图从原来的四条线收敛成四个点。

4.3 相偏估计前后星座图分析



左侧为相位校正前，右侧为相位校正后，可以看到原本扭转的角度被消除了，星座图很接近与 BPSK 理论上的星座图，同步处理基本完成，可以进行解码操作了。

五 总结和思考（20 分）

5.1 实验过程中遇到的问题及解决方法（4 分）

实验中有许多求和以及求均值的步骤，直接写比较麻烦，可以使用 matlab 自带的函数 sum（）和 mean（），阅读时需要着重理解。

M 次幂的选择并无一定，本实验中只需是 2 的整数倍即可，但是次数选择过大会影响计算速度，因此合适即可。

同时需要注意选择门限值的大小，太大太小均会出现问题。

5.2 帧组同步一定要在频偏估计之前吗？为什么？（8 分）

不一定，因为和相偏估计不同，频偏估计并不需要使用本地的 M 序列，所以也就不需要将本地训练序列与接受训练序列对齐，从而不需要找到准确的 M 序列开始位置。但这可能造成用与纠正频偏的序列并不是 M 序列，而是接收序列中普通的一段，所以也就没有 M 序列的一些优良性质，比如说四个符号数量近似等，可能会造成频偏纠正效果变差。

但是一般情况下，帧同步应该在频偏估计之前进行。这是因为如果帧同步不准确，将导致数据帧的边界被错误地确定，进而影响频偏估计的准确性。

5.3 含频偏和相偏的 M 序列，自相关性会受影响吗？（8 分）

会有一定影响，因为在帧组同步的过程中使用的是理论上的序列和接收序列求相关，理论上的序列$r_{local} = e^{ja_k\pi}$，而接收到的序列$r_k = e^{ja_k\pi} \cdot e^{j(\Delta\omega_t+\Delta\varphi)}$，不完全一致，所以自相关函数和 M 序列理论上的自相关函数会略有差别。正因如此才需要合理调整门限 threshold，

使系统既可以在自相关达不到理论值时正确检测出来 M 序列，又不会将其他序列误判为 M 序列。