北京郵電大學

信息与通信工程学院

通信系统建模与仿真实验报告

实验题目：航天宏图卫星下行数据接收

姓名	班级	学号	联系电话
程梓睿	2022211114	2022210532	18814736488

实验日期：2025 年 6 月 11 日

目录

一、实验目的：

1. 掌握常见的数字调制技术如：QPSK

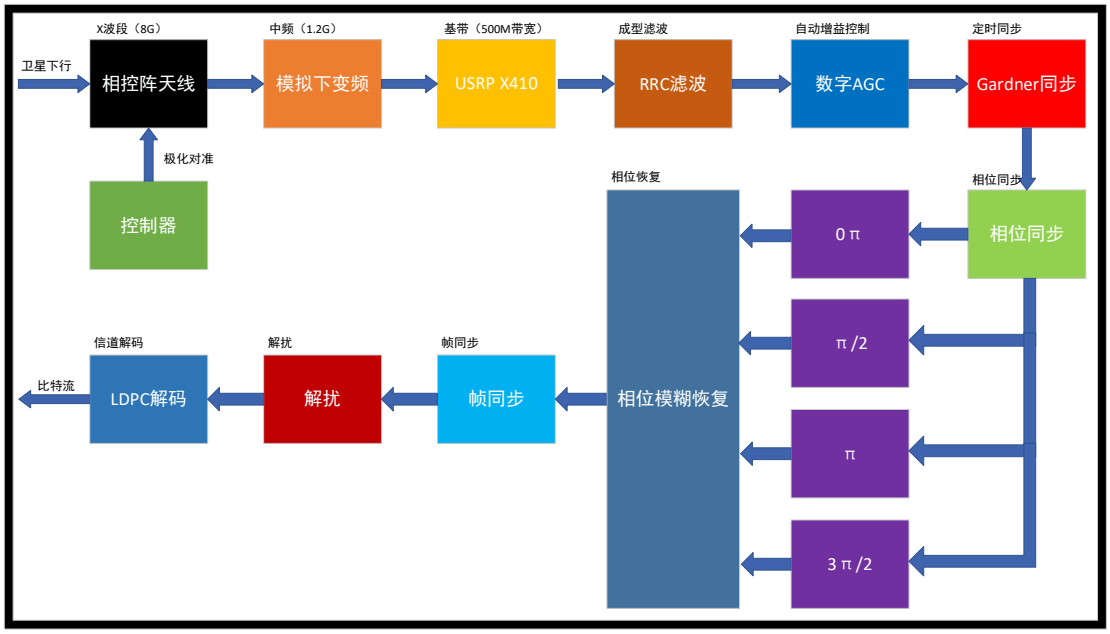
2. 了解卫星下行帧结构，掌握卫星信号接收流程。

3. 熟练使用 MATLAB 等软件，对实际信号进行接收与分析。

二、实验原理：

1. 卫星下行数传信号接收：

本实验所接收的卫星属于 SAR 遥感卫星，该类型卫星主动发射和接收微波信号来获取地表图像信息，具有全天时，全天候成像能力。本实验所使用的 SAR 卫星处于 X 波段（8-12G），其下行数传链路采用 QPSK 调制，帧结构符合 AOS 标准，本实验卫星下行信号接收框图如下：



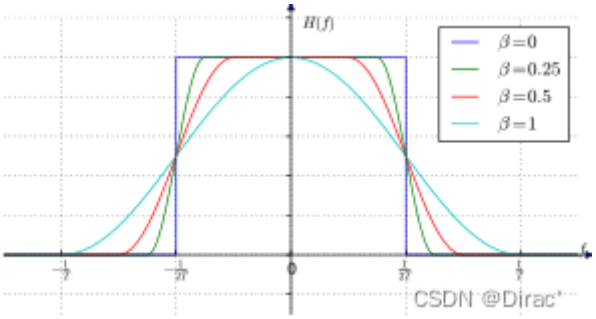
卫星接收框图

整体接收流程：

使用电子相控阵列天线，调整极化方式以及波束方向对卫星进行高精度跟踪，将含有多普勒的中频接收信号（1200G）通过 USRP X410 进行录制和接收，RRC 滤波和下变频得到 IQ 双路基带信号，使用 MATLAB 进行后处理。

2. RRC 滤波：

RRC（根升余弦滤波器）是低通奈奎斯特滤波器的一种实现，其具有对称性，常用于发送端的成型滤波，以及接收端的匹配滤波器。



其数学形式描述如下：

时域：

$$h(t)=\begin{cases}\dfrac{\pi}{4T_s}\sin c(\dfrac{1}{2\alpha}), & t=\pm\dfrac{T_s}{2\alpha}\\ \dfrac{1}{T_s}\sin c(\dfrac{t}{T_s})\dfrac{\cos(\frac{\pi\alpha t}{T_s})}{1-(\frac{2\alpha t}{T_s})^2}, & otherwise\end{cases}$$

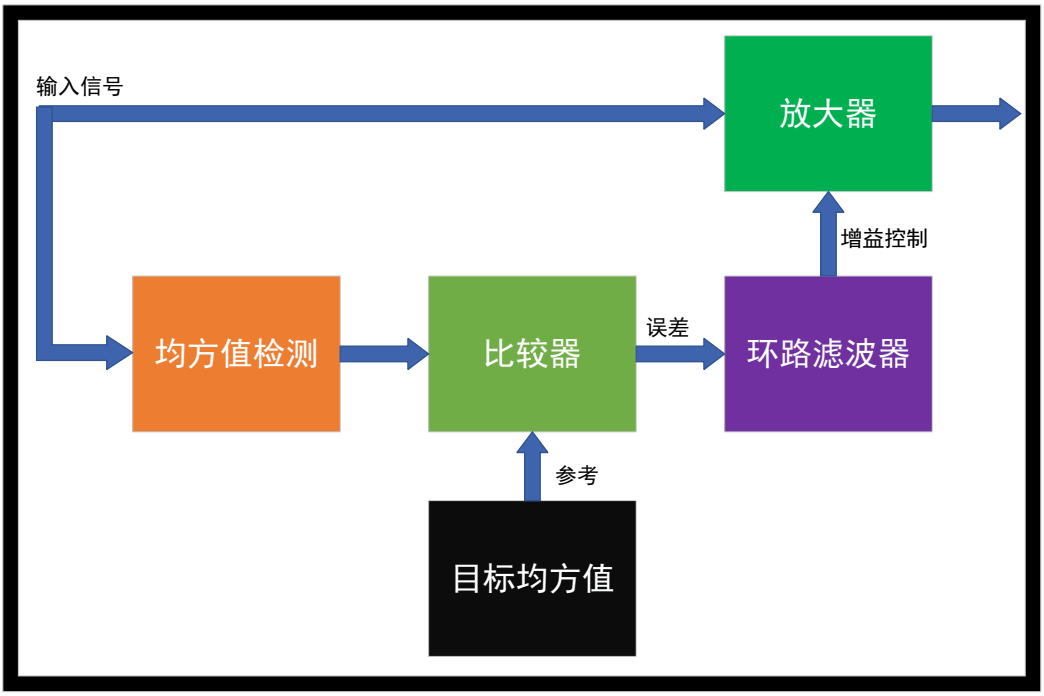
频域：

$$H(f)=\begin{cases}1, & |f|\le\dfrac{1-\alpha}{2T}\\ \dfrac{1}{2}\left[1+\cos\left(\dfrac{\pi T_s}{\alpha}\left[|f|-\dfrac{1-\alpha}{2T_s}\right]\right)\right], & \dfrac{1-\alpha}{2T}<|f|\le\dfrac{1+\alpha}{2T}\\ 0, & otherwise\end{cases}$$

通过改变滚降因子 α ，可以改变滤波器的过渡带宽，从而实现不同的滤波性能，在一般系统中， α 取 0.5 左右。

3.数字 AGC：

AGC（自动增益控制）是一种在接收机系统中广泛采用的信号处理算法，其核心目标是在输入信号的幅度变化较大的情况下，自动调整增益使得输出信号维持在相对稳定的幅度范围内，其算法框图可描述如下：



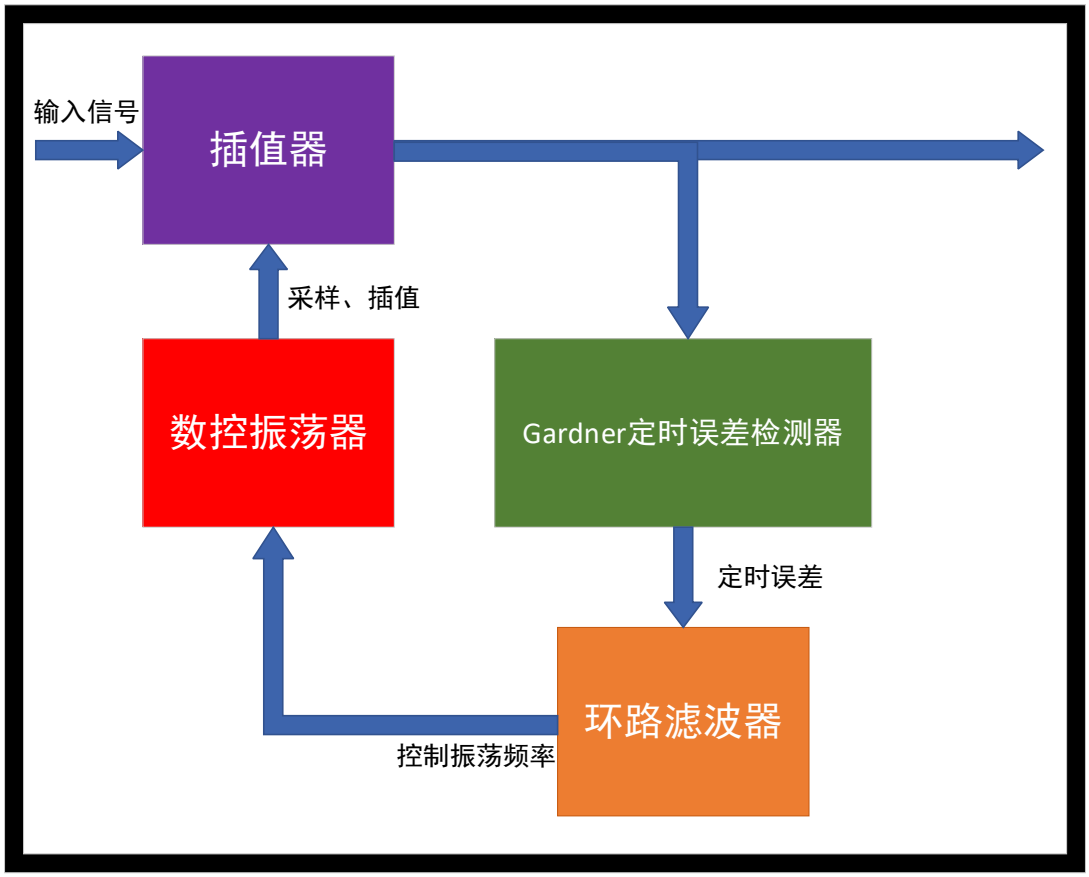
AGC算法框图

在 AGC 算法中，输入信号通过检测器，获得其功率信息（均方根），然后送入比较器，和目标功率水平对比，计算误差信号，通过环路滤波器后将控制信号输出到可变增益放大器，当环路收敛时，实现稳定的信号输出。在数字 AGC 中，提取功率通过求信号在一段窗口的 rms 实现，环路滤波可以通过二阶数字 PI 滤波器实现。

4. Gardner 定时同步：

由于接收端和发送端之间的时钟存在微小偏差（即符号率偏差），以及接收信号的初始相位具有随机性，接收系统必须对接收信号进行定时同步，以确保在正确的采样时刻提取符号信息，本实验中，利用 Gardner 环路算法对接收到的信号进行同步，Gardner 环路算法不依赖于载波相位，不需要数据辅助，且可以工作于两倍过采样下（sps=2），非常适合对卫星下行信号（QPSK）的定时同步。

Gardner 算法框图如下：



Gardner定时同步算法框图

插值器：

Gardner 环路最常用的插值器为 Farrow 插值器，它利用一组多项式进行插值运算，其实现相当于一个 FIR 滤波器，因此在数字系统中可以复用 FIR 滤波器结构加速运算，它具有较强的可扩展性，可以使用不同阶数获得不同的性能。本实验使用三阶 Farrow 插值结构，具有折中的性能，兼具精度和速度优势。插值函数描述如下：

$$f_1 = 0.5x(m) - 0.5x(m-1) - 0.5x(m-2) + 0.5x(m-3)$$
$$f_2 = 1.5x(m-1) - 0.5x(m) - 0.5x(m-2) - 0.5x(m-3)$$
$$f_3 = x(m-2)$$
$$y_1(k) = f_1 u(k) + f_2 u(k) + f_3$$

上式中，m-2 为插值的中心点，f1-f3 为三个插值多项式，u(k)为分数间隔，及当前抽样时钟与采样时钟比值的小数部分。

当 sps 为整数，且抽样时钟同步时，u(k)=0，此时插值器直接输出原始值，不做插值运算。

Gardner 误差检测器：

Gardner 误差检测算法中，每个符号只需要使用两个采样点，一个是 strobe 点，即最佳采样点，一个是 midstrobe 点，即两个采样点之间的观察点。其误差公式可描述如下：

$$\tau(n) = y(n-\frac{1}{2})\cdot[y(n)-y(n-1)]$$

其中，$y(n)$ 是当前时刻最佳采样符号，$y(n-1)$ 是上一时刻的最佳采样符号，$y(n-\frac{1}{2})$ 是采样点中间时刻的内插值。

考虑几种情况，假设采样时刻相对于理想采样时刻滞后，令 y(n)<0，y(n-1)>0，则 $y(n-\frac{1}{2})$<0，此时输出误差为正，此时应降低抽样时钟；若抽样时刻相对于理想采样时刻提前，则 $y(n-\frac{1}{2})$>0，此时输出误差为负，应当降低抽样时钟频率，其他情况依次类推。

综上，Gardner 定时误差检测通过斜率信息辅助估计，能够准确检测得到定时误差量，以正确调节抽样时钟的频率。

环路滤波器：

Gardner 算法使用的环路滤波器为差分 PI 结构，其公式可描述如下：

$$w(k+1) = w(k) + C_1(e(k)-e(k-1)) + C_2(e(k))$$

其中，C_1 为差分项的权值，C_2 为比例项的权值，$e(k)$ 为估计的定时误差，每次滤波器对过去的误差信号进行处理，预测得到一个将来的控制信号 $w(k+1)$。

数控振荡器：

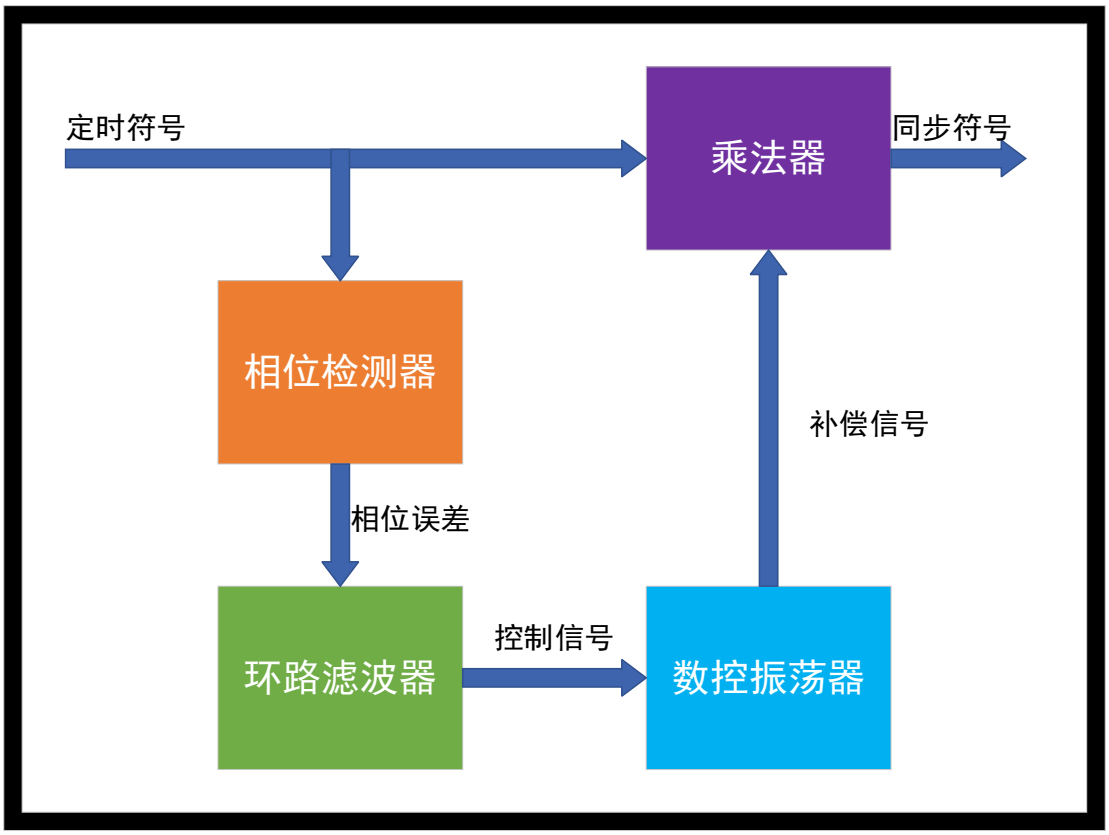
数控振荡器提供抽样时钟，对输入信号进行抽样，输出抽样后的判决符号，其中心频率为符号速率，并接受控制信号的大小来进行上下浮动，数学公式表达如下：

$$\eta(m+1) = [\eta(m)-w(m)]\bmod 1$$

该公式说明了数控振荡器可以描述为一个相位递减器，$w(m)$ 是环路滤波器输出的控制信号，对数控振荡器的相位递减步长进行调控。

5.载波同步：

可以利用普通锁相环完成信号的载波同步，当接收到一个 QPSK 符号时，假设信号存在多普勒或接收端与发射端的中心频率存在微小差异，会导致接收到的星座图随着时间发生旋转，会导致误码，此时需要利用锁相环，跟踪接收符号的相位旋转，对信号进行反向补偿。可以利用接收符号与标准星座点之间的相位差，作为锁相环的输入信号，将锁相环的输出接入数控振荡器，当存在相位差时，数控振荡器会输出差频信号，与原始信号相乘，实现对接收信号星座图的反向补偿，从而实现载波同步。框图如下图所示：



载波同步框图

其中，相位检测器负责将输入的符号与标准星座点对比，计算相位误差，这个误差信号用于后级星座补偿，公式表示如下：

$$e(n) = angle((I(n) + jQ(n)) * conj(I_{idea} + jQ_{idea}))$$

其中，I(n)和 Q(n)是当前接收到的 IQ 符号，I_{idea} 和 Q_{idea} 是距离这个符号最近的理想星座点，公式表示如下：

$$I_{idea} = \mathrm{Re}\{\frac{1}{\sqrt{2}}[(I(n)>0)+j(Q(n)>0)]\}$$
$$Q_{idea} = \mathrm{Im}\{\frac{1}{\sqrt{2}}[(I(n)>0)+j(Q(n)>0)]\}$$

计算得到当前星座点与理想星座点的相位误差后，将误差信号输入环路滤波器，这里为二阶 PI 滤波器，公式表示如下：

$$w(n) = K_p e(n) + K_i \sum_{i=0}^{n} e(n)$$

其中 e(n)为误差信号，K_p是 PI 滤波器的比例项系数，K_i是 PI 滤波器的积分项系数。

之后输出的 w(n)用于调整数控振荡器的振荡频率，从而实现载波同步。

6. 相位模糊恢复：

由于 QPSK 的星座图呈四点对称，因此同步得到的星座图存在$0,\frac{\pi}{2},\pi,\frac{3\pi}{2}$的相位旋转，需要进行相位恢复，否则接收的信号误码率大幅上升。在本实验中，实现相位模糊的恢复是通过 AOS 协议中的同步字实现的。在 AOS 协议中，信息以帧为单位传输，规定每个帧前的 4 个字节用于传输不加扰的同步字。因此，相位恢复的一种方法就是在信号本地构建同步字序列，并穷举四种相位旋转角，对输入信号进行主动相位补偿，数学表示如下：

$$x'(n) = x(n)\cdot e^{j\frac{2\pi k}{4}}, k = 0,1,2,3$$

然后提取同步字长度的相位补偿后序列，与本地的参考同步字做互相关运算，取四种旋转角中得到互相关谱最大值的 k 值作为估计的相位旋转，然后对整个帧应用相位旋转即可完成相位恢复。

但是，本实验中，由于 IQ 两路的同步字完全相同，因此相位恢复后的信号无法区分 IQ 两路的次序，将在后续解扰中进一步恢复。

7. 帧同步：

当完成对信号的相位模糊恢复后，用本地得到的参考同步字序列与接收到的信号进行互相关，寻找大于某个阈值（如：0.8）的互相关谱峰值的时域索引，作为估计的帧起始点。

8. 解扰：

当完成帧同步后，我们能够提取得到完整的 AOS 帧，长度为 1024 字节，其中由 4 字节同步字，6 字节 AOS 帧头，886 字节数据负载和 128 字节 LDPC 校验码组成，如下图所示：

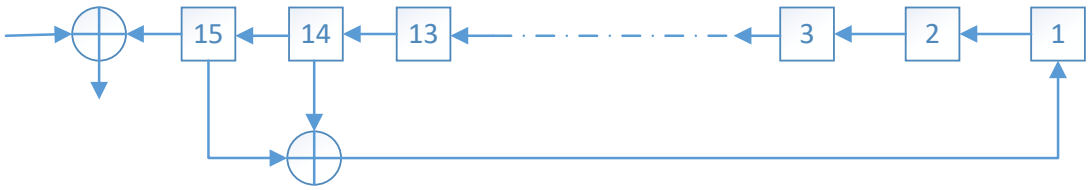
同步字（4字节）	AOS帧头（6字节）	数据负载（886字节）	LDPC校验码（128字节）

其中，除了同步字外的帧内容，在发送前都做了加扰运算，其目的是打乱比特流中的连续 0 或连续 1，消除直流分量以及抑制频谱的尖峰，使频谱白噪声化，从而提高时钟恢复性能，利于接收机的定时同步。

本实验中使用的加扰多项式如下：

$$1+X^{14}+X^{15}$$

对应的加扰模型：



其中，IQ 两路的加扰器使用不同的初相，如下：

$$\phi_I = 111111111111111$$

$$\phi_Q = 000000011111111$$

初相决定了解扰器中的 LSFR（线性反馈移位寄存器）的初始值，确定初始值后，LSFR 每次运算都会将输出的结果与输入的信号做异或运算，由于异或运算的可逆性，当解扰器与加扰器所使用的初相相同时，对加扰信号进行二次加扰即可实现解扰，恢复原始序列。

在相位恢复中，我们提到了同步字实现相位模糊恢复中存在无法区分 IQ 两路次序的问题，若 IQ 两路实现了交换，则由于 IQ 两路解扰器使用的初相不同，则解扰会将原始数据扰乱，无法进一步解码。

为了解决这一问题，可以通过帧内容进行验证，具体包括：

1. AOS 头有效性检查：

可以使用 AOS 中的一些已知位，如：版本号，卫星编号，传输速率标识等，若正确解扰，能够通过 AOS 头有效性校验。

2.LDPC 校验：

可以对数据进行 LDPC 软解调，若正确解扰，且帧数据正确，则 LDPC 软解调器可以在较短的迭代周期内收敛，否则，LDPC 无法收敛。

3.尾部检查：

由于本实验中使用的帧结构采用 LDPC 系统码编码，根据 CCCDS 手册关于卫星帧使用的 LDPC 编码规范，帧末尾的两个比特恒为 00，可以通过判断解扰后末尾两个比特是否为 00 来判断解扰的有效性。

本实验中，LDPC 软解调需要消耗大量的算力，AOS 帧头校验需要事先知道卫星的相关信息，因此使用方案三，通过检测尾部 00 比特来判断 IQ 两路是否发生交换，校正得到正确的 IQ 数据流。

三、实验思路：

1.观察频谱结构，分析信号参数：

已知卫星数传速率为 150Mbps，采用 QPSK 调制方式，并使用根升余弦滤波器进行成型滤波，我们需要确定 USRP 录制信号的采样率，带宽等参数。利用 MATLAB 观察录制信号的功率谱，利用无 ISI 传输条件和奈奎斯特准则，带宽应处于 75M-150M,采样率至少大于 150M（IQ 采样），通过对频谱测量以确定上述参数。

2.按照实验原理，搭建解调链路：

按照实验原理中对卫星信号的分析，编写相应 MATLAB 代码依次实现：RRC（匹配滤波器），AGC（自动增益控制），，Gardner 环路（定时同步），载波同步，相位恢复，帧同步，解扰，LDPC 解码模块。最后应当得到帧字节数组。

3.观察结果图，调整参数：

由于实际录制信号存在噪声，多普勒，本振偏移等情况，默认的参数可能无法实现稳定的信号解调，需要对环路进行参数调节，如：阻尼系数，环路带宽，直到观察到星座图稳定清晰，帧同步有序无差错为止。

四、实验过程：

1. 加载信号，观察频谱推断参数：

录制信号格式为 int16 存储的 IQ 双路信号，由于信号过大（50G），无法直接加载到 MATLAB 中，因此通过文件指针的方式，加载某一段数据（约 100 万点）进行分析，代码如下：

从文件中加载数据：（SignalLoader.m）

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SignalLoader
function y = SignalLoader(filename,pointStart,Nread)

% 打开文件
fid = fopen(filename, 'rb');

% 设置搜索指针
fseek(fid, (pointStart - 1) * 8, 'bof');

% 读取数据
raw = fread(fid, [2, Nread], 'int16');% float32 int16
y = complex(raw(1,:), raw(2,:));

%关闭指针
fclose(fid);

end
```

绘制频谱并观察：（SatelliteQPSKShow.m）

```
% 参数
filename = "sample_06111";
fs = 150e6;
startPoints = 200e6;
symbolLength = 1e6;

s_qpsk = SignalLoader(filename,startPoints,symbolLength);

% 绘制频谱
[Pxx, f] = pwelch(s_qpsk,[],[],[],fs,'centered');
subplot(1,1,1);
plot(f/1e6, 10*log10(Pxx));
xlabel('频率 (MHz)');
ylabel('功率谱密度 (dB/Hz)');
title('卫星滤波后QPSK调制信号 - 频谱');
grid on;
```

代码解释：

利用 fopen 打开文件，获得文件指针，然后在指定的数据点索引处以 int16 的格式读取 IQ 信号，并打包成复数形式输出。之后对提取的信号利用 pwelch 函数计算功率谱并绘图，测量其带宽以推断采样率等参数。

2.确定参数后，按照参数进行解调：

通过分析频谱图，得到了录制信号的参数（采样率 500M，RRC 成型滤波，带宽 100M），根据此参数进行解调。

1.重采样：

由于录制信号采样率为 500M，不满足 sps（每符号采样数）为整数的条件，需要对信号进行重采样。考虑到计算性能，将原始信号重采样到 150M，此时 sps=2，满足 Gardner 环路同步的最低条件，同时尽可能节约算力，提高运行速度，代码如下：

```
%% 执行重采样
s_qpsk = resample(s_qpsk,resampleMolecule,resampleDenominator);
```

代码解释：

利用 MATLAB 自带的 resample 函数进行重采样，它接受三个参数，分别是输入信号，上采样因子和下采样因子，本实验中，分别为 3 和 10，即将

500M 采样率的输入信号重采样到 150M，便于进一步处理。

2.RRC 滤波：

对录制信号应用 RRC 滤波，与信号所使用的 RRC 成型滤波匹配，等效实现一个完整的 RC 滤波，从而有效对信号进行限带，代码如下：

函数调用：

```
%% 执行RRC滤波
s_qpsk = RRCFilterFixedLen(fb/2,fs,s_qpsk,0.5,"RRC");
```

函数实现：（RRCFilterFixedLen.m）

```
function y = RRCFilterFixedLen(fb, fs, x, alpha, mode)

% 参数
span = 8; % 滤波器长度（单位符号数）
sps = floor(fs / fb); % 每符号采样数

% 生成滤波器
if strcmpi(mode, 'rrc')
    % Root Raised Cosine
    h = rcosdesign(alpha, span, sps, 'sqrt');
elseif strcmpi(mode, 'rc')
    % Raised Cosine
    h = rcosdesign(alpha, span, sps, 'normal');
else
    error('Unsupported mode. Use ''rrc'' or ''rc''.');
end

% 卷积，保持输入输出长度一致
y = conv(x, h, 'same');
end
```

代码解释：

基于 MATLAB 工具箱中的 rcosdesign 设计升余弦滤波器，根据参数 RRC 和 RC 选择根升余弦还是升余弦模式，滤波器长度为 8，这是一个较为通用的参数，将输入信号与设计得到的滤波器进行等长度卷积，保证输入输出长度一致。

函数接收五个参数，即：符号速率（比特速率一半），待滤波信号，滚降因子，滤波器模式。

3.AGC：

由于卫星在时刻运动，卫星在运动到不同角度时，天线接收到的功率不同，因此需要使用 AGC 对信号进行功率补偿，代码如下：

函数调用：

```
%% 执行AGC
s_qpsk = AGC_Normalize(s_qpsk,1,0.01);
```

函数实现：（AGC_Normalize.m）

```
function y = AGC_Normalize(x, target_power, agc_step)
% 初始化增益
gain = 1.0;

% 预分配输出
y = zeros(size(x));

% 实时逐点更新AGC（模拟时序处理）
for n = 1:length(x)
    % 当前输入样本
    sample = x(n);

    % 当前功率
    current_power = abs(sample * gain)^2;

    % 误差
    error = target_power - current_power;

    % 更新增益
    gain = gain + agc_step * error * gain;

    % 防止增益爆炸
    if gain < 1e-6
       gain = 1e-6;
    elseif gain > 1e6
       gain = 1e6;
    end
    % 应用增益
    y(n) = gain * sample;
end
end
```

代码解释：

利用简单的逐点 AGC 计算方式，计算每个点的瞬时功率（幅度的平方），调控信号的功率使其逼近目标水平，接收三个参数：输入信号，目标功率水平以及调节增益步长。

4.Gardner 环路（定时同步）：

使用 Gardner 环路实现 QPSK 符号的定时同步，Gardner 环路原理在实验原理分析部分已给出，这里不在叙述，代码如下：

函数调用：

```
%% 执行定时同步
s_qpsk_sto_sync = GardnerSymbolSync(s_qpsk, sps, 0.0001, 0.707);
```

函数实现（主要部分）：

```
%% Gardner 同步主循环
for m = 6 : length(s_qpsk)-3
    % NCO相位累加 (每个输入样本前进 wFilterLast 的相位)
    % 当 ncoPhase 越过 0.5 时，产生一个中点或判决点采样
    ncoPhase_old = ncoPhase;
    ncoPhase = ncoPhase + wFilterLast;

    % 使用 while 循环处理
    while ncoPhase >= 0.5
        % --- 关键修复 1: 正确计算插值时刻 (mu) ---
        % 计算过冲点在当前采样区间的归一化位置
        mu = (0.5 - ncoPhase_old) / wFilterLast;
        base_idx = m - 1;

        % --- 使用Farrow 立方插值器 ---
        y_I_sample = FarrowCubicInterpolator(base_idx, real(s_qpsk), mu);
        y_Q_sample = FarrowCubicInterpolator(base_idx, imag(s_qpsk), mu);

        %disp(y_I_sample);

        if isStrobeSample
            % === 当前是判决点 (Strobe Point) ===

            % --- Gardner 误差计算 ---
            % 误差 = 中点采样 * (当前判决点 - 上一个判决点)
            timeErr = mid_I * (y_I_sample - y_last_I) + mid_Q * (y_Q_sample - y_last_Q);

            % 环路滤波器
            wFilter = wFilterLast + c1 * (timeErr - timeErrLast) + c2 * timeErr;

            % 存储状态用于下次计算
```

```
            % 存储状态用于下次计算
            timeErrLast = timeErr;
            y_last_I = y_I_sample;
            y_last_Q = y_Q_sample;

            % 将判决点采样存入结果数组
            y_I_Array(end+1) = y_I_sample;
            y_Q_Array(end+1) = y_Q_sample;

        else
            % === 当前是中点 (Midpoint) ===
            % 存储中点采样值，用于下一次的误差计算
            mid_I = y_I_sample;
            mid_Q = y_Q_sample;
        end

        % 更新环路滤波器输出 (每个判决点更新一次)
        if isStrobeSample
            wFilterLast = wFilter;
        end

        % 切换状态: 判决点 -> 中点, 中点 -> 判决点
        isStrobeSample = ~isStrobeSample;

        % NCO相位减去已处理的0.5个符号周期，并为下一次可能的触发更新“旧”相位
        ncoPhase_old = 0.5;
        ncoPhase = ncoPhase - 0.5;
    end
end
```

代码解释：

Gardner 环路算法核心为，以 2 倍 sps 速率对输入信号进行插值，得到理想判决点和中间判决点，利用理想判决点和中间判决点计算得到定时误差并送入环路滤波器，由环路滤波器输出控制信号驱动采样时钟频率进行频率微调。由于算法以 2 倍速率进行插值，输出的判决符号也要进行降采样，通过控制插值点输出频率实现。

函数接收四个参数：输入信号，每符号采样数，环路带宽和阻尼系数，其中环路带宽和阻尼系数用于调控定时环路的灵敏度以及同步速度。通常，环路带宽取 0.0001fs，阻尼系数取 0.707。

5. 载波同步：

基于传统锁相环实现，但输入相位误差信号为输入信号与最近理想

星座点之间的相位差，输出驱动信号驱动数控振荡器步长频率偏移，代码如下：

函数调用：

```
%% 执行载波同步
s_qpsk_cfo_sync = QPSKFrequencyCorrectPLL(s_qpsk_sto_sync,0,fs,ki,kp);
```

函数实现（QPSKFrequencyCorrectPLL.m）：

```
%% 全局变量
theta = 0;
theta_integral = 0;

y = zeros(1,length(x));
err = zeros(1,length(x));

%% 主循环
for m=1:length(x)
    % 应用初始相位到x
    x(m) = x(m) * exp(-1j*(theta));

    % 判断最近星座点
    desired_point = 2*(real(x(m)) > 0)-1 + (2*(imag(x(m)) > 0)-1) * 1j;

    % 计算相位差
    angleErr = angle(x(m)*conj(desired_point));

    % 二阶环路滤波器
    theta_delta = kp * angleErr + ki * (theta_integral + angleErr);
    theta_integral = theta_integral + angleErr;

    % 累积相位误差
    theta = theta + theta_delta + 2 * pi * fc / fs;

    % 输出当前频偏纠正信号
    y(m) = x(m);
    err(m) = angleErr;
end
```

代码解释：

接收信号符合标准格雷码 QPSK 映射方式，通过检测接收符号所在象限即可确定最近理想星座点，计算相位误差后，环路滤波器输出控制信号到数控振荡器以校正频偏。

函数接受五个参数：输入信号，中心频率（基带为 0），采样率，积分项常数，比例项常数。

6.相位恢复+帧同步：

由于相位模糊基于同步字相关性检测实现，因此在估计相位模糊的时候就实现了帧同步，故将二者融合在一个函数中实现，相位恢复和帧同步的原理在实验原理分析部分已解释，故这里不再赘述，代码如下：

函数调用：

```
%% 执行帧同步，输出同步后的序列数组
sync_frame = FrameSync(s_qpsk_cfo_sync);
```

函数实现（FrameSync.m）：

```
%% 定义同步字
sync_bits_length = 32;
syncWord = uint8([0x1A, 0xCF, 0xFC, 0x1D]);
syncWord_bits = ByteArrayToBinarySourceArray(syncWord,"reverse");
ref_bits_I = syncWord_bits;
ref_bits_Q = syncWord_bits;
```

```
%% 开始处理主循环
for m=1:length(s_symbol)-frame_len
    % 提取I路和Q路信号
    s_frame = s_symbol(1,m:m+frame_len-1);

    % 处理相位模糊，依次旋转i路和q路
    for n=1:3
       s_frame = s_frame * (1i); %逆时针旋转90度
       %s_frame_bits = SymbolToIdeaSymbol(s_frame); % 先获取原始帧数据

       % 序列同步
       s_sync_frame = s_frame(1,1:sync_bits_length); % 提取帧数据中的同步字
       s_sync_frame_bits = SymbolToIdeaSymbol(s_sync_frame); % 提取同步字

       i_sync_frame_bits = real(s_sync_frame_bits);
       q_sync_frame_bits = imag(s_sync_frame_bits);

       % 检查是否相等
       if i_sync_frame_bits == ref_bits_I & q_sync_frame_bits == ref_bits_Q
           disp('序列匹配');
           disp('编号'+string(m));

           % 匹配才提取完整帧数据
           s_frame_bits = SymbolToIdeaSymbol(s_frame);
           sync_frame_bits = [sync_frame_bits;s_frame_bits];
           break;
       end

    end
end
```

代码解释：

本代码进行了一些优化，这是基于实际接收信号的效果而言的，由于接收场地处于开阔地带，且使用电子相控阵天线能够与卫星保持高度对齐，所以信道质量相当好，几乎不存在多径时延和噪声的干扰，实际测试发现相关性可达1，因此采用直接序列比对来代替相关性计算，以提高运算速度。通过对符号一次旋转判决，对判决后符号与本地同步字进行逐比特匹配，完全匹配对应的旋转因子即为估计的相位模糊。

函数接受一个参数，即同步符号。

7. 解扰：

按照数据帧加扰多项式实现对应的解扰结构，并通过检测末尾 2bits 来判断 IQ 是否反向，代码如下：

函数调用：

```
%% 执行解扰
[I_array,Q_array] = FrameScramblingModule(sync_frame);
```

函数实现（ScramblingModule.m）

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Scrambling/DeScrambling Module
function scrambled_data = ScramblingModule(data,InPhase)
%% 定义加扰解扰逻辑
N = length(data);
scrambled_data = zeros(1,N);
for m=1:N
    scrambled_data(m) = bitxor(InPhase(15),data(m));
    scrambled_feedback = bitxor(InPhase(15),InPhase(14));

    % 更新模拟移位寄存器
    for n=0:13
       InPhase(15-n) = InPhase(14-n);
    end

    InPhase(1) = scrambled_feedback;
end

end
```

函数实现（FrameScramblingModule.m）

```
for m=1:rows
    I_row_bits = I_bits(m,:);
    Q_row_bits = Q_bits(m,:);

    % 尝试解扰，考虑IQ未反向
    I_deScrambling = ScramblingModule(I_row_bits,InPhase_I);
    Q_deScrambling = ScramblingModule(Q_row_bits,InPhase_Q);

    % 检查是否合法
    if I_deScrambling(8159) == 0 && I_deScrambling(8160) == 0 && Q_deScrambling(8159) == 0 && Q_deScrambling(8160) == 0
        disp("序列合法");
        I_array(m,:) = I_deScrambling;
        Q_array(m,:) = Q_deScrambling;
    else
        % 假设未解扰成功
        % IQ两路交换，然后解扰
        I_deScrambling = ScramblingModule(I_row_bits,InPhase_Q);
        Q_deScrambling = ScramblingModule(Q_row_bits,InPhase_I);

        % 检查是否合法
        if I_deScrambling(8159) == 0 && I_deScrambling(8160) == 0 && Q_deScrambling(8159) == 0 && Q_deScrambling(8160) == 0
            disp("合法，但翻转");
            I_array(m,:) = Q_deScrambling;
            Q_array(m,:) = I_deScrambling;
        else
            % 维持原样输出
            I_array(m,:) = I_deScrambling;
            Q_array(m,:) = Q_deScrambling;

            disp("误码率过高");
        end
```

代码解释：

ScamblingModule 为自己封装实现的一个解扰模块，基于加扰多项式结构实现的一个 LSFR（线性反馈移位寄存器），通过初相初始化寄存器，将产生的伪随机序列与输入的加扰信号进行异或实现解扰。

FrameScramblingModule 中，提取判决 IQ 两路比特，依次进行正反解扰，通过检测末尾 2bits 判断是否存在 IQ 反向，若两次都无法解扰，则认为序列存在误码，输出警告。

ScramblingModule 函数接受两个参数，待解扰数据和寄存器初相；FrameScramblingModule 函数接受一个参数，即待解扰的复数比特。

8.LDPC 译码：

由于 MATLAB R2021a 未提供适配于 AOS 帧的 LDPC 译码函数，且自己实现的 LDPC 译码尚未做稀疏矩阵优化，实际运算时，需要处理 8000x7000 规模的矩阵运算，几乎无法在有限时间内完成运算。且考虑到信噪比较高，几乎不存在误码，因此并未实现 LDPC 译码，直接输出编码序列，对于 LDPC 系统码，其编码未改变信息位，只是在原始数据后附加了 128 字节的校验位，因此可以通过数据截断直接提取原始信息。

9. 交织输出到文件：

将解码得到的比特流输出到文件，采用三种输出方式：I 路单独输出，Q 路单独输出，IQ 交织输出，代码如下：

```
%% 提取到IQ字节
[rows,columns] = size(I_array);
I_bytes = [];
Q_bytes = [];
for m=1:rows
    I_bytes = [I_bytes,BinarySourceToByteArray(I_array(m,:))];
    Q_bytes = [Q_bytes,BinarySourceToByteArray(Q_array(m,:))];
end

%% 交织到文件
BytesStream = zeros(1,length(I_bytes)*2,'uint8');

for m=1:length(I_bytes)
   BytesStream(2*m-1) = I_bytes(m);
   BytesStream(2*m) = Q_bytes(m);
end

%% 数据验证，选取一帧打印出AOS
aosFrameHead = AOSFrameHeaderDecoder(I_array);

%% 写入到文件
WriteUint8ToFile(I_bytes,IBytesFilename);
WriteUint8ToFile(Q_bytes,QBytesFilename);
WriteUint8ToFile(BytesStream,IQBytesFilename);
```

代码解释：

输出到三个文件，分别为 Ibytes.txt，Qbytes.txt，IQbytes.txt。

10. 验证 AOS 帧头：

由于恢复卫星图像需要大量帧数据，因此采用验证 AOS 帧头的方式来验证数据有效性，基于 AOS 帧头格式编写的 AOS 帧头解码代码如下：

代码实现（AOSFrameHeaderDecoder.m）：

```
%% 核心解码逻辑
% 获取版本ID号
if join(string(aosFrameBits(1,1:2)),"") == "01"
    aosFrameHead.versionId = 1;
end

% 获取卫星组号
satelliteGroupId = aosFrameBits(1,3:10);
if (BinarySourceToByte(satelliteGroupId) >= 36) && (BinarySourceToByte(satelliteGroupId) <= 39)
    aosFrameHead.satelliteType = "02组";
elseif (BinarySourceToByte(satelliteGroupId) >= 40) && (BinarySourceToByte(satelliteGroupId) <= 43)
    aosFrameHead.satelliteType = "03组";
else
    aosFrameHead.satelliteType = "无效";
end

% 获取虚拟信道标识
satelliteVirtualChannelId = join(string(aosFrameBits(1,11:16)),"");
if satelliteVirtualChannelId == "000001" && aosFrameHead.satelliteType == "02组"
    aosFrameHead.satelliteVirtualChannelId = "02组 SAR数据";
elseif satelliteVirtualChannelId == "000010" && aosFrameHead.satelliteType == "02组"
    aosFrameHead.satelliteVirtualChannelId = "02组 星上处理数据";
elseif satelliteVirtualChannelId == "111111" && aosFrameHead.satelliteType == "02组"
    aosFrameHead.satelliteVirtualChannelId = "02组 填充帧";
elseif satelliteVirtualChannelId == "000000" && aosFrameHead.satelliteType == "03组"
    aosFrameHead.satelliteVirtualChannelId = "03组 有效数据";
elseif satelliteVirtualChannelId == "111111" && aosFrameHead.satelliteType == "03组"
    aosFrameHead.satelliteVirtualChannelId = "03组 填充帧";
else
    aosFrameHead.satelliteVirtualChannelId = "无效";
```

```
% 获取VCDU循环计数
satelliteVCDUCounter = BinarySourceToInt(aosFrameBits(1,17:40));
aosFrameHead.satelliteVCDUCounter = satelliteVCDUCounter;

% 标志域数据解码
% 获取实时回放标识
satelliteReplyId = join(string(aosFrameBits(1,41)),"");
if satelliteReplyId == "0"
    aosFrameHead.satelliteReplyId = "实时传输";
else
    aosFrameHead.satelliteReplyId = "回放";
end

% 获取下传标识
satelliteDownloadId = join(string(aosFrameBits(1,42)),"");
if satelliteDownloadId == "0"
    aosFrameHead.satelliteDownloadId = "单路下传";
else
    aosFrameHead.satelliteDownloadId = "双路下传";
end

% 获取IQ数据标识
satelliteIQDataId = join(string(aosFrameBits(1,43:44)),"");
if satelliteIQDataId == "00"
    aosFrameHead.satelliteIQDataId = "合路";
elseif satelliteIQDataId == "01"
    aosFrameHead.satelliteIQDataId = "I路";
elseif satelliteIQDataId == "10"
    aosFrameHead.satelliteIQDataId = "合路";
else
    aosFrameHead.satelliteIQDataId = "无效";
```

```
% 获取传输速率标识
satelliteDigitalSpeed = join(string(aosFrameBits(1,45:48)),"");
if satelliteDigitalSpeed == "0111"
    aosFrameHead.satelliteDigitalSpeed = "1500Mbps";
elseif satelliteDigitalSpeed == "1100"
    aosFrameHead.satelliteDigitalSpeed = "1200Mbps";
elseif satelliteDigitalSpeed == "1001"
    aosFrameHead.satelliteDigitalSpeed = "900Mbps";
elseif satelliteDigitalSpeed == "0110"
    aosFrameHead.satelliteDigitalSpeed = "600Mbps";
elseif satelliteDigitalSpeed == "0101"
    aosFrameHead.satelliteDigitalSpeed = "300Mbps";
elseif satelliteDigitalSpeed == "1010"
    aosFrameHead.satelliteDigitalSpeed = "150Mbps";
elseif satelliteDigitalSpeed == "0011"
    aosFrameHead.satelliteDigitalSpeed = "75Mbps";
else
    aosFrameHead.satelliteDigitalSpeed = "无效";

end
```

代码解释：

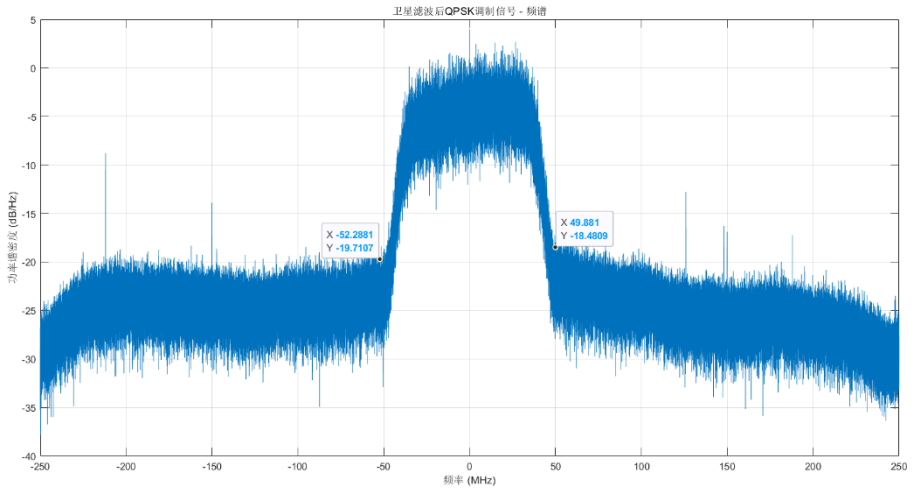
依据该帧头格式实现解码：

AOS 帧头格式

定义	版本号	VCDU 识别		VCDU 计数	标志域			
		航天器标识符	虚拟信道标识		实时回放标识	下传标识	IQ 数据标识	传输速率标识
Bit	2	8	6	24	1	1	2	4
内容	0b01	24H~27H：宏图二号 02 组（A-D 星） 28H~2BH：宏图二号 03 组（A-D 星）	02 组 0b000001:SAR 数据 0b000010:星上处理数据 0b111111 填充帧 03 组 0b000000 有效数据 0b111111 填充帧	循环计数	0b0:实传 0b1:回放	0b0：单路下传 0b1：双路下传	0b00：合路 0b01：I 路 0b10：Q 路	0b0111:1500Mbps 0b1100:1200Mbps 0b1001:900Mbps 0b0110:600Mbps 00b0101:300Mbps 0b1010:150Mbps 00b0011:75Mbps

五、实验结果及分析：

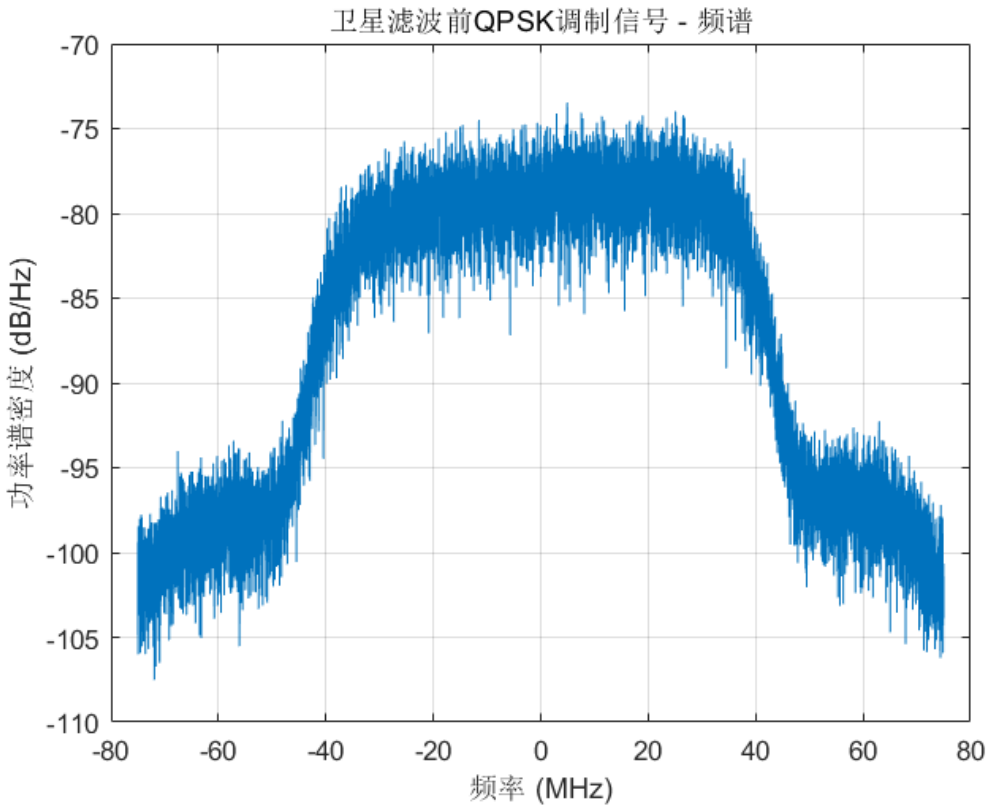
1. 信号参数判断：

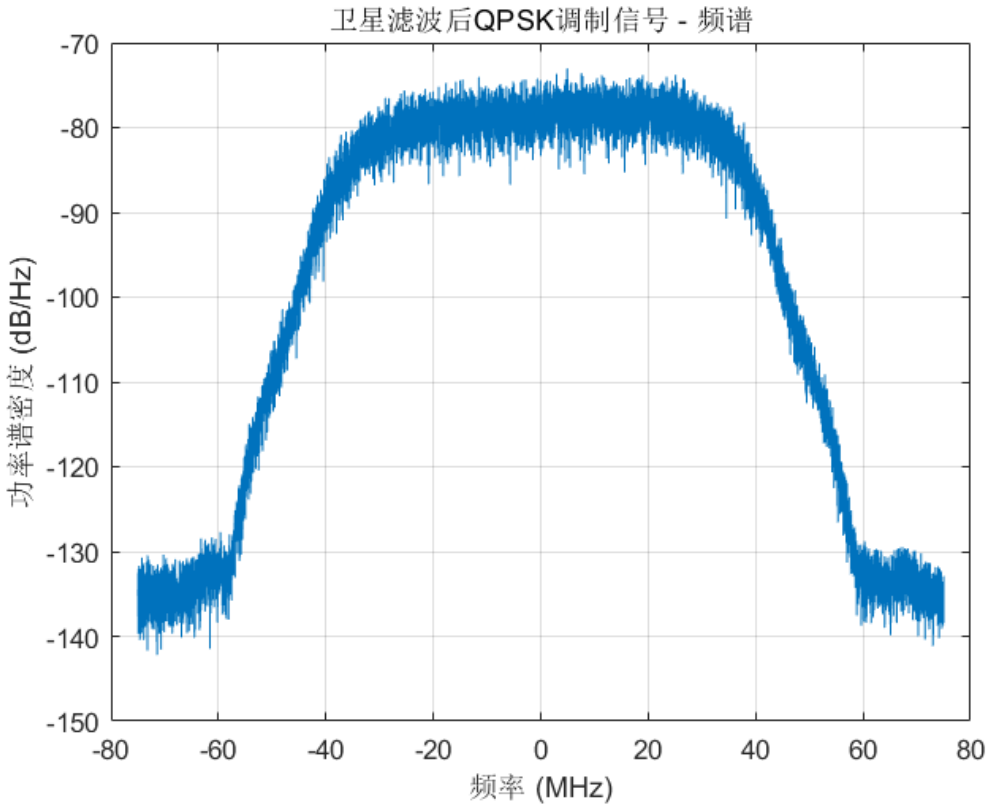


结果分析：

尝试采样率 500M，测量带宽为 100M，满足理论分析带宽范围（75M-150M），将其确定为信号参数。

2. 滤波前后频谱对比：

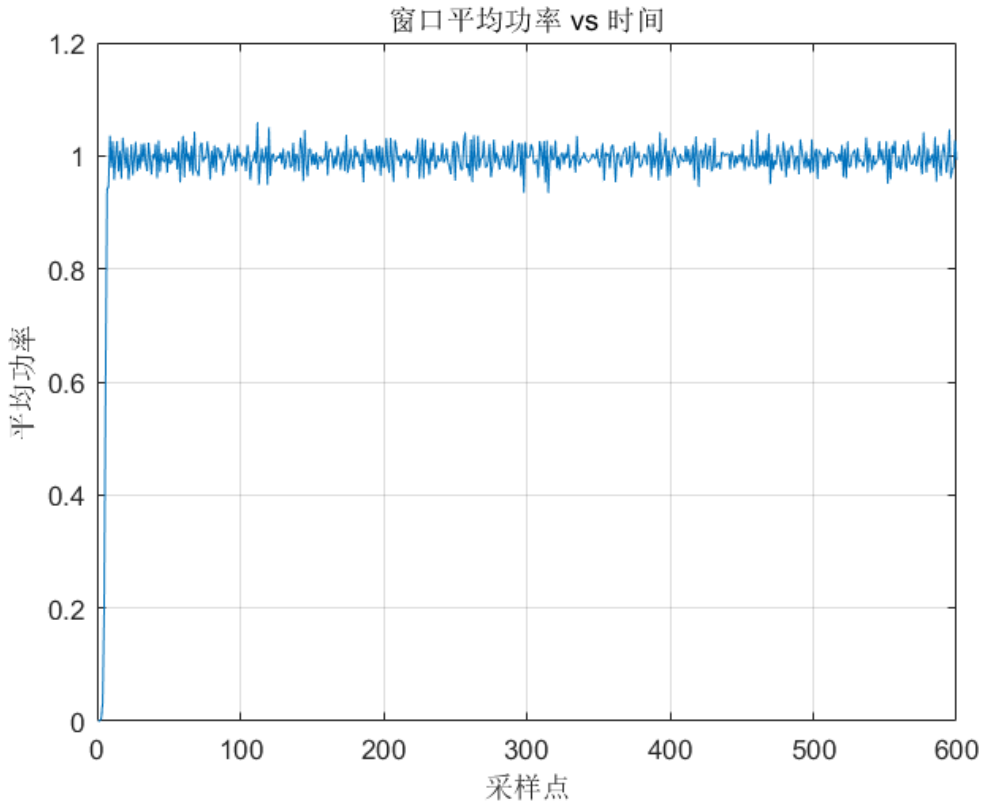




结果分析：

从图中可以看出，RRC 滤波后的频谱，其过渡带更短，边带抑制效果显著提高。

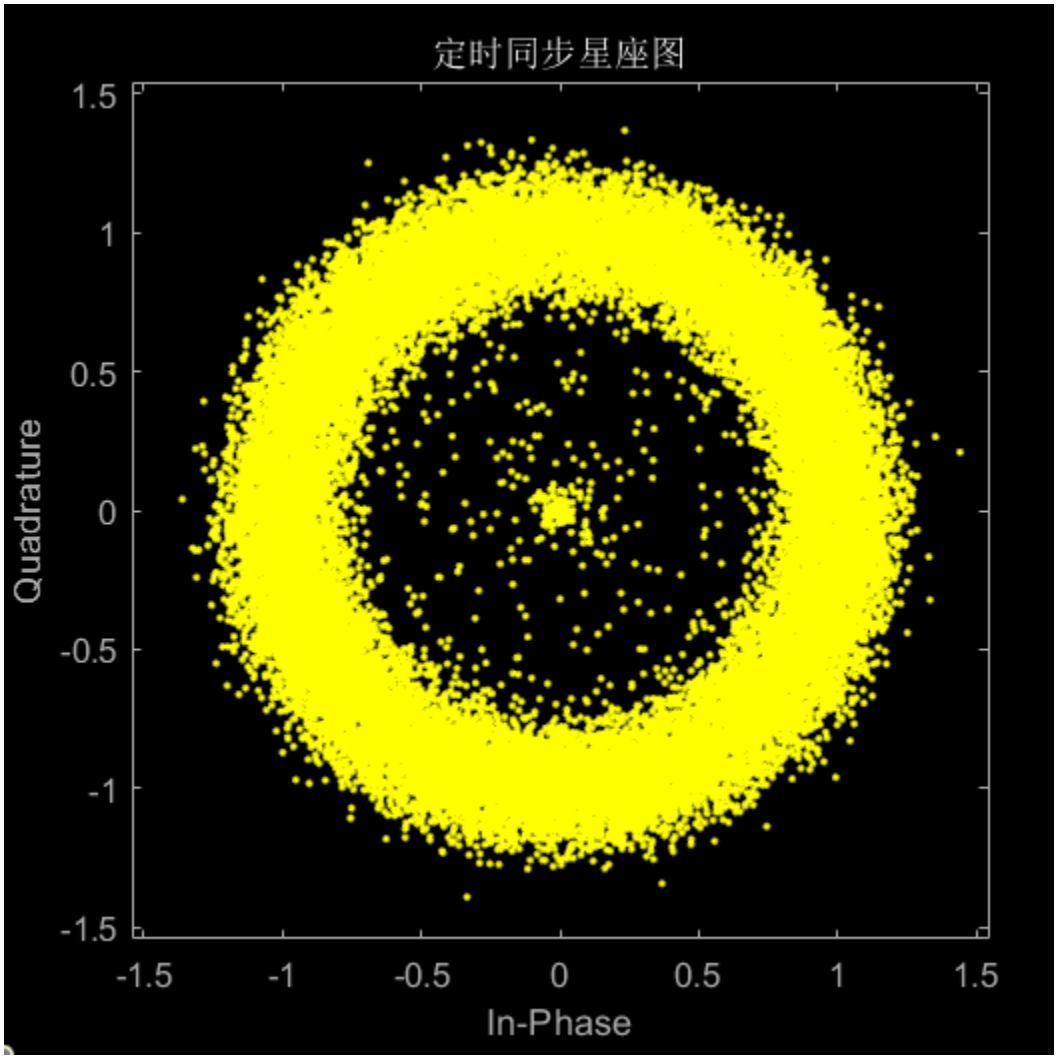
3. AGC 效果：



结果分析：

选择长度为 100 的窗口计算平均功率，可以看出信号功率基本稳定在 1 左右，体现 AGC 具有较好的功率控制效果。

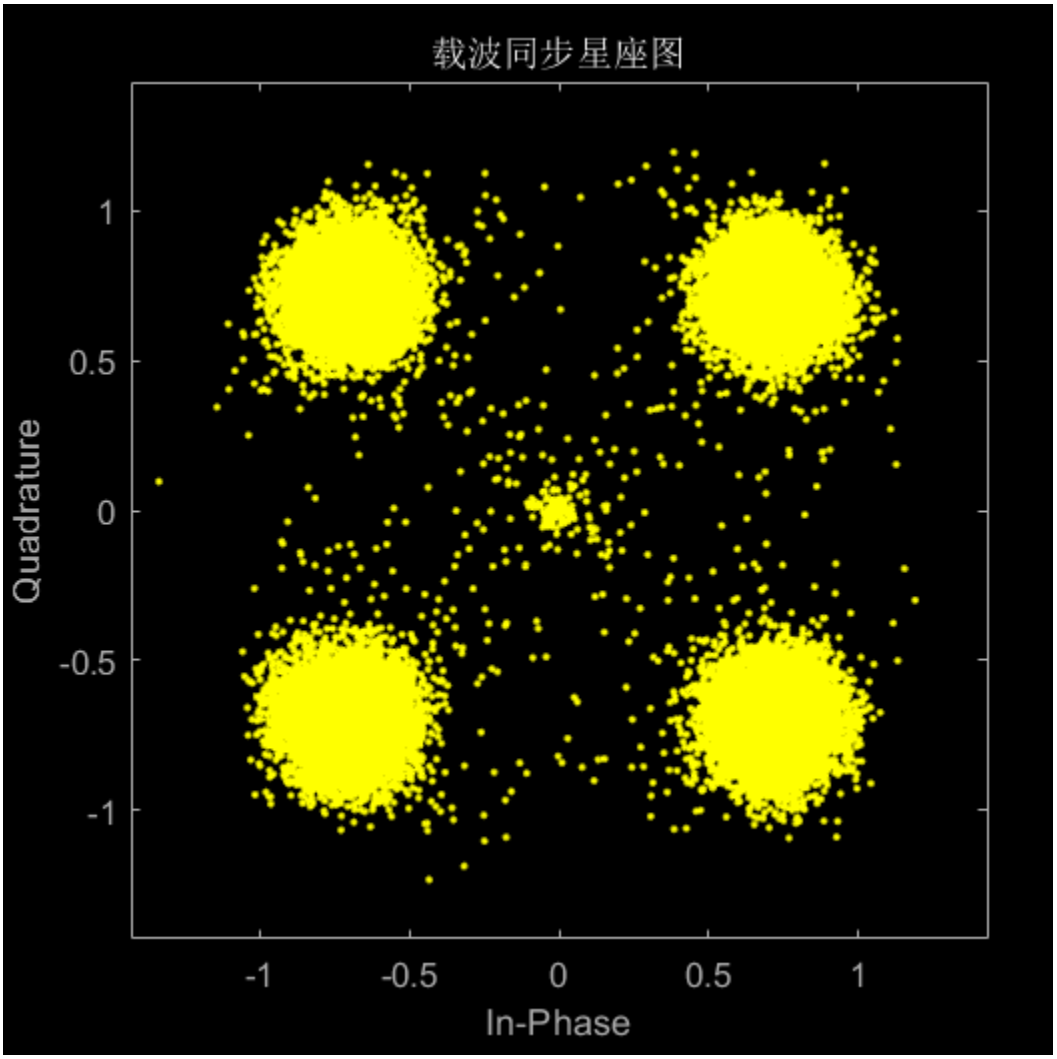
4. Gardner 环路同步效果：



结果分析：

从图中可以看出，星座图为标准圆环，体现了较好的定时同步效果，因为此时尚未进行载波同步，信号存在多普勒频偏以及本振漂移，星座图不是标准的四点形状。

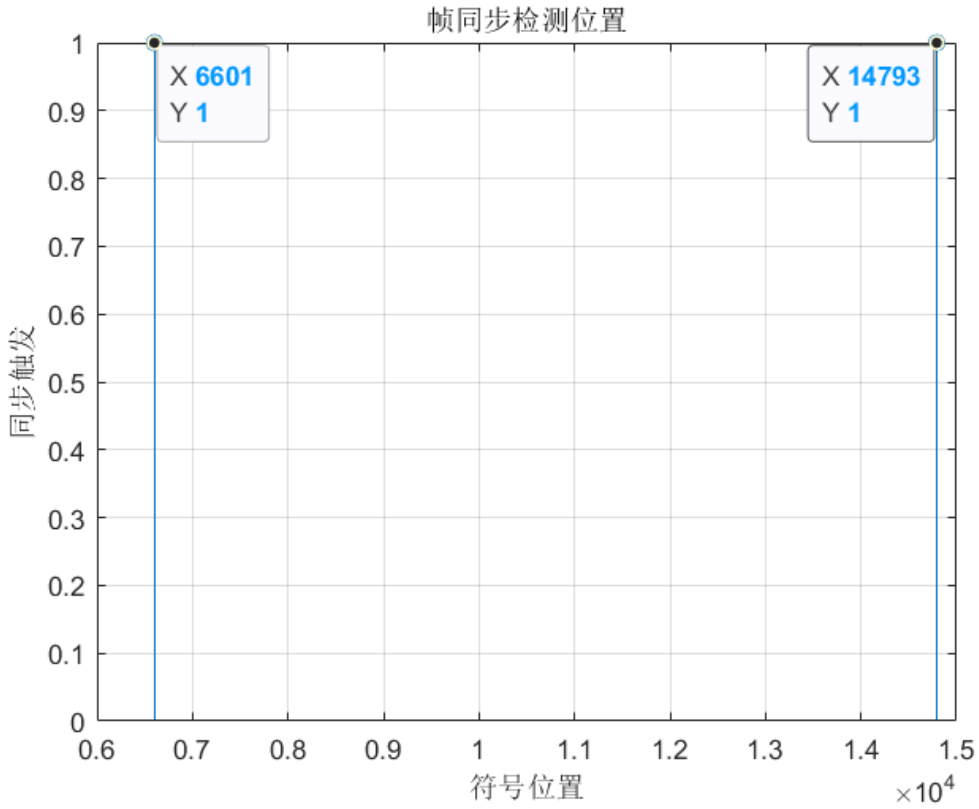
5. 载波同步效果：



结果分析：

从图中可以看出，利用锁相环消除多普勒偏移以及本振漂移后，星座图呈现标准的 QPSK 四点式，且彼此间隔较远，体现了较高的信噪比。由于星座图是绘制整个信号处理的结果，而锁相环同步需要时间，所以星座图会呈现出一些噪点，表示环路尚未同步时候的星座图。

6. 帧同步：



结果分析：

从图中可以看出，在处理的信号点数中，检测到了两个帧同步点，且距离 8192 符号，符合 AOS 帧长度（1024 字节），证明帧同步良好。

7. 解扰：

```
序列匹配
编号 6601
序列匹配
编号 14793
序列合法
序列合法
```

结果分析：

对帧同步点提取帧进行解扰，并成功检测到了解扰后数据的末尾 00 比特，控制台输出序列合法，表示解扰验证成功且未发生 IQ 翻转。

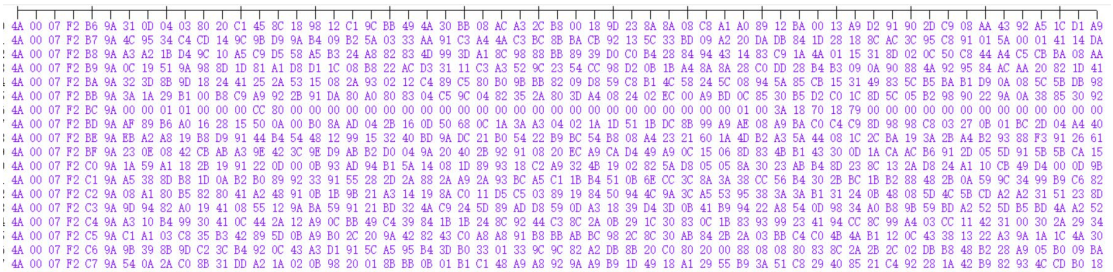
8. AOS 帧头验证：

```
                versionId: 1
            satelliteType: "03组"
satelliteVirtualChannelId: "03组 有效数据"
     satelliteVCDUCounter: 521619
         satelliteReplyId: "回放"
      satelliteDownloadId: "单路下传"
        satelliteIQDataId: "I路"
    satelliteDigitalSpeed: "150Mbps"
```

结果分析：

对第一帧的 AOS 帧头进行打印，与实际信号 AOS 帧头结果匹配（版本号 01，I 路，单路下传，150Mbps 速率）。

9.证明（数据帧解调截图）



六、遇到的问题：

1.卫星信号含多普勒，干扰锁相环锁定：

答：

卫星处于不断运动中，由于卫星运动速率较高，根据多普勒计算公式：

$$f_m = \frac{v}{\lambda}\cos\theta$$

其多普勒频移较高，甚至可以达到 200khz，且处于时刻变化中，需要调节锁相环进行锁定。

分析得到以下解决方案：

信号采样率相当高，重采样后 150M 采样率下，200khz 的频偏，相对频偏并不大，且卫星虽然运动速度快，但是角速度变化很慢，锁相环有充足的时间锁定卫星的动态多普勒频偏，调节锁相环参数，成功对含多普勒频偏的卫星信号完成锁定。