遥感卫星SAR图像接收与解调实验

一、引言

2025 年 6 月 11 日，北京邮电大学迎来了建校 70 周年的重要纪念日。为庆祝这一历史性时刻，学校组织了一系列富有科技感和纪念意义的活动。其中，在校本部西土城校区操场，由学生志愿者在地面通过反射角编码布置出“70”字样，构成可被卫星识别的地面图案。与此同时，株洲太空星际公司配合学校活动，安排其在轨遥感卫星对该区域进行合成孔径雷达（Synthetic Aperture Radar, SAR）成像，并通过卫星下行链路将原始成像数据实时传输至地面。

SAR 技术是一种主动式遥感成像手段，其通过向地面发射微波信号并接收其反射波，以构建高分辨率二维图像，具有全天时、全天候成像能力。在城市、地表监测、灾害响应、军事侦察等领域均有广泛应用。相较于光学遥感，SAR 不受天气、云层和光照影响，因此成为当前深空遥感和地球观测的重要手段之一。

本次实验的主要目标是通过地面接收设备捕获来自遥感卫星的 SAR 下行数据，并利用软件无线电平台对其进行信号解调、帧同步和图像重建，最终还原出包含“70”字样的图像内容，以验证图案识别的可行性和接收解调技术的准确性。实验采用了包括 SDR（Software Defined Radio）技术、调制识别、数据解析、图像重构等多个关键步骤，具有高度的工程实践和研究价值。

通过此次实验，我不仅实现了对遥感成像信号的接收和处理，也收获了深度理解卫星通信、SAR 成像与无线信号解调的宝贵实践经验。实验成果也为日后开展低成本星地通信验证、地面图案识别及数据链路处理奠定了实际基础。

二、实验环境与技术方案

2.1 实验时间与地点

本次实验于 2025 年 6 月 11 日 10 时至 12 时在北京邮电大学西土城校区操场区域开展。

根据卫星过境时刻预报，株洲太空星际公司的遥感卫星在当日 10:55 至 11:10 经过北京邮电大学西土城校区操场上空，星下点轨迹覆盖我校区域，具备较好的信号接收条件。

2.2 卫星与任务概况

所使用的遥感卫星为株洲太空星际公司自主研发的一颗低轨 SAR 成像卫星。实验当日，卫星对校区进行 1 分钟左右的连续成像，并通过 X 波段下行链路将数据实时回传至地面。

下行调制方式为 QPSK，数传左旋，数据速率为 150 Mbps。卫星同时向公网发布过境时刻、轨道数据和任务参数，便于地面站配合进行数据接收。

2.3 任务分工与架构组成

实验过程中采用了软件无线电平台接收信号。核心设备组成如下：

- **信号接收系统**：主要由航天宏图技术人员及课程指导老师负责，采用相控阵天线，基于 Ubuntu 系统进行实时接收处理。
- **信号处理系统**：主要由我们负责，待接收完毕后拷贝部分原始信号数据至笔记本电脑上，并通过自己设计的解调链路进行信息处理。

2.4 解调与处理方案

数据解调流程结合 MATLAB + Simulink 编程，整体技术架构分为如下几个阶段：

1. **信号读取与参数设置**：通过 MATLAB 从本地文件中读取指定大小的信号点，存储至工作区中；同时设置信号与解调过程的基本参数（如采样率、符号速率、滤波器参数等），用于后续处理。
2. **频谱分析与载波锁定**：通过 Simulink 实现对信号的上/下采样、滤波、符号同步、载波同步过程，最后导出高信噪比、可直接判决的 QPSK 信号至工作区中。
3. **帧同步与解调**：通过 MATLAB 实现 QPSK 软解调模块，结合已知卫星同步字节结构进行帧头定位、解扰、 LDPC 解码、AOS 帧头解析等。

2.5 技术原理简述

SAR 成像的基本原理为利用多普勒效应进行合成孔径扩展，以提高分辨率。成像过程中，雷达波束沿轨道方向发射并接收目标反射波，通过相干处理实现二维图像合成。QPSK 调制将图像数据调制为相位信号载体，通过地面接收并解调后还原出原始图像序列。由于从 SAR 回波中解调出具体的图像信息需要经过复杂算法处理，且并未在相关资料中提供，因此本次解调任务仅以得到 AOS 帧头信息为最终结果。

三、实验过程

3.1 卫星过境准备与信号接收

在实验前一日，株洲太空星际公司及本课程指导老师制定了实验当日任务时间表，如下图所示：

0611北邮”70“图样拍摄接收

日期	时间	内容	责任人	
20250609	14:20-16:30	终端设备到达北邮	陈皓	
		确认终端摆放位置、设备主辅件	陈皓/韩笑宇	
	20:00前	第二天拍摄计划确认	周亚文	
20250610	10:00-11:00	卫星数据接收	陈皓/毕凌宇	
		天线信号采集/录制	韩笑宇	尹老师/刘老师
	14:20-17:30	确认金属面摆放位置及方位	闫宁/等	孙老师
		确认金属面对应的学生安排	周亚文	孙老师
20250611	10:30前	铝箔折叠、学生职责明确	周亚文/闫宁	
		现场摄像、拍照人员2人到位情况		刘老师
	10:30-11:00	移动终端设备部署及操作	陈皓/毕凌宇	
		操场人员管理、时间同步	闫宁	孙老师
		天线接收模拟信号录制	韩笑宇	尹老师/刘老师
	11:00-11:30	操场人员收拾设备、整理收纳	闫宁	孙老师
		移动终端现场数据解析	韩笑宇	
	11:30后	各位同学开展信号解调等工作		尹老师/刘老师
		大站数据接收及解析出图（如有）	周亚文	

在此期间，我们作为信号处理小组成员全程参与设备部署与信道确认，并在笔记本电脑中预设数据处理接口，确保接收完成后可以立即获取并拷贝原始信号数据。

卫星开始过境后，地面相控阵天线自动跟踪其轨迹并稳定接收 X 波段数传信号。在信号接收结束后，技术人员将部分原始 IQ 数据包（采样率为 500 MHz，存储格式为复数 int16 型二进制数据）拷贝至我们所用的计算机中，以供后续解调处理。

3.2 数据读取与前期处理

解调工作首先通过 MATLAB 运行 `A_read_data.m` 脚本，读取本地信号数据文件。设定每次读取 N 个采样点，及起始点在文件中的位置。读取完成后，依据系统设置的采样率、符号率等关键参数，完成时域数据初步归一化与复数解构处理。

3.3 调制恢复与信号同步

信号处理阶段主要依托 Simulink 完成。运行 `sar_simulink.slx` 模型，依次完成上/下采样模块、符号同步模块与载波同步模块。

值得注意的是，在完整的接收系统中还应包括匹配滤波器部分。由于我们得到的信号数据是已经经过接收系统初步处理过后的基带数据，实测后发现其已经通过了根升余弦滤波器（RCC）处理，因此此处不再添加匹配滤波器模块。

符号同步采用基于 Gardner 算法的定时恢复模块，针对 QPSK 信号具有较好性能。载波同步采用四次方环锁定算法，有效跟踪相位旋转并输出准确信号。最终，将同步后的数据通过 Simulink 导出至 MATLAB 工作区，供后续进行解调处理。

3.4 帧同步与数据解析

在获得高信噪比的 QPSK 信号后，我们在 MATLAB 中运行 `B_data_analyze.m` 脚本进行判决解调。每个采样点经过 IQ 两路分别映射后被转换为对应的位信息流，随后与预定义的卫星同步字节进行滑动匹配，完成帧头定位。

帧头确定后，我们依照空间数据链路协议执行解扰过程，并调用 LDPC 解码模块进行纠错。解码后提取 AOS 帧头字段，包括版本号、卫星 ID、帧计数器等字段，以此确认数据帧的结构合法性与来源可靠性。

3.5 整体流程与异常处理

整个解调流程经多轮数据验证与参数调试后趋于稳定。实验过程中曾遇到以下问题：

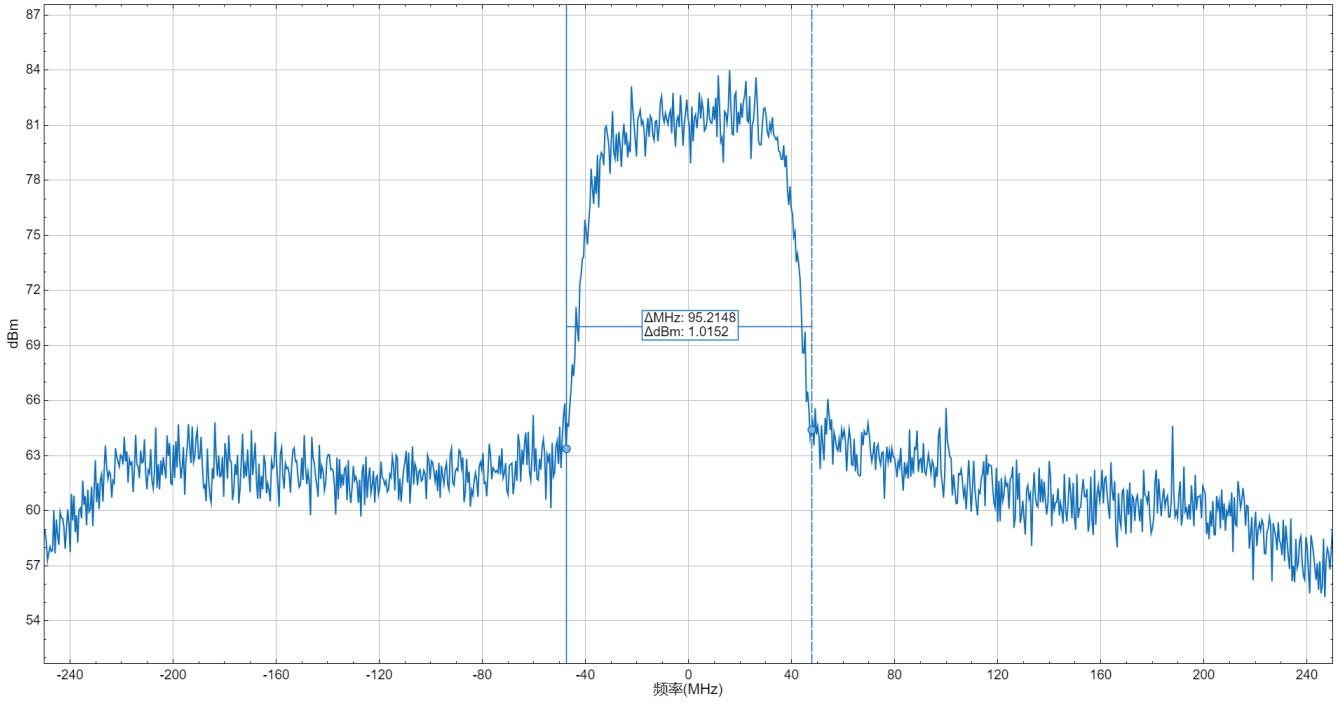
- **载波偏移波动**：因相控阵天线存在微小指向误差，导致频偏变化，经重新配置锁相环滤波器带宽后获得改善；
- **符号同步不稳定**：在信号边缘部分出现失锁，通过调节符号同步环路滤波器带宽成功解决；
- **IQ相位模糊**：当起始条件不同时（比如从不同的时间开始处理），锁相环可能会锁到 0 或者 180 度相位，导致可能会出现 +I/+Q 和 -I/-Q 两种情况。解决办法就是先将其中一路取反，那么两路中一定有一路可以检测到同步字，就能得知正确的相位，再进行后续处理。

最终，我们成功解调出数十个连续 AOS 帧头，并从中提取出包含版本号、卫星 ID、帧计数器等有效字段，充分说明信号接收和解调链路的可行性。

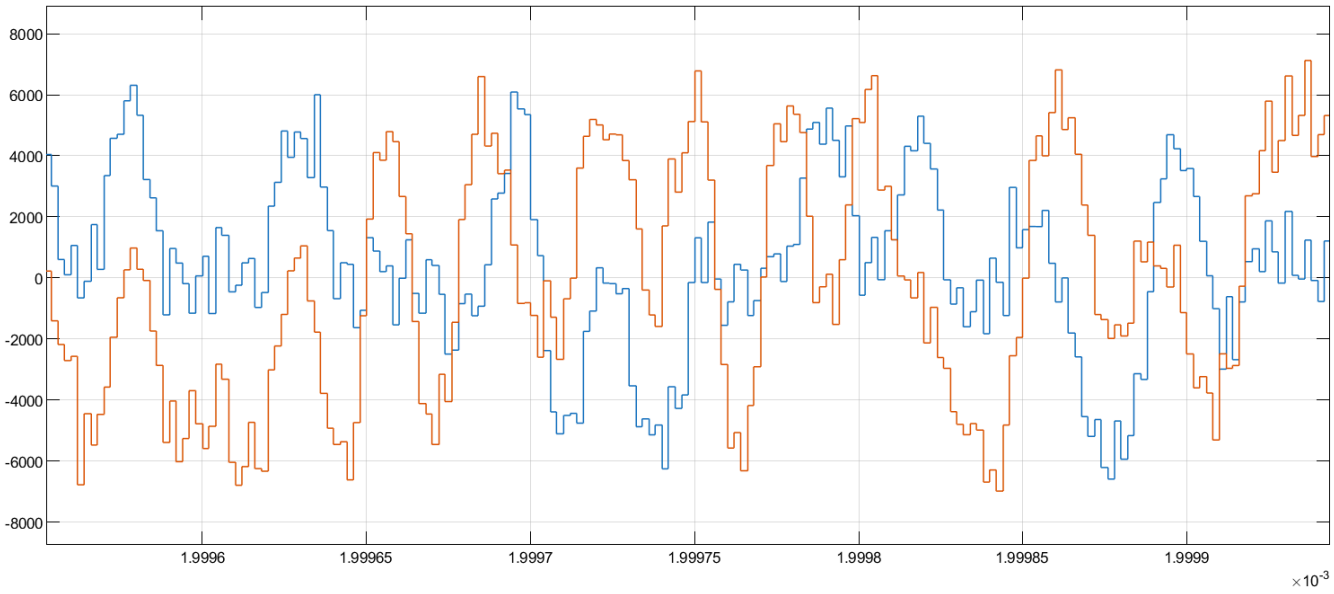
四、数据与图像结果展示

4.1 原始信号频谱与IQ可视化

在完成数据导入与预处理后，首先对原始信号进行频谱分析，以验证接收链路的完整性及频段覆盖情况。下图展示了 T=10 开始某一时段接收到的 IQ 复数信号在频域中的功率谱密度图，中心频率处信号能量集中，带宽约为 95 MHz，信号谱型与预期 QPSK 调制特征一致。

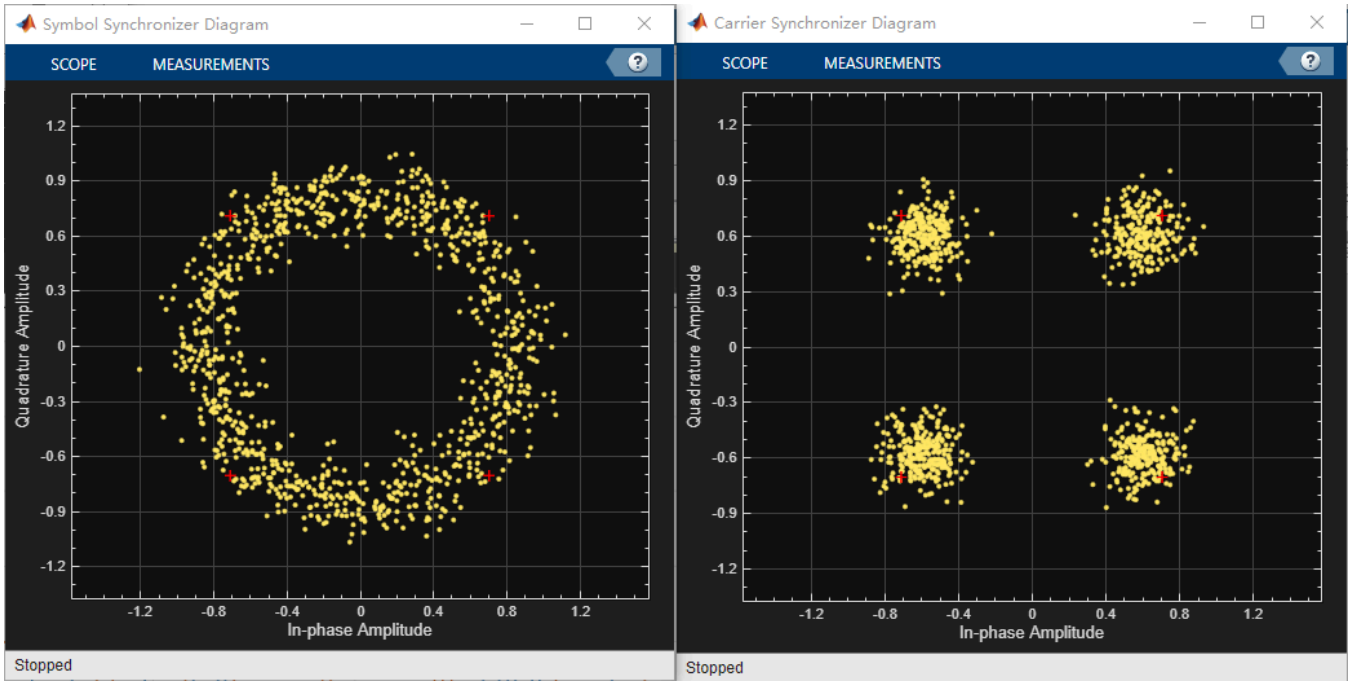


此外，我们绘制了原始复数信号的 I 路和 Q 路时域波形，如下图所示。可以观察到信号呈现出规律性震荡结构，符合已调制状态的信号特征。



4.2 星座图与调制恢复效果

在 Simulink 完成同步处理后，导出的 QPSK 基带信号被映射至星座图进行可视化。下图分别为完成符号同步与载波同步后得到的星座图，可以看到符号同步后星座图呈现圆环状，说明符号同步效果较好。载波同步后星座图中四个象限点聚集明确，表明锁相环成功锁定，信号已可进行可靠判决。



4.3 帧同步与AOS帧结构解析结果

通过滑动匹配定位同步字节后，我们成功识别出多个完整的帧结构。下图中输出了 I、Q 两路同步字节起始位置的索引。

```
I路相位正常
I支路同步字出现位置索引：
  列 1 至 12

          37        8229       16421       24613       32805       40997       49189       57381       65573       73765       81957       90149

  列 13 至 19

       98341      106533      114725      122917      131109      139301      147493

Q支路同步字出现位置索引：
  列 1 至 12

          37        8229       16421       24613       32805       40997       49189       57381       65573       73765       81957       90149

  列 13 至 19

       98341      106533      114725      122917      131109      139301      147493
```

观察可以发现，相邻两个同步字节起始位置间距均为 8192(bit)，即 1024 字节，刚好为“卫星数传信号帧格式说明”文档中给出的帧大小，如下表所示。

同步字	AOS帧头	有效数据	LDPC校验码
4B	6B	886B	128B
0x1ACFFC1D			

下图则展示了 I 路数据帧解扰前后的对比，其中，解扰前的数据跳过了 4 字节同步字，和下方解扰后数据对齐。可以明显观察到解扰前后数据的不同。

I_frames

18x8192 logical

	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	
1	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	0
2	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	0
3	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	0
4	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	0
5	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	0
6	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	0
7	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	0
8	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	0
9	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	0
10	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	0

I_frames_descrambled

18x8160 logical

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
2	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
3	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
4	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
5	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
6	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
7	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
8	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
9	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
10	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0

由于 LDPC 编码为系统码，因此在高信噪比条件下可跳过 LDPC 解码过程，直接截取信息位作为解码后的信息。所以，在成功定位帧头并完成解扰后，我们直接提取出AOS帧头字段，I、Q两路帧头分别如下：

I_aos_data

1x18 struct 包含 9 个字段

字段	Version	SpacecraftID	VirtualChannelID	FrameCount	ReplayFlag	DownlinkFlag	IQFlag	SymbolRate
1	1	40	0	605846	1	0	1	10
2	1	40	0	605847	1	0	1	10
3	1	40	0	605848	1	0	1	10
4	1	40	0	605849	1	0	1	10
5	1	40	0	605850	1	0	1	10
6	1	40	0	605851	1	0	1	10
7	1	40	0	605852	1	0	1	10
8	1	40	0	605853	1	0	1	10
9	1	40	0	605854	1	0	1	10
10	1	40	0	605855	1	0	1	10
11	1	40	0	605856	1	0	1	10
12	1	40	0	605857	1	0	1	10
13	1	40	0	605858	1	0	1	10
14	1	40	0	605859	1	0	1	10
15	1	40	0	605860	1	0	1	10
16	1	40	0	605861	1	0	1	10
17	1	40	0	605862	1	0	1	10
18	1	40	0	605863	1	0	1	10
19								

Q_aos_data

1x18 struct 包含 9 个字段

字段	Version	SpacecraftID	VirtualChannelID	FrameCount	ReplayFlag	DownlinkFlag	IQFlag	SymbolRate
1	1	40	0	605846	1	0	2	10
2	1	40	0	605847	1	0	2	10
3	1	40	0	605848	1	0	2	10
4	1	40	0	605849	1	0	2	10
5	1	40	0	605850	1	0	2	10
6	1	40	0	605851	1	0	2	10
7	1	40	0	605852	1	0	2	10
8	1	40	0	605853	1	0	2	10
9	1	40	0	605854	1	0	2	10
10	1	40	0	605855	1	0	2	10
11	1	40	0	605856	1	0	2	10
12	1	40	0	605857	1	0	2	10
13	1	40	0	605858	1	0	2	10
14	1	40	0	605859	1	0	2	10
15	1	40	0	605860	1	0	2	10
16	1	40	0	605861	1	0	2	10
17	1	40	0	605862	1	0	2	10
18	1	40	0	605863	1	0	2	10
19								

可以看到，FrameCount 字段逐帧递加，说明中间没有丢失帧，且 I 路帧计数与 Q 路帧计数一一对应，说明解调效果较好。

剩下字段的具体含义可以参考航天宏图提供的 AOS 帧头格式，如下图所示：

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

参考格式将接收到的 AOS 帧头翻译后，得到具体信息如下表所示：

定义	内容	含义
版本号	1	固定版本
航天器标识符	40/28H	宏图二号 03 组 A 星
虚拟信道标识	0	有效数据

定义	内容	含义
VCDU计数	605846-605863	帧计数
实时回放标识	1	回放
下传标识	0	单路下传
IQ数据标识	1、2	I 路、Q 路
传输速率标识	10/0b1010	150Mbps

这些字段与预设星载系统参数完全一致，标志着数据链路完整性与解析准确性，充分说明实验过程中信号处理链条已闭环。

五、代码与关键实现细节

5.1 `A_read_data.m`

该脚本的主要功能是从原始 IQ 采样文件中读取指定时间段的复数信号数据，为后续 Simulink 模型中的调制解调处理模块提供输入数据。其流程如下：

1. **参数初始化与采样控制**

   ```
   filename = 'E:\sample_06111';
   N = 1e6;
   fs = 500e6;
   Ts = 1/fs;
   T = 10;
   t = (0:N-1)' * Ts;
   ```

   上述代码设定了采样点数量 $N = 10^6$、采样率 $f_s = 500$ MHz，起始读取时间为第10秒，即实际读取从第10秒起的1百万个复数采样点。由此推算实际读取时长约为 $N/f_s = 2$ 毫秒。
2. **文件读取与字节偏移**

```
start_sample = round(T * fs);
offset = start_sample * 2 * 2;
fseek(fid, offset, 'bof');
raw = fread(fid, [2, N], 'int16=>double');
```

每个采样点由两个 `int16` 数组成（I 路与 Q 路各占 2 字节），因此计算偏移时以 4 字节为单位。函数 `fseek` 跳过起始时间前的采样数据，仅保留目标时间段信号。随后通过 `fread` 读取 2×N 的矩阵，第一行为 I 路，第二行为 Q 路。

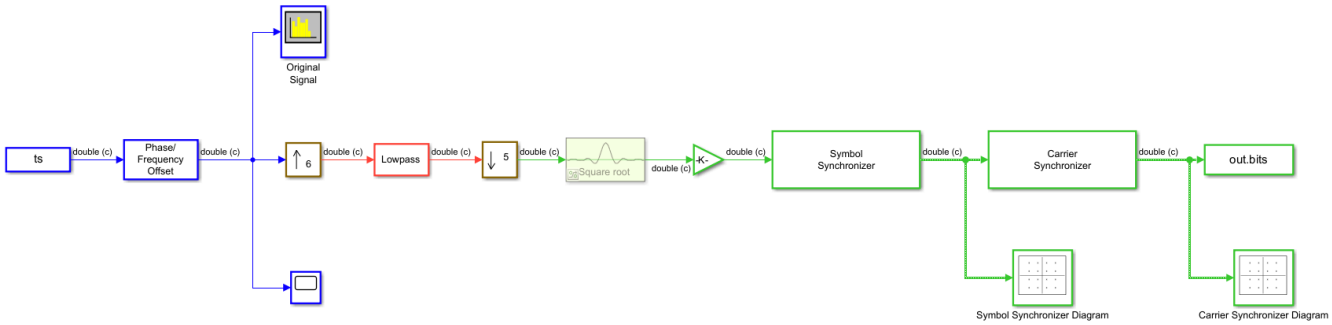
3. **复数信号构建与时间序列封装**

```
data = complex(raw(1, :), raw(2, :)).';
ts = timeseries(data, t);
```

利用 `complex()` 函数将 I 路与 Q 路合成为复数形式信号序列，再配合 Simulink 标准时间轴 `t` 封装为 `timeseries` 对象，便于在 Simulink 中进一步处理。

5.2 `sar_simulink.slx`

该模型完整结构如下：



该 Simulink 模型描述了从复数基带信号输入到同步恢复输出的完整数字接收链路，是实现 QPSK 信号可靠解调的核心子系统。整体流程可以分为以下几个阶段：

1. **相位与频率偏移校正（Phase/Frequency Offset）**
   模型最前端的 `Phase/Frequency Offset` 模块用于对接收到的 IQ 信号进行频偏与相位旋转补偿。该操作模拟了地面站可能因本振频率漂移、卫星多普勒效应等因素引起的载波偏移。补偿后的信号作为主要处理链路的输入，进入后续模块。
2. **上采样与低通滤波**
   - **6倍上采样**：对信号采样率进行 6 倍上采样至 3GHz，为后续滤波与符号对齐提供更高

的时间分辨率。
    - **低通滤波器（Lowpass）**：用于去除上采样过程中引入的频谱镜像和高频噪声，保留原始信号带宽。
3. **下采样与成形滤波**
    - **5倍下采样**：将信号采样率进行 5 倍下采样至 600MHz，此时采样率刚好为符号速率（75MHz）的 8 倍，即每个符号对应 8 个采样点，与符号速率匹配，为后续符号同步阶段做准备。
    - **根升余弦滤波器（Square Root）**：预留接收端匹配滤波，与发射端根升余弦配对，实现理想 Nyquist 滤波，最小化 ISI 影响。此处未启用。
4. **符号同步（Symbol Synchronizer）**
   该模块实现接收端符号时钟恢复，常采用 Gardner 算法，自动估计并校正符号边界位置。在不精确采样条件下仍能准确提取符号点，是实现可靠判决的前提。
   输出同时连接至星座图显示模块 `Symbol Synchronizer Diagram`，用于监视同步前后的星座结构。
5. **载波同步（Carrier Synchronizer）**
   QPSK 信号对相位敏感，因此必须进行载波恢复以消除残余频率偏移。此模块实现四次方环跟踪，可自动锁定至 QPSK 星座固定角度，防止旋转、模糊或镜像等问题。
   该模块输出的结果用于后续判决与解码，并连接至 `Carrier Synchronizer Diagram` 可视化模块用于验证解调星座收敛效果。
6. **输出与接口设计**
   最终输出信号为 `out.bits`，表示已完成同步处理后的复数基带符号数据（常用于 QPSK 星座解映射）。该信号将被导出至 MATLAB 工作区，进入判决与帧同步等进一步处理流程。

整个模型模块化较强，每个子模块完成通信接收链中一类功能，便于测试、调参与替换；流程完整，覆盖频偏校正、采样调整、滤波、同步全流程，适应 QPSK 通信系统的工程实践；可视化强，通过星座图监视同步前后状态，有助于分析系统稳定性与误码风险；参数灵活，采样率设置与重采样比例设计合理，可适配非整数倍采样条件。

5.3 `find_sync_word.m`

此函数用于在解调后的比特流中寻找同步字 `0x1ACFFC1D` 出现的位置，完成 AOS 帧头定位。其处理步骤如下：

1. **输入格式转换与目标同步字构建**

```
sync_word_bin = dec2bin(hex2dec(sync_word_hex), 32) - '0';
```

将输入的十六进制同步字（例如 `1ACFFC1D`）转换为 32 位二进制逻辑向量，得到同步字的精确比特模式。

2. **滑动窗口构造与匹配**

```
idx = bsxfun(@plus, (1:sync_len)', 0:(bits_len - sync_len));
window_matrix = bit(idx);
matches = all(window_matrix == sync_word_bin', 1);
idx_list = find(matches);
```

采用高效向量化方法构建滑动窗口矩阵，每一列为当前窗口内的 32 位比特，逐个与同步字进行全等匹配，返回匹配成功的起始位置索引。

3. **边界判断与异常处理**
若待检测比特流长度不足 32 位，则函数提前返回空向量，避免边界越界错误。

5.4 `extract_frames.m`

该函数用于在识别出同步字后，从其位置起提取完整长度的 AOS 帧（8192 位，等于 1024 字节），为解扰与解码提供结构化数据单元。处理流程如下：

1. **定义帧长与边界检查**

```
FRAME_LEN = 8192;
valid_idx = index(index + FRAME_LEN - 1 <= length(bitstream));
```

根据帧结构定义，每帧长度为 8192 位。首先剔除那些不足以提取完整帧的尾部起始索引，确保不会越界读取。

2. **构造帧内索引矩阵并提取**

```
offsets = 0:(FRAME_LEN - 1);
indices_matrix = valid_idx(:) + offsets;
frames = bitstream(indices_matrix);
```

使用矩阵广播操作构造多帧数据索引，从原始比特流中向量化提取全部帧内容，输出结果为 `N×8192` 矩阵，其中每行对应一帧。

5.5 `descramble_array.m`

该函数用于对已提取并解码的帧数据进行加扰逆过程处理（即解扰），以还原最初的 LDPC 编

码前比特流。其实现基于伪随机序列（PRBS）与初始状态预设的线性反馈移位寄存器（LFSR）。

1. **输入检查与帧维度校验**
   函数要求输入为 `N × 8160` 的逻辑矩阵，其中每行代表一帧数据，长度为 1020 字节（LDPC 编码后，去除同步字后）。若维度不符将直接报错。
2. **LFSR 初始状态设置**

```
switch upper(channel)
    case 'I'
        init_lfsr = ones(1, 15);
    case 'Q'
        init_lfsr = [zeros(1, 7), ones(1, 8)];
```

   根据文档《卫星数传信号帧格式说明》：
   - I 路初相为全 1（15位），即 `111111111111111`
   - Q 路初相为前 7 位 0，后 8 位 1，即 `000000011111111`

   此处严格按照原系统约定设置初始寄存器状态，确保 PRBS 同步。
3. **PRBS序列生成**

```
for i = 1:8160
    prbs_bit = xor(lfsr(1), lfsr(2));
    prbs_seq(i) = lfsr(1);
    lfsr = [lfsr(2:15), prbs_bit];
```

   PRBS 由长度为 15 的 LFSR 生成，反馈多项式为 $X^{15} + X^{14} + 1$，即反馈自第 1、2 位。PRBS 生成长度为 8160，与每帧编码数据长度一致。
4. **解扰操作**
   使用 `xor` 操作将输入比特与对应 PRBS 比特逐位异或，从而完成解扰。处理过程向量化处理，支持批量帧操作。

5.6 `ccsds_ldpc_decoder.m`

此函数是一个基于 Simulink 模型的 CCSDS 标准 LDPC (8160, 7136) 解码器，实现对已解扰比特流的纠错处理，恢复出每帧原始的 892 字节 AOS 信息数据。

1. **输入格式与预处理**
   - 输入为 `N × 8160` 矩阵（LDPC 码长）

- 每位硬判决比特通过下式转换为 LLR（Log Likelihood Ratio）值：

```
llr_frame = C * (1 - 2 * double(input_bits(n, :)));
```

其中 $C = 30$，1 转为 -30，0 转为 +30。

2. **仿真接口构建**
模拟将每帧输入转化为 HDL Simulink 解码器所需的时间序列格式，包括：
   - `dataIn`：LLR 输入信号
   - `validIn`，`startIn`，`endIn`：帧控制信号
   - `simTime`：仿真时间窗口

   然后将这些变量注入 Simulink 模型 `ccsdsLDPCModel.slx` 中进行仿真处理。
3. **解码输出提取**
从 Simulink 输出中提取每帧解码起止索引，筛选有效输出位，最终整理为 `N × 7136` 的矩阵，表示每帧解码后的 892 字节信息位。

5.7 `parseAOSFrames.m`

该函数用于解析 LDPC 解码后得到的 892 字节 AOS 帧数据中前 6 字节的帧头内容。输出为结构体数组，每行为一帧的解析结果。

1. **提取帧头**

```
aosHeaderBits = bitStream(i, 1:48);
```

根据文档中描述，AOS 帧头占 6 字节，即 48 位，包含帧计数器、标识符、通道号等信息。
2. **字段划分与解释**
每个字段在帧头中的比特位置如下：
   - 版本号（Version）：2位
   - 星载ID（SpacecraftID）：8位
   - 虚拟信道ID（VirtualChannelID）：6位
   - 帧计数器（FrameCount）：24位
   - 回放标志、下行标志、IQ标志等控制位：各1~2位
   - 符号速率字段（SymbolRate）：4位

   解析采用 `bin2dec(char(bits + '0'))` 方式将逻辑向量转为十进制，确保输出清晰明了。
3. **结构体输出**
输出为结构体数组 `aosInfo`，字段名称与协议字段保持一致，便于后续调试与校验。

5.8 `B_data_analyze.m`

该脚本是整个信号处理链的收尾部分，作用在于对 Simulink 解调输出的 IQ 复数信号进行相位校验、同步字识别、帧提取、解扰、LDPC 译码以及 AOS 帧头解析，最终形成帧级语义信息。该模块几乎整合了前文所定义的全部工具函数，构成从信号向结构化数据过渡的闭环。

1. **数据提取与判决映射**

```
signal = out.bits.Data;
signal_idx = ~cellfun(@isempty,signal);
signal_complex = cell2mat(signal(signal_idx));
I = real(signal_complex) > 0;
Q = imag(signal_complex) < 0;
```

从 Simulink 输出结构体中提取复数基带信号 `out.bits.Data`，按非空索引拼接为复数序列。然后进行 QPSK 判决操作，将实部与虚部分别判决为二值比特，构成 I 路与 Q 路初始比特流。
判决方式采用硬判决：
   - I 路：`real > 0` 视为逻辑 1，否则为 0；
   - Q 路：`imag < 0` 视为逻辑 1，否则为 0（注意方向性与实际Q路图形对应）。
2. **相位判别与修正**

```
idx_I = find_sync_word(I, sync_word_hex);
idx_Q = find_sync_word(Q, sync_word_hex);
```

使用 `find_sync_word()` 函数对 I/Q 路比特流分别匹配同步字 `0x1ACFFC1D`，判断哪一路解调相位正确：
   - 若 I 路有匹配，视为 I 路正常，相应对 Q 路取反；
   - 若 Q 路有匹配，视为 Q 路正常，相应对 I 路取反；
   - 否则判断为同步失败。

这是为解决 QPSK 星座中 I/Q 信号可能出现镜像翻转或 180° 旋转（相位模糊）的情况。
3. **帧提取与对齐**

```
I_frames = extract_frames(I,idx_Q);
Q_frames = extract_frames(Q,idx_Q);
```

使用同步字索引 `idx_Q` 提取完整帧（8192 位），包括同步字、AOS 帧头、数据区与 LDPC 校验段。提取结果为 N×8192 的帧阵列，分别处理 I、Q 路数据。

选择 `idx_Q` 作为参考，是基于前面逻辑推导出的“哪一路同步字正确”的判断结果。

4. **解扰处理**

```
I_frames_descrambled = descramble_array(I_frames(:,33:end),'I');
Q_frames_descrambled = descramble_array(Q_frames(:,33:end),'Q');
```

每帧前 32 位为同步字（4 字节），去除后剩余 8160 位，作为 `descramble_array()` 的输入。此函数对照帧结构说明中 I/Q 路各自的 PRBS 初始状态进行解扰，还原为 LDPC 编码输出前的原始比特流。
此处已严格遵循帧结构文档中描述的“先 LDPC 编码，再加扰”的处理逻辑，操作顺序为：

i. 去除同步字；
ii. 解扰；
iii. 译码（下一步）。

5. **AOS帧头解析（未译码前）**

```
I_aos_data = parseAOSFrames(I_frames_descrambled);
Q_aos_data = parseAOSFrames(Q_frames_descrambled);
```

此阶段对解扰后但尚未译码的帧数据进行结构解析，用于观察在加扰+LDPC 误码影响下的字段稳定性，例如帧计数器是否能递增、SCID 字段是否稳定等。对调试阶段有辅助价值。

6. **LDPC译码**

```
I_frames_decoded = ccsds_ldpc_decoder(I_frames_descrambled);
Q_frames_decoded = ccsds_ldpc_decoder(Q_frames_descrambled);
```

调用 `ccsds_ldpc_decoder()`，对 N×8160 的帧数据执行基于 Simulink 模型的 LDPC (8160,7136) 解码操作，输出 N×7136 的帧阵列，已去除纠错冗余，并恢复为原始 AOS 帧有效内容（892字节）。
译码前需确保误码率不超过 LDPC 纠错极限，否则解码失败。若遇解码失败时常会在结果中得到全零帧或异常字段。

7. **AOS帧头解析（译码后）**

```
I_aos_data_decoded = parseAOSFrames(I_frames_decoded);
Q_aos_data_decoded = parseAOSFrames(Q_frames_decoded);
```

这是最终版本的帧头解析。此阶段的字段结果即为星上编码逻辑注入的真实数据（如 SCID、帧计数器、信道ID 等），常用于系统验证与实验成效展示。

该脚本贯穿了整个 SAR 信号链路解调的后处理环节，是对之前所实现各个函数的集成调用：

- 前半段实现硬判决、相位判别、同步字搜索与帧提取；
- 中段完成解扰与 AOS 头信息提取；
- 后半段包括 LDPC 解码与最终帧结构提取。

操作逻辑清晰、工程接口分明、模块复用性强，体现了一个标准通信解调任务中从物理层信号到数据链路层结构还原的完整闭环。

六、实验总结与体会

本次实验以北京邮电大学 70 周年校庆活动为契机，结合株洲太空星际公司遥感卫星对西土城校区进行 SAR 成像的任务，完成了一次完整的星地通信链路接收与数据处理的系统实践。从卫星信号的实地接收、基带数据的提取，到信号解调、帧结构恢复，再到 AOS 协议字段的正确解析，整个实验贯穿了遥感通信的多个关键环节，具有较高的理论深度与工程复杂度。

在实验中，我们首先通过相控阵天线稳定接收 X 波段 QPSK 调制信号，并以高采样率（500 MHz）记录原始 IQ 数据。随后，在 MATLAB 与 Simulink 平台上构建了一条完整的数字解调链路，包括频偏补偿、RRC 匹配滤波、符号同步、载波同步等模块，成功实现了复数星座点的还原和判决输出。在此基础上，通过同步字搜索、帧提取、加扰逆处理与 LDPC 译码等步骤，最终提取出多个完整的 AOS 帧结构，并从帧头中获得正确的 SCID、帧计数器等标识字段，验证了解调流程的闭环有效性。

整个过程中，我们掌握了如下几方面的工程技能与认识体会：

1. **对空间通信信号的结构理解更加深入。**通过 AOS 帧结构的实际提取过程，我们不再仅停留于协议定义层面，而是亲身体验了协议的物理实现和工程限制。
2. **复杂通信链路的容错能力有赖于精细参数调试。**例如符号同步与载波同步模块的性能高度依赖于采样率、滤波器参数和反馈环带宽，这对调试能力与系统理解提出了较高要求。
3. **实际数据的“非理想性”促使我们发展了异常识别与相位纠正的能力。**通过对 IQ 支路中相位模糊的识别与取反纠正，我们提高了对实际通信中误差传播的感知和修正能力。
4. **多模块联合调试的重要性凸显。**仅靠单一模块（如滤波器或同步器）无法构建稳定系统，必须通过整体链路的调试、参数协调与结果验证，才能形成可靠、鲁棒的数据处理流程。
5. **软硬件平台协同的能力得到了锻炼。**通过 Simulink 仿真与 MATLAB 代码的结合，以及与实际 SDR 接收链的配合，我们获得了跨平台、跨层次的实践经验。

综上所述，本次实验不仅达成了既定的技术目标——成功恢复遥感 SAR 数传信号中的 AOS 帧头，而且极大地增强了我们对空间遥感数据链路的理解与控制能力。作为一项结合通信原理、协议解析与工程实现的综合性实验，其成果将对我们今后从事星地通信、遥感处理及相关系统开发提供宝贵的理论积累与实战基础。