Breakdown of Sentinel 1 Decoder

By Khavish Govind

# Introduction

Sentinel-1 is a satellite mission operated by the European Space Agency (ESA) as part of the Copernicus Programme, designed for Earth observation with a primary focus on radar imaging. Sentinel-1's main objective is to provide continuous all-weather, day-and-night radar imagery for various applications such as land and ocean monitoring, emergency response, and security. Sentinel-1 is a cornerstone of the Copernicus Programme, leveraging radar technology to offer essential Earth observation capabilities.

# Brief

**Level 0 Data:** Level 0 data consists of unprocessed signals directly received from the satellite's instruments. This data includes raw telemetry, timing information, and uncalibrated sensor readings. It requires further processing and calibration to convert it into meaningful scientific data.

1. **Raw Data Archiving:** Level 0 data serves as the foundation for creating higher-level products and is crucial for archival purposes. It preserves the original sensor readings, enabling future reprocessing with improved algorithms.
2. **Algorithm Development:** Researchers use Level 0 data to develop and refine algorithms for data processing, calibration, and image reconstruction.
3. **Customized Processing:** Different scientific and operational applications may require customized processing techniques

A decoder for Level 0 data from satellites like Sentinel-1 would be highly beneficial for several reasons:

1. **Data Interpretation:** Level 0 data is raw and requires interpretation and transformation into meaningful information. A decoder would convert the raw telemetry, timing information, and sensor readings into a format that users can work with.
2. **Automation of Processing:** Manual processing of Level 0 data can be time-consuming and prone to errors. A decoder automates the initial processing steps.

Overall, a decoder for Level 0 data from satellites like Sentinel-1 plays a critical role in transforming complex raw telemetry into information, particularly in addressing Radio Frequency Interference (RFI). By detecting, identifying, and mitigating RFI sources.

# Theory

**Range Compression –** Compressing a long transmitted pulse into a shorter one using matched filtering. Allowing for better resolution.

**Range Cell Mitigation Correction (RCMC) -** A moving radar system sees a target from different angles over time, causing the target's echo to appear in different range cells. RCMC aligns these echoes into the correct range cell to ensure accurate image formation.

The RCMC filter in the frequency domain is,

# Data Structure

A diagram of a level 0 sentinel

Description automatically generated

A diagram of a data field

Description automatically generated

* A single space packet will generally contain a single echo, thus a burst will generally contain many space packets
* Measurement data is stored as a .dat File and is binary, thus the data must be read in serially and is processing intensive

# Main Classes

1. **IOfile**

* The class contains multiple bursts (azimuth blocks) of radar data from the SAR Level 0 data file (.SAFE) provided.
* The class uses Level0Decoder to decode metadata from the file and organizes it into bursts with constant swath numbers.
* Metadata for these bursts is stored in a pandas DataFrame, and burst-specific radar data is decoded on demand, with optional caching to .npy files.
* The class provides methods to access the filename, packet metadata, ephemeris data, and burst-specific metadata and radar data. Private methods include generating cache filenames and indexing the metadata DataFrame on bursts, with utility functions to ensure data consistency.

1. **IOdecoder**

* It includes a method *decode\_metadata* to read and decode the header of each packet in the file, storing the decoded metadata in a Pandas DataFrame.
* The *decode\_packets* method processes the user data from the specified packets, ensuring that all packets belong to the same swath and contain the same number of quads. The method reads the file and decodes each packet's data using the *user\_data\_decoder*function.
* *Read\_single\_packet* method reads individual packets, extracting primary and secondary headers and the user data payload.

1. **User\_data\_decoder**

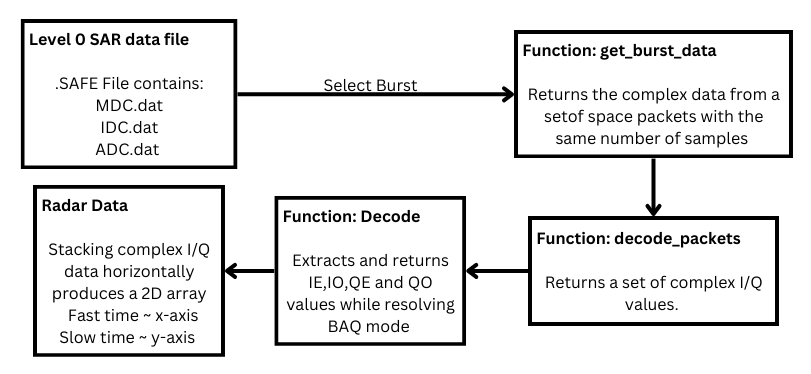
* The class is designed as a facade to interface with various sample extraction and reconstruction classes. It checks the baq\_mode and num\_quads.
* The decode method processes the data according to the specified BAQ mode which can be refered to SAR Space Protocol Data Unit specification document pg.56
  + For mode 0, it decodes bypass data as simple 10-bit words using *decode\_bypass\_data*
  + For modes 3, 4, and 5, an exception is raised as these modes are not yet implemented.
  + For modes 12, 13, and 14, it uses FDBAQDecoder to extract sample codes, bit-rate codes (BRCs), and threshold indexes (THIDXs). It then reconstructs the channel values using *rec.reconstruct\_channel\_vals*.

The decoded data is re-ordered into complex I/Q values and returned

# Basic flow idea

1. Load in the measurement data file contained with the Level 0 .SAFE file
2. Initialize file as a IOfile object
3. Select a burst (i.e 1,2,3, etc) to extract data (only bursts with single type 0 [echo] data will be used)
4. Initialize a variable to store the output of the *get\_burst\_data* function
5. The *get\_burst\_data* function calls the *decode\_packets* function. The function:
   1. Use the primary header to obtain the number of quads and swaths to ensure the selected space packets are correct
   2. Initializes a complex 2D array determined by the number of packets and twice the number of quads
   3. A loop for the number of packets, the *read\_single\_packet* function is called
      1. Each packet’s raw bytes are read in serially
      2. Separating the secondary heading (specifically 62 bytes) from the user data
   4. The information (BAQ Mode, Quads) from the secondary heading is stored and used to decode the User data
6. The *decode* function from within the *decode\_packets* is called. The function:
   1. Using the user data, quads and BAG mode supplied, the decoding of the user data is done separating the data into raw complex I/Q data. Specifically, into In-Phase even (IE), In-Phase odd (IO), Quadrature odd (QO) and Quadrature even (QE).

\*This is due to the ADC samples per PRI are processed in the Rx chain in 4 data channels

1. The IE, IO, QO, QE are arranged and stacked in the 2D array and returned to the variable of the *get\_burst\_data* function

* Note: a selected burst will take a consecutive number of space packets where the number of samples within the packets are the same. Thus, a loop is instantiated in the *decode packets* function to handle the multiple packets

# Image Reconstruction

#### **Define Auxiliary Parameters**

#### This section initializes various parameters required for the SAR processing steps. Key components include image sizes, transmitted pulse parameters, sample rates, fast time vector, slant range vector, frequency axes, spacecraft velocity, and the cosine of the instantaneous squint angle .

#### **Image Sizes**: Determine the size of the radar data in range and azimuth.

#### **Tx Pulse Parameters**: Define constants:

#### speed of light,

#### range decimation factor,

#### pulse repetition interval,

#### rank,

#### suppressed data time,

#### range start time,

#### wavelength,

#### transmitted frequency ​.

#### **Sample Rates**: Compute the sample rates in range and azimuth directions.

#### **Replica Pulse**: Synthesize a replica of the transmitted pulse using the provided pulse parameters

* 1. **Fast Time Vector**:Calculate the fast time vector for range samples.
  2. **Slant Range Vector:** - Compute the slant range for each range cell.
  3. **Frequency Axes**: Define the frequency axes in range and azimuth directions after transforming to the frequency domain.
  4. **Spacecraft Velocity**: Calculate the effective spacecraft velocity .
  5. **Cosine of Squint Angle**: Calculate .

#### **Convert Data to 2D Frequency Domain**

#### Transform the radar data to the frequency domain using FFT along the azimuth and range axes.

#### **Range Compression**

#### Create and apply a matched filter for range compression. The matched filter is synthesized from the transmitted pulse replica ~ given in the replica pulse. This focuses the image along the range axis.

#### **Range Cell Migration Correction (RCMC)**

#### Correct for the range cell migration due to collector motion. The RCMC shift is applied to align phase history associated with each target into a single range bin.

#### **Convert to Range-Doppler Domain**

#### Inverse FFT back to the range domain along the range axis. The image will still be in the frequency domain in azimuth.

#### **Azimuth Compression**

#### Create and apply a matched filter for azimuth compression. This focuses the image along the azimuth axis.

#### **Transform Back to Range-Azimuth Domain**

#### Inverse FFT of each azimuth line to transform back to the range-azimuth domain.

# References

Ahmed Azouz, Mohamed Fouad, et al. "SAR image formation scheme implementation and endorsement sprouting from Level-0 data decoding." (2023).

<https://sentinels.copernicus.eu/documents/247904/2142675/Sentinel-1-SAR-Space-Packet-Protocol-Data-Unit.pdf/d47f3009-a37a-43f9-8b65-da858f6fb1ca?t=1547146144000>

sentinel1decoder (2023) RichHall [[Source code]](https://github.com/Rich-Hall/sentinel1decoder?tab=readme-ov-file)

<https://github.com/Rich-Hall/sentinel1decoder?tab=readme-ov-file>

Addition onto sentinel1decoder (2024)Dr.-Ing. Matthias Weiß, Chief Scientist Office, Bereich Multifunktionale Hochfrequenz- und Radarsysteme (MFR)