xsarslc: SAR processor for ocean Level-1 SLC -> Level-1B XSP

Frederic Nouguier, Alexis Mouche, Antoine Grouazel

Feb 16 2024 at 14:59

HOME

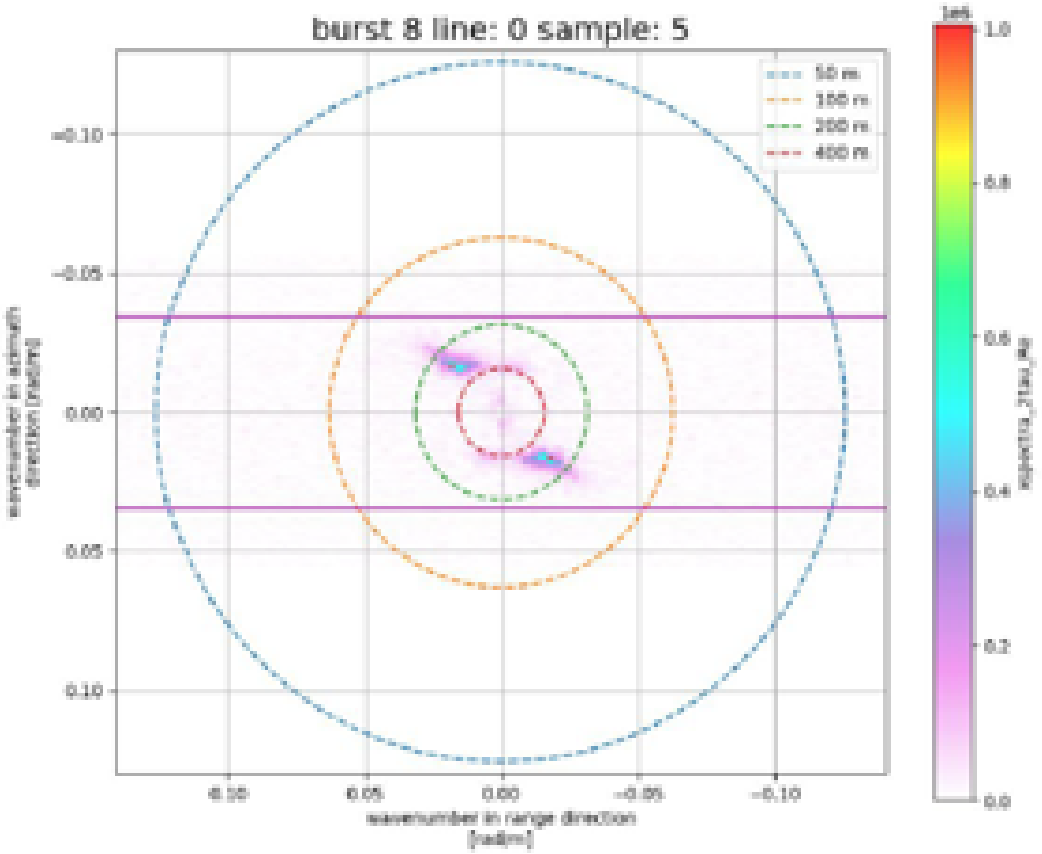
xsarslc is a library to compute cross spectrum from level 1 SAR SLC products. Objets manipulated are all xarray[1].

Acquisition modes handled by **xsarslc** are IW, EW and WV.

The input datatree[2] object can come from any reader library but the original design has been done using xsar[3]

```
import xsarslc
print(xsarslc.__version__)
```

```
2024.1.32.dev10+gc5fbfbb.d20240213
```



0.1 Documentation

0.1.1 Overview

xsarslc can compute both intra burst and inter (i.e. overlapping bursts) burst cross spectrum.

To have comparable cross spectrum among bursts and sub-swaths, we choose to have constant *dk* values, it means that the number of pixels used to compute the cross spectrum is not always the same.

The algorithm is performing 4 different sub-setting in the original complex digital number images:

1) bursts sub-setting

[1] http://xarray.pydata.org
[2] https://github.com/xarray-contrib/datatree
[3] https://github.com/umr-lops/xsar

2) tiles sub-setting

3) periodograms sub-setting

4) looks (in azimuth) sub-setting

Default configuration is set to:

- 20x20 km tiles in the bursts. (less in inter burst where we have about 3 km of overlapping).
- 0% overlapping tiles
- $2.tau$ saved cross spectrum

0.1.2 Algorithm Technical Baseline Document

Note: The Algorithm Technical Baseline Document (ATBD) describes implemented processing steps from Sentinel-1 SLC product to cross spectra

- *Algorithm Level-1 SLC to Level-1B XSP cross spectrum Sentinel-1*

0.1.3 Sentinel-1 Level-1B IFREMER Product Description

Note: It describes the format, the files and the content of Level-1B SAR product.

- *Level-1B SAR Ifremer Product Description*

0.1.4 Examples

Note: here are some examples of usage

- *SAR Sentinel-1 cross spectrum computation from large image Interferometric Wide-swath SLC*
- *Example to compute image cross spectrum between different looks within a WV Sentinel-1 SLC product*
- *A notebook to illustrate how to compute and correct Impulse Response in IW SLC product*
- *A notebook to illustrate how to compute and correct Impulse Response in WV SLC product*
- *illustration of the default Impulse Response (IR) files from xsarslc*

0.1.5 Reference

- *API reference*

0.2 Get in touch

- Report bugs, suggest features or view the source code on github[4].

Last documentation build: Feb 16 2024 at 15:08

0.2.1 Installation

xsar_slc use *xarray-datatree* object as input, any Sentinel-1 TOPS SLC reader could be used to retrieve digital numbers and useful annotations (doppler estimates, bursts, FM-rates, orbits, …). xsar[5] is the reader used to develop this cross spectra estimator library. Installation in a conda[6] environment is recommended.

conda install

```
conda create -n xsar_slc
conda activate xsar_slc
conda install -c conda-forge xsar
cd xsar_slc
pip install .
```

Update xsar_slc to the latest version

To be up to date with the development team, it's recommended to update the installation using pip:

```
pip install git+https://github.com/umr-lops/xsar-slc.git
```

[4] https://github.com/umr-lops/xsar_slc
[5] https://cyclobs.ifremer.fr/static/sarwing_datarmor/xsar
[6] https://docs.anaconda.com/anaconda/install/

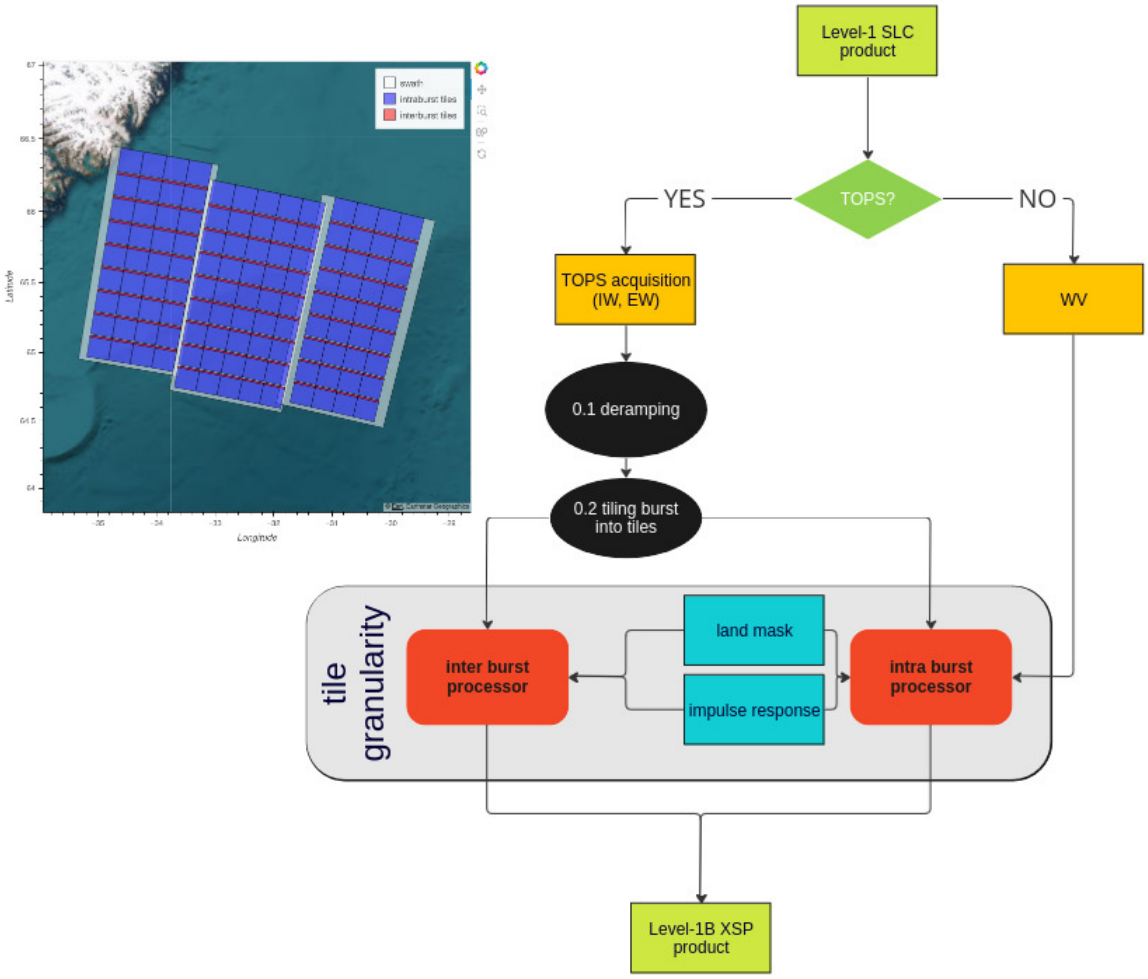
0.2.2 Algorithm Level-1 SLC to Level-1B XSP cross spectrum Sentinel-1

This page stands as the A.T.B.D. (Algorithm Technical Baseline Document) for Sentinel-1 Level-1B product generated at IFREMER .

This ATBD describes the processing steps to transform Sentinel-1 SLC (Single Look Complex) product into a Level-1B XSP. Most of the steps applied in Ifremer SARWAVE Level-1B XSP processor are also described in the official ESA OSW ATBD, see Jonhsen *et al.* [2021].

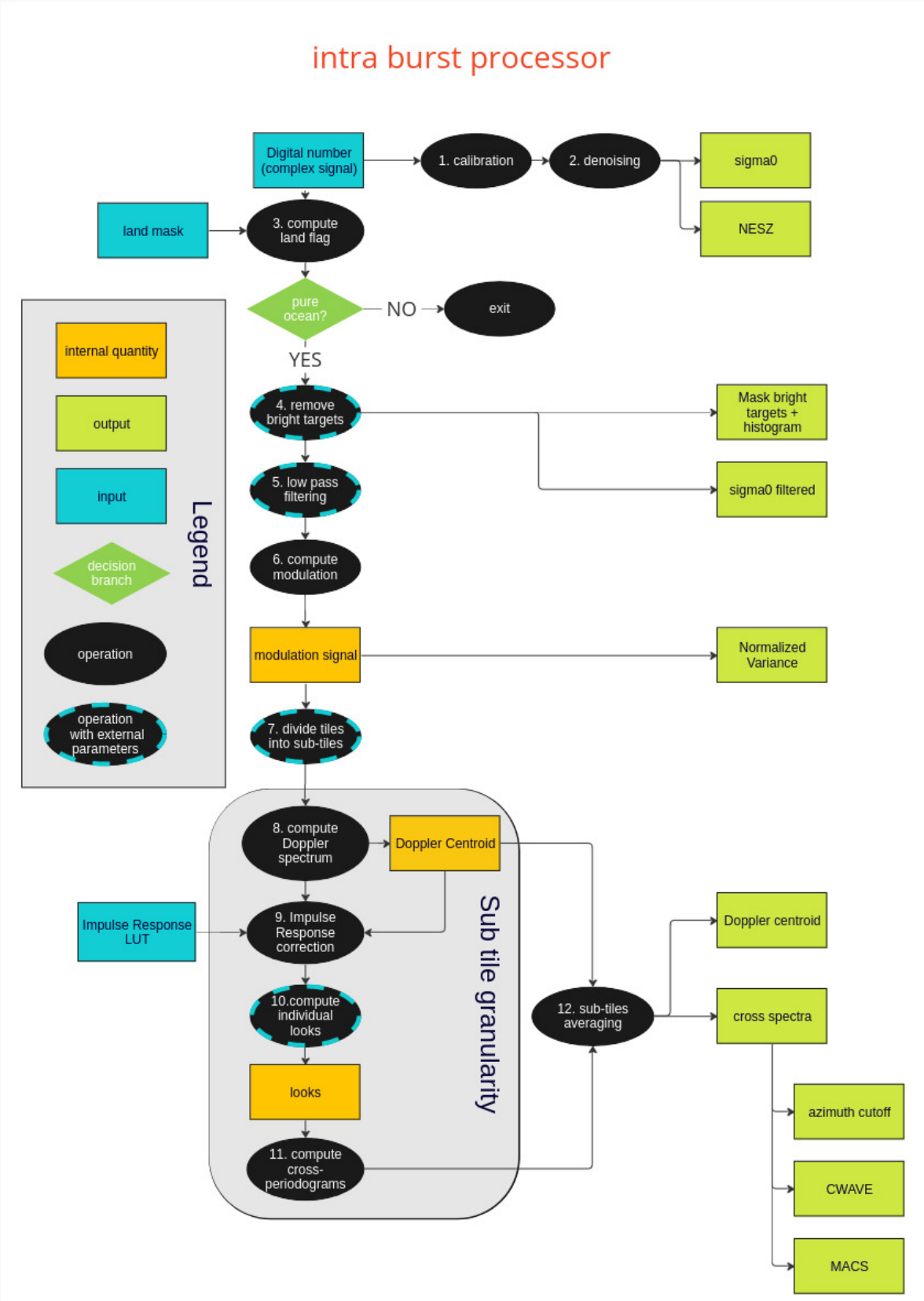
Processors steps overview

This section describes the processing steps using diagrams.

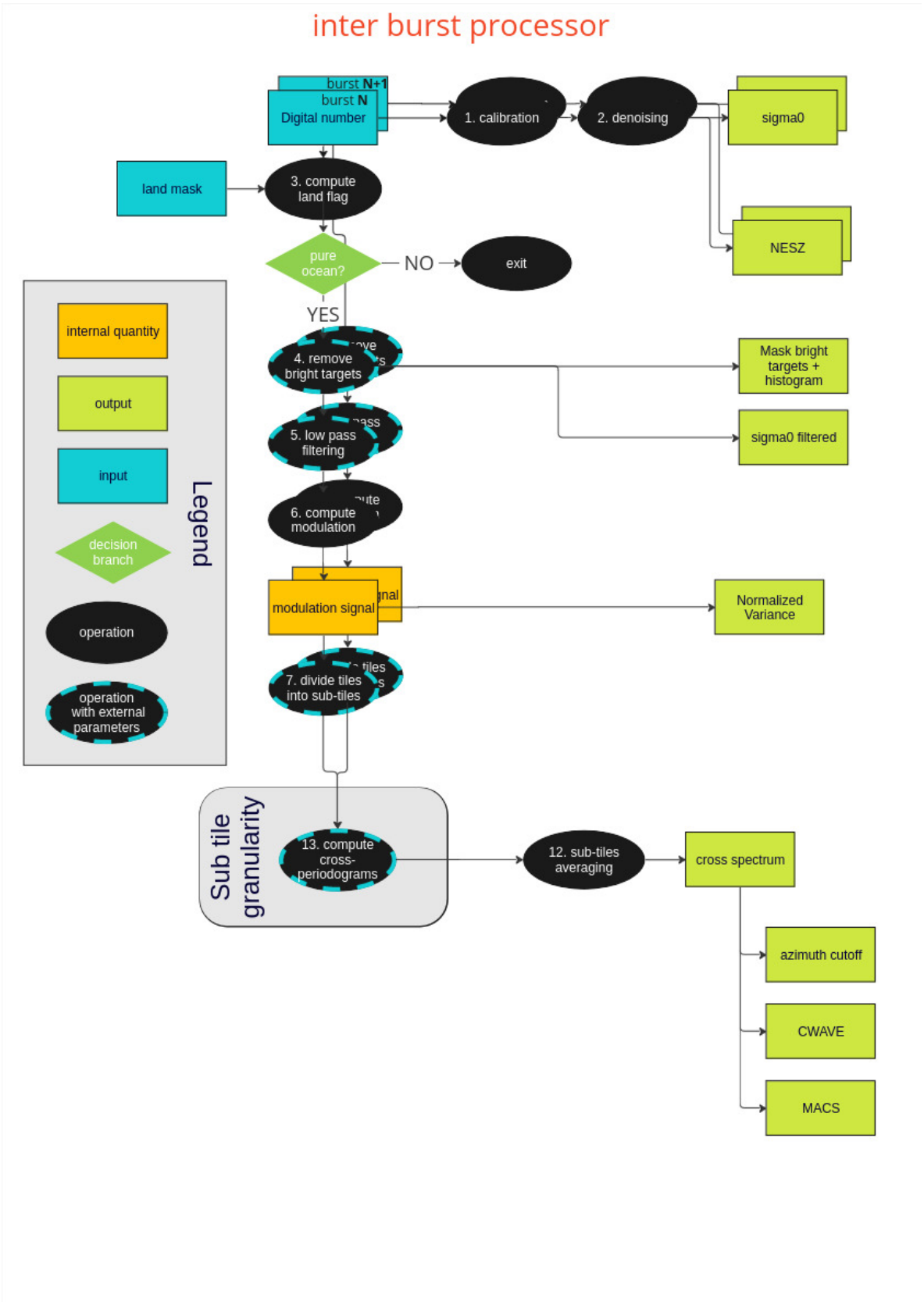


This figure gives a macroscopic description of the steps that allow to process a Sentinel-1 IW SLC acquisition up to a

SARWAVE Level-1B XSP product.



This figure details the different operations (inputs/outputs) that are performed in the intraburst areas of each TOPS subswath.



This figure details the different operations (inputs/outputs) that are performed in the interburst (ie TOPS overlapping)

areas of each TOPS subswath.

Links to detailed processing step descriptions:

Deramping of digital numbers

Lets call $DN(sample, line)$ the 2D matrix of RAW digital numbers with dimensions: sample (fast time) and line (slow time) provided in L1 SLC IW,EW or WV products. Fast time is in the range direction and slow time is the azimuth direction.

Due to the TOPS mode of IWS acquisition, it is mandatory to deramp the complex digital numbers.

- The deramping procedure has to be applied to compensate for the antenna steering rate during the acquisition.
- The deramping procedure do not need to be applied for Wave Mode acquisitions because of the absence of steering.
- The deramping procedure for IW acquisitions follows the steps described in the ESA Technical note Miranda [2017].

The deramped digital number writes:

$$\overline{DN} = DNe^{i\phi} \qquad \text{with} \qquad \phi = -\pi k_t(\tau)(\eta - \eta_{ref})^2$$

Definition of ϕ and how to compute it from L1 SLC products is described in ESA Technical note COPE-GSEG-EOPG-TN-14-0025[7].

uniform ground tiling for Sentinel-1 SLC burst

Burst division into constant ground segment (`tiles`)

SAR SLC product are defined in the radar geometry on a uniform (slant range distance x azimuth distance) grid. In order to define ground segment (`tiles`) with a constant size in range direction, it is mandatory to define and adaptative variable number of slant range points per segment.

Denoting $theta_i$ the incidence angle on the ground at pixel location i, we define the cumulative length as

$$C_0[n] = \Delta s \sum_{i=i_0}^{i_0+n} \frac{1}{\sin(\theta_i)}$$

where Δs is the slant range spacing.

The total length of a ground segment defined between pixel i_0 and $i_N = i_0 + N$ writes:

$$l_b = C_0[N] = \Delta s \sum_{i=i_0}^{i_N} \frac{1}{\sin(\theta_i)} \tag{1}$$

In order to divide l_b into equidistant segments of constant ground length l_t with possible overlaping length l_o ($l_o < l_t$) we define the ground limits (s_n, e_n) of segment n as:

$$\begin{aligned} s_n &= \\ n(l_t - l_o), n \in [0, 1, 2, ..., N] & \\ e_n &= \\ s_n + l_t & \end{aligned} \tag{2}$$

where N is the larger possible integer such as $e_N \leq l_b$.

[7] https://sentinel.esa.int/documents/247904/1653442/sentinel-1-tops-slc_deramping

Centering tiles

Defined as in equation (**??**), the N segments are not centered over the total length l_b. To center the N segments it is enough to add $\frac{l_b - e_N}{2}$ to the segment location, i.e.

$$s_c^n = n(l_t - l_o) + \frac{l_b - e_N}{2}, n \in [0, 1, 2, ..., N]$$

$$e_c^n = s_c^n + l_t + \frac{l_b - e_N}{2}$$

Pixel indexes pertaining to segment n are thus in $[i_{min}^n, i_{max}^n]$ defined such as:

$$i_{min}^n = \text{larger } i \text{ such as } s_c^n > C_0[i]$$

$$i_{max}^n = \text{smaller } i \text{ such as } e_c^n < C_0[i]$$

In practice, l_b is the ground length of a burst ($l_b \approx 80$ km), the slant spacing is $\Delta s \approx 2.5$ m

Computation of calibrated denoised sigma0

Calibration

Sentinel-1 Level-1 SLC product contains radiometric calibration Look-Up Table (LUT) and denoising LUT to compute the required quantity. This procedure is recalled in this note Nuno Miranda [2015] and detailed in this document[8]. A short summarize recalled below can also be found here[9].

The radiometric calibration is applied by the following equation:

$$value(i) = \frac{|DN_i|^2}{A_i^2}$$

where $value(i)$ can be $\beta^0(i)$, $\sigma^0(i)$ or $\gamma(i)$ depending of the used calibration table A_i on the digital number DN_i. The A_i LUT is provided at a lower resolution than the DN and should thus be bi-linearly interpolated.

Denoising

The denoising applied on the Sentinel-1 SLC NRCS in Ifremer SARWAVE Level-1B XSP processor is following the procedure described in Piantanida *et al.* [2017]. The range and azimuth de-noise LUTs must be calibrated matching the radiometric calibration LUT applied to the DN as:

$$noise^{rg/az}(i) = \frac{|\eta_i^{rg/az}|^2}{A_i^2}$$

where $\eta_i^{rg/az}$ are the range and azimuth noise LUTs and $noise^{rg/az}(i)$ are calibrated noise profiles depending on the used calibration table A_i.

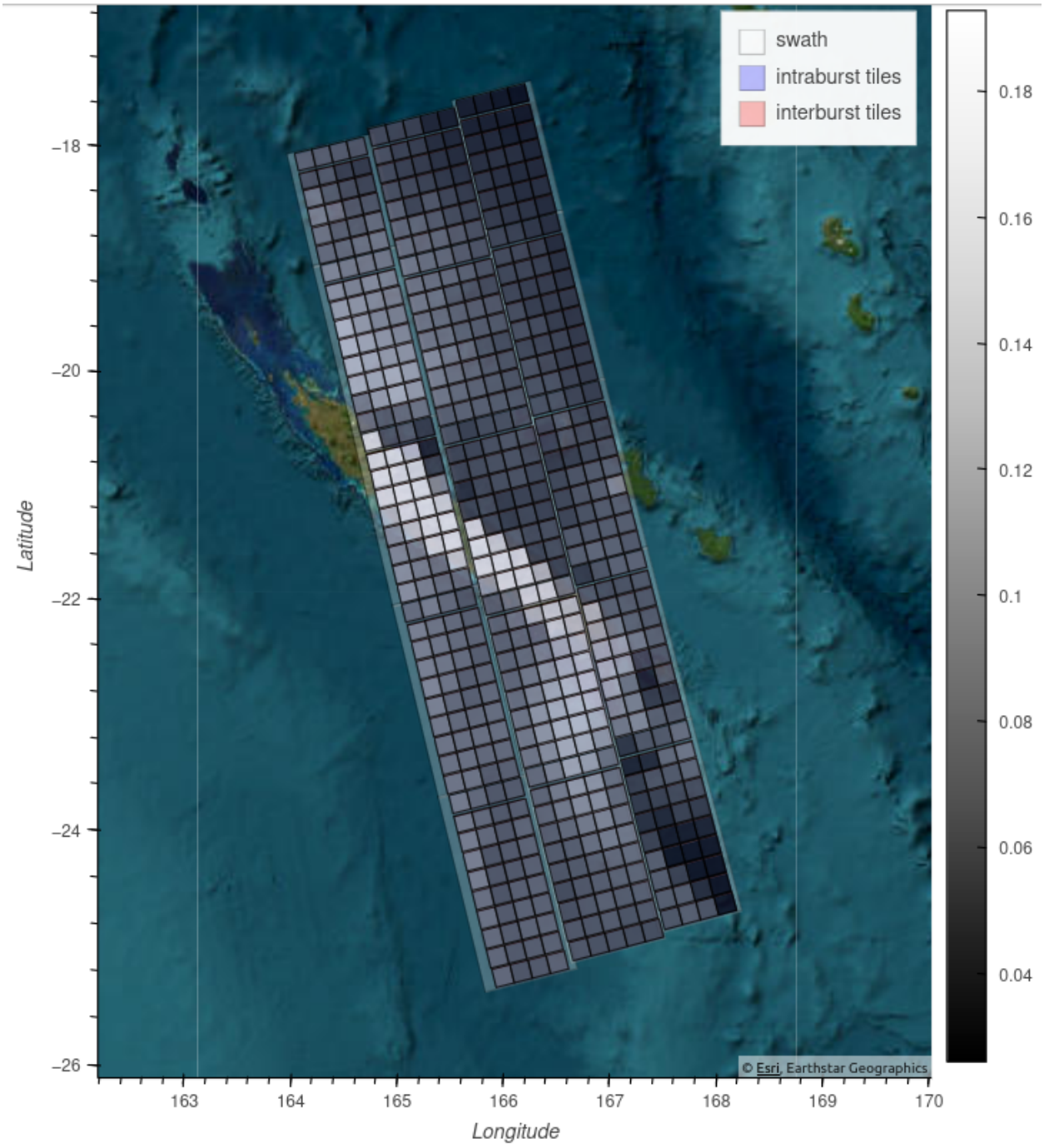
[8] https://sentinel.esa.int/documents/247904/2142675/Thermal-Denoising-of-Products-Generated-by-Sentinel-1-IPF
[9] https://sentinel.esa.int/web/sentinel/radiometric-calibration-of-level-1-products

The radiometrically calibrated and denoised σ^0 thus writes:

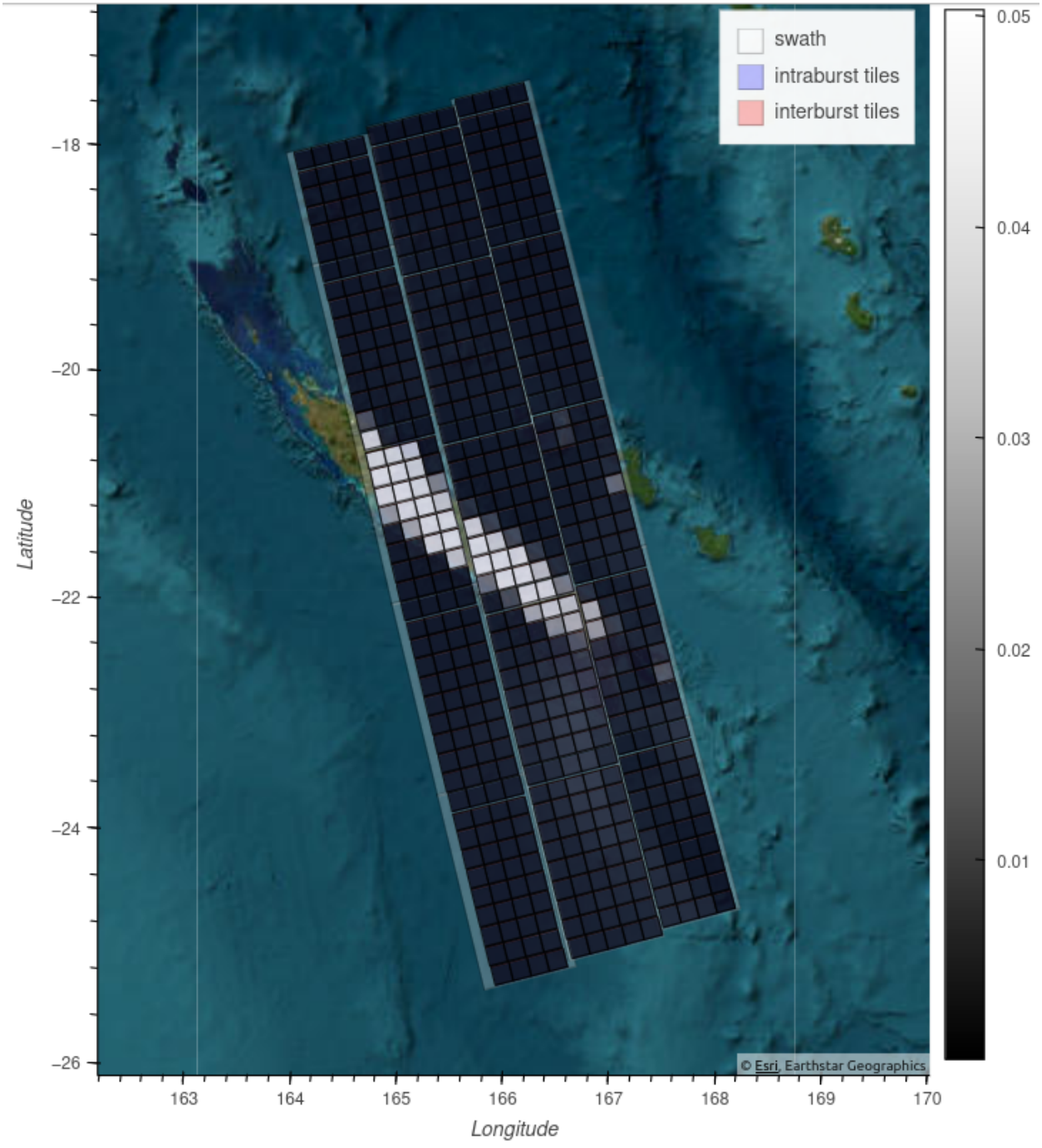
$$\sigma^0(i) = \frac{DN_i^2 - \eta_i^{rg}\eta_i^{az}}{\Sigma_i^0}$$

where η_i^{rg}, η_i^{az} and Σ_i^0 are respectively the range noise, azimuth noise and calibration LUT interpolated over the Digital Number locations i.

Note: η_i^{rg} is both range and azimuth dependent and η_i^{az} is azimuth dependent.



Sentinel-1 IW SLC L1B SARWAVE VV polarization sigma0 denoised (one value per intra burst tile) over Tropical Cyclone NIRAN 2021 (New Caledonia).



Sentinel-1 IW SLC L1B SARWAVE VH polarization sigma0 denoised (one value per intra burst tile) over Tropical Cyclone NIRAN 2021 (New Caledonia).

Compute land mask

To get a land flag (boolean) the intersection between a third-party land-mask and the tile footprint is computed. If the intersection is not null then the computation of spectra is not performed.

Note: currently the processor used the 10 m land-mask distributed through cartopy Python package (see Patterson and Kelso [2024]).

Bright-Target mask computation

Bright target on SAR images are manifestations of extreme return due to target with significantly higher back-scattered signal (boats, wind turbines, …) compared to background. Bright target strongly affect most of significant radar quantities computation such as cross-section, normalized variance or image cross-spectrum evaluation. To derive significant estimation of these radar parameters (from a sea-state estimation perspective), these bright targets must be, as much as possible, removed from the original data before applying conventional algorithms. To this purpose, we will rely on the Cell-Averaging CFAR (Constant False Alarm Rate) algorithm to identify the bright target location. The CFAR method spots (Cell-averaging) bright targets having significantly different signal statistical characteristics compared to the surounding (clutter) cells. A guard distance (guard cells) from the target location ensures to keep bright target spatial contamination from the reference clutter (training cells).

The CA-CFAR algorithm is applied on the absolute value of Digital Numbers $|\widetilde{DN}|$.

The bright target mask is computed as follow:

1. define a target typical size [m] and corresponding number of pixels
2. define a guard size [m] (g_{rg}, g_{az}) and corresponding number of pixels
3. define a clutter size [m] (c_{rg}, c_{az}) and corresponding number of pixels
4. Convolve original data by a uniform kernel having target typical sizes. Resulting data writes $\langle t \rangle$
4. Define a cluster kernel based on the aforementioned guard and clutter sizes.

The convolving kernel writes:

$$K(rg, az) = \begin{cases} 1 & \text{if } \left(\frac{rg}{g_{rg}}\right)^2 + \left(\frac{az}{g_{az}}\right)^2 > \frac{1}{2^2} \quad \text{and} \quad \left(\frac{rg}{c_{rg}}\right)^2 + \left(\frac{az}{c_{az}}\right)^2 < \frac{1}{2^2} \\ 0 & \text{else} \end{cases}$$

where $g_{rg}, g_{az}, c_{rg}, c_{az}$ respectively stands for guard sizes and clutter sizes in range and azimuth direction.

5. Convolving the original (respectively abs-squared) data with this kernel provide reference clutter mean $\langle c \rangle$ and variance $\langle (c - \langle c \rangle)^2 \rangle$ per pixel.
6. Difference of target mean and reference mean normalized by square root of reference standard deviation gives a relative normalized ratio r_T that is to compare with a fixed treshold $BT_{treshold}$.

$$r_T = \frac{\langle c \rangle - \langle t \rangle}{\sqrt{\langle (c - \langle c \rangle)^2 \rangle}}$$

7. Bright target mask is defined as follow

$$BT_{mask} = \begin{cases} \text{True if } r > BT_{treshold} \\ \text{False if } r < BT_{treshold} \end{cases}$$

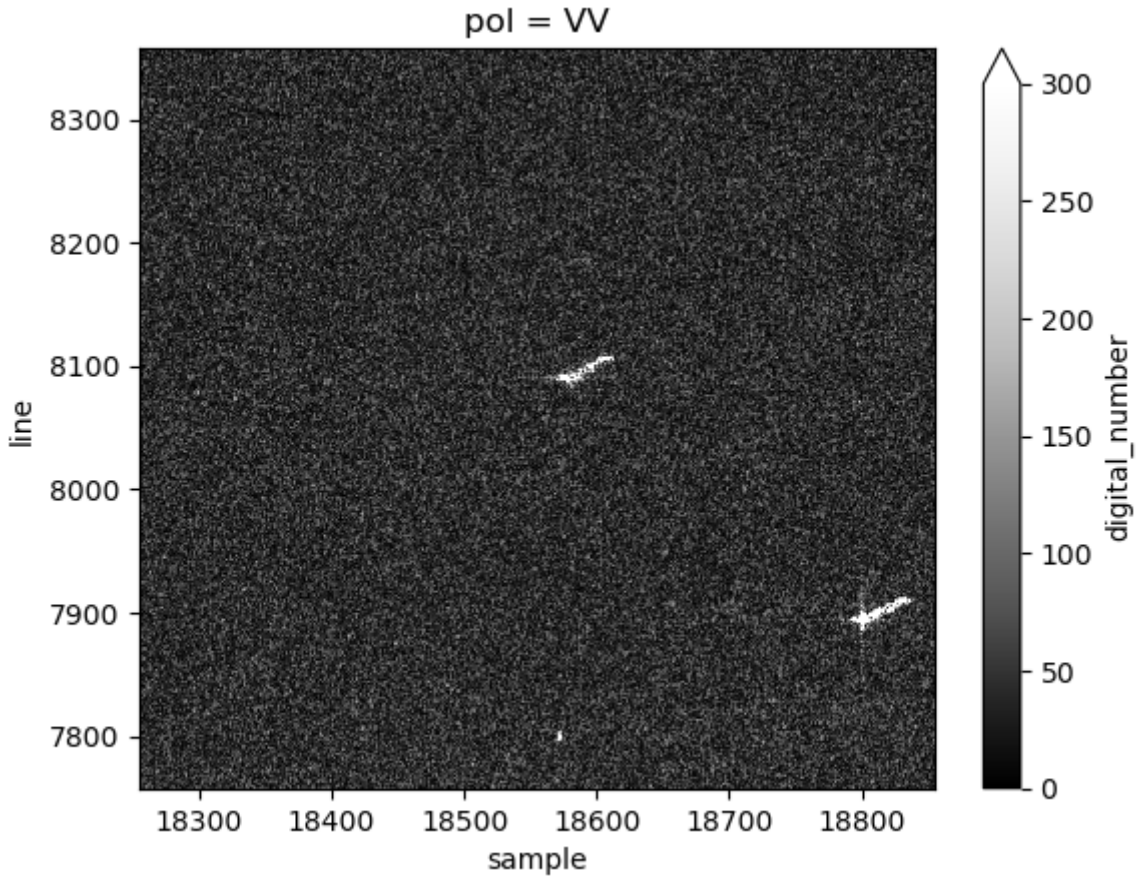
8. Original data is masked with the resulting BT_{mask} and steps 5 to 7 is then recursively applied. The recursive operation progressively remove targets that could have been missed in previous iteration du to spatial contamination of another close target.

9. An optional additional procedure called neighbor filtering can be applied after step 7. Synthetic Aperture Radar azimuthal compression makes bright targets signature to span over a larger distante than the actual size of the bright target. As a consequence, it is common to have spread brigh points around a stronger bright target. To remove such points, bright target mask derived in step 7 is dilated (binary dilation) and relative ratio r_T of points in the dilation radius are tested against a neighboor treshold $BT'_{treshold}$ smaller than $BT_{treshold}$. Bright target mask is dilated for dilated pixels where $r_T > BT'_{treshold}$.

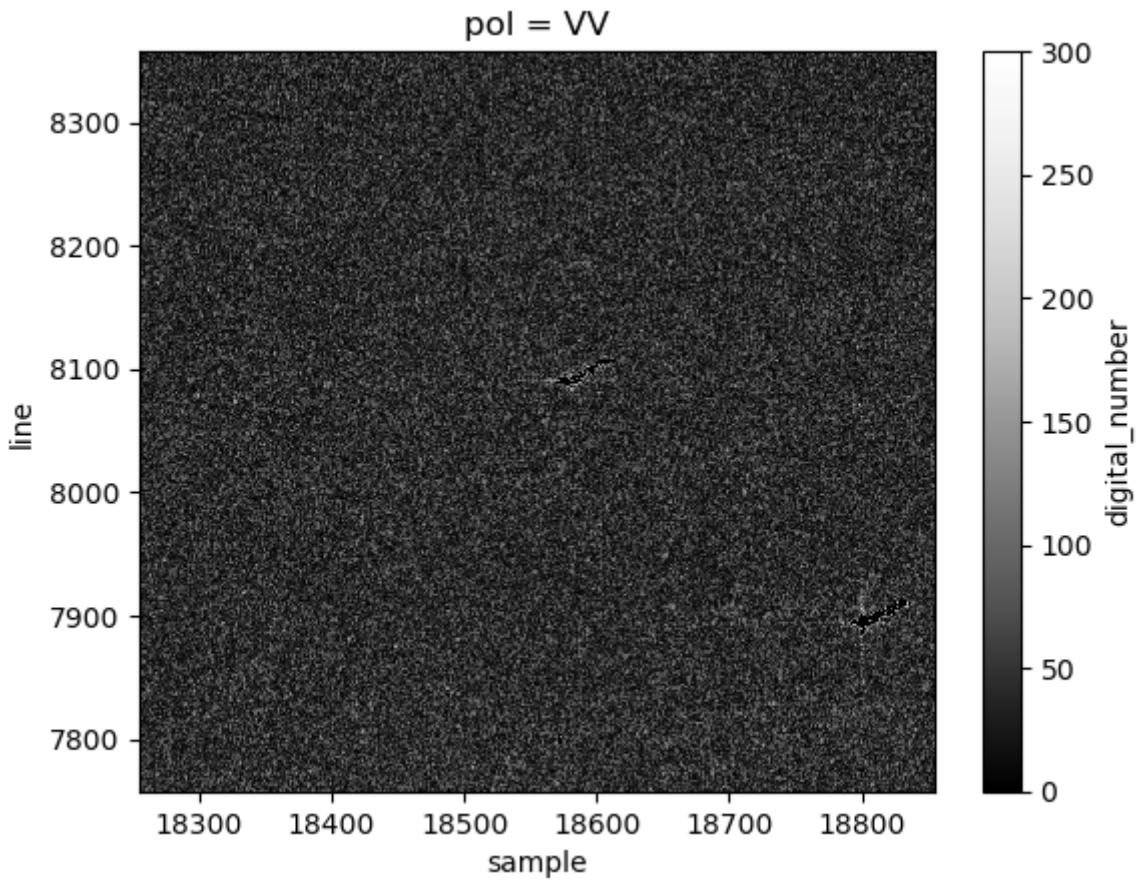
Typical values for parameters listed in this procedure are: $(t_{rg}, t_{az}) = (5m, 5m), (g_{rg}, g_{az}) = (350m, 350m), (c_{rg}, c_{az}) = (1km, 1km), BT_{treshold} = 10, BT'_{treshold} = 5$.

Illustration of the algorithm on the first subswath in VV polarisation coming from S1A_IW_SLC__1SDV_20220612T173328_20220612T173355_043633_05359A_1153.SAFE.

Before:



After:



Complex modulation signal

The SAR image cross-spectra is meant to quantify the spatial frequency content of the cross-section modulations. The modulations are the relative variation of the radar cross-section relatively to the mean radar cross section of the image. As we are interested with the frequency content due to wave modulation, we define the mean radar cross-section as the average of the radar cross-section over a prescribed image extension (typically 1 km x 1 km).

It writes:

$$I_{low} = |\overline{DN}|^2 \star G$$

where $\star$ operators stands for the convolution and with G a normalized Gaussian-filter with a customizable 1 km x 1 km standard deviations.

The complex Digital Number modulations thus writes:

$$\widetilde{DN} = \frac{\overline{DN}}{\sqrt{I_{low}}} \tag{3}$$

Computation of the sigma0 normalized variance

The normalized variance is the variance of the Digital number defined over a prescribed spatial extension. In the baseline L1B SLC processor, the variance is computed at a tile level.

It writes:

$$nv \triangleq \frac{\left\langle (m - \langle m \rangle)^2 \right\rangle}{\langle m \rangle^2}$$

where $m = \left|\widetilde{DN}\right|^2$ and $\widetilde{DN}$ is defined in equation (**??**) .

Divide tiles into sub-tiles

The spectrum of the modulation signal over a full tile is noisy du to speckle contamination. It also contains very low frequency signal which is not usefull to derive wave spectrum. To reduce noise level and derive better wave modulation content, we use a Welsh-like methodology. Each tile is divived into (overlapping) sub-tiles with parametrized size (typically 1.8 km x 1.8 km). Doppler cendroids and look cross-periodogram derived in each sub-tile are then averaged to increase signal-to-noise ratio.

Computation of azimuthal Doppler centroid

Lets call rg and az as respectively the range and azimuth spatial vectors associated with the selected image. rg = sample $\times$ d_{rg} and az = line $\times$ d_{az} where d_{rg} and d_{az} are respectively the sample and line spacing in meters. The azimuthal Doppler spectrum writes:

$$D(rg, f_{az}) = \left| \int \widetilde{DN}(rg, az) e^{-i2\pi f_{az} az} d_{az} \right|^2 \tag{4}$$

The azimuthal Doppler centroid is the mean azimuth frequency of the azimuthal Doppler spectrum. In order to get a good estimation of the Doppler centroid, the average of the Doppler spectrum is computed on the range direction:

$$\overline{D}(f_{az}) = \langle D(rg, f_{az}) \rangle_{rg}$$

Azimuthal Doppler spectrum (5)

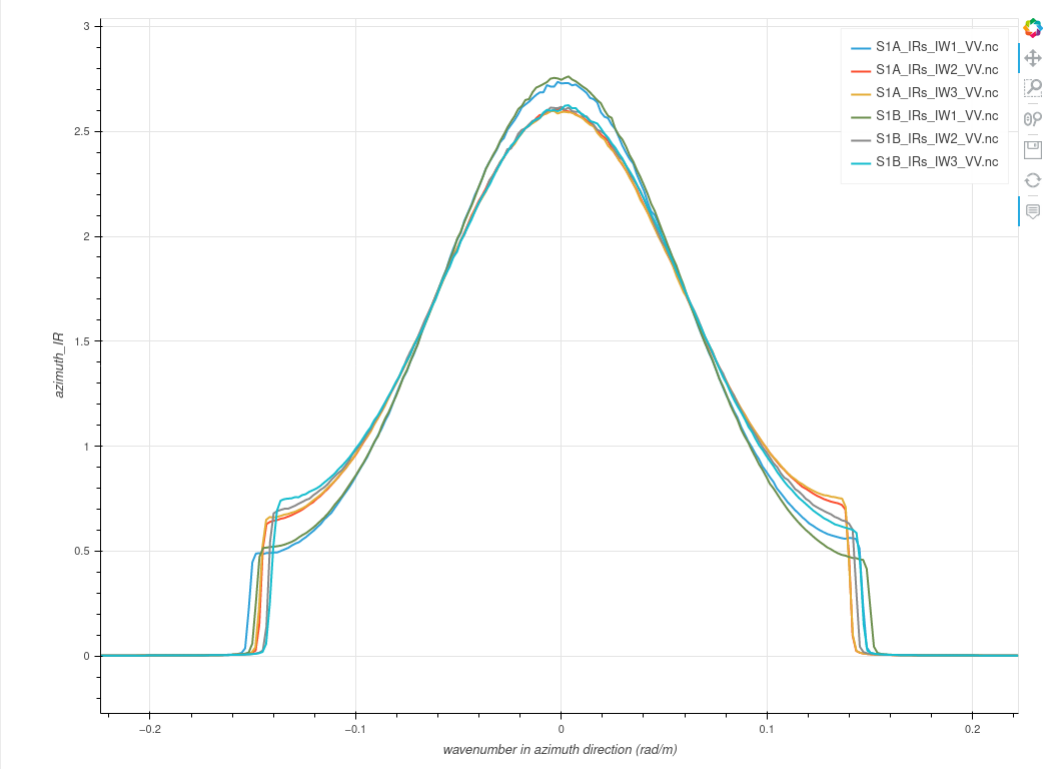


figure (**??**):Example of Doppler spectrum.

The azimuthal Doppler centroid is, by definition, the mean (first order moment) of the Doppler spectrum, namely:

$$DC \triangleq \frac{\int f_{az} \overline{D}(f_{az}) df_{az}}{\int \overline{D}(f_{az}) df_{az}} \tag{6}$$

However, since the azimuthal Doppler spectrum is not symmetric due to windowing processing applied during the generation of the L1 SLC the estimation of the DC using equation (**??**) is biased. In practise, the DC is computed by fitting a Gaussian curve on the Doppler spectrum to find the position of the maximum.

Note: This should be updated in the future.

Computation of centered and normalized Doppler spectrum

The that the Doppler spectrum is not centered around zero nor symmetric relatively to its maximum. Several explanations can be given to explain this two characteristics. The not centered value of the azimuthal centroid can be due, among others, to some geophysical aspects such as the observed scene mean motion but also on some instrument uncorrected geometry and uncompensated antenna properties.

The disymmetric shape can also be due to some uncompensated instrument effect but also on applied signal processing such as windowing or interpolation.

In order to correctly further process the Doppler spectrum, it is mandatory to compensate as much as possible these effects with a two step processing:

1. centering the Doppler spectrum
2. Normalize the Doppler spectrum by the Impulse Response of the instrument

Centering the Doppler spectrum

Centering the Doppler spectrum and computing the 2D Fourier Transform of the complex modulation signal writes:

$$FT^{2D}\left[\widetilde{DN}_c\right] = \int \widetilde{DN}(rg, az) e^{-i2\pi\ DC\ az} e^{-i2\pi(f_{az}az + f_{rg}rg} d_{az} d_{rg} \tag{7}$$

Normalization of the Doppler spectrum by the Impulse Response of the instrument

These Impulse Responses have been computed over homogeneous and motion-less surfaces, averaged and stored. The dataset used to compute theses response is available here and the numerical code to produce them refers to `xsarslc.processing.impulseResponse.compute_IR`.

The normalization of the doppler spectrum is performed by *`xsarslc.processing.intraburst.compute_looks`* method.

Range Doppler spectrum IW VV (8)

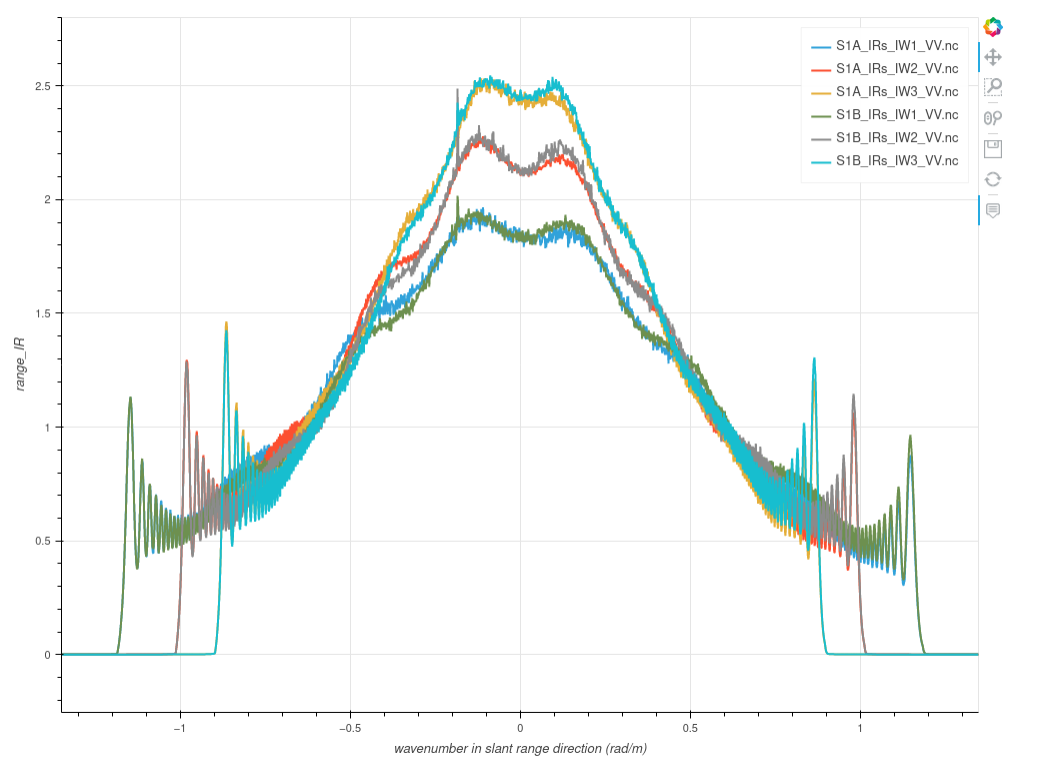


figure (**??**) :Example of Doppler spectrum along range.

Azimuth Doppler spectrum WV VV (9)

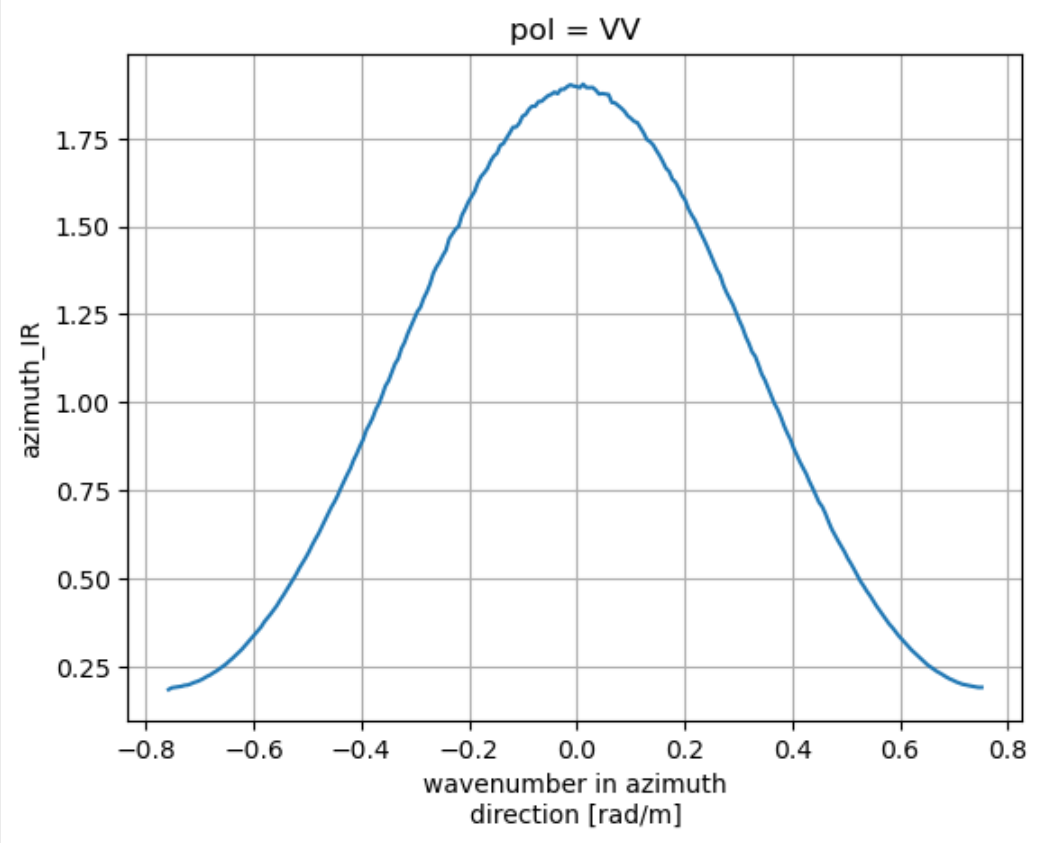


figure (**??**) :Example of Doppler spectrum along azimuth.

Range Doppler spectrum WV VV (10)

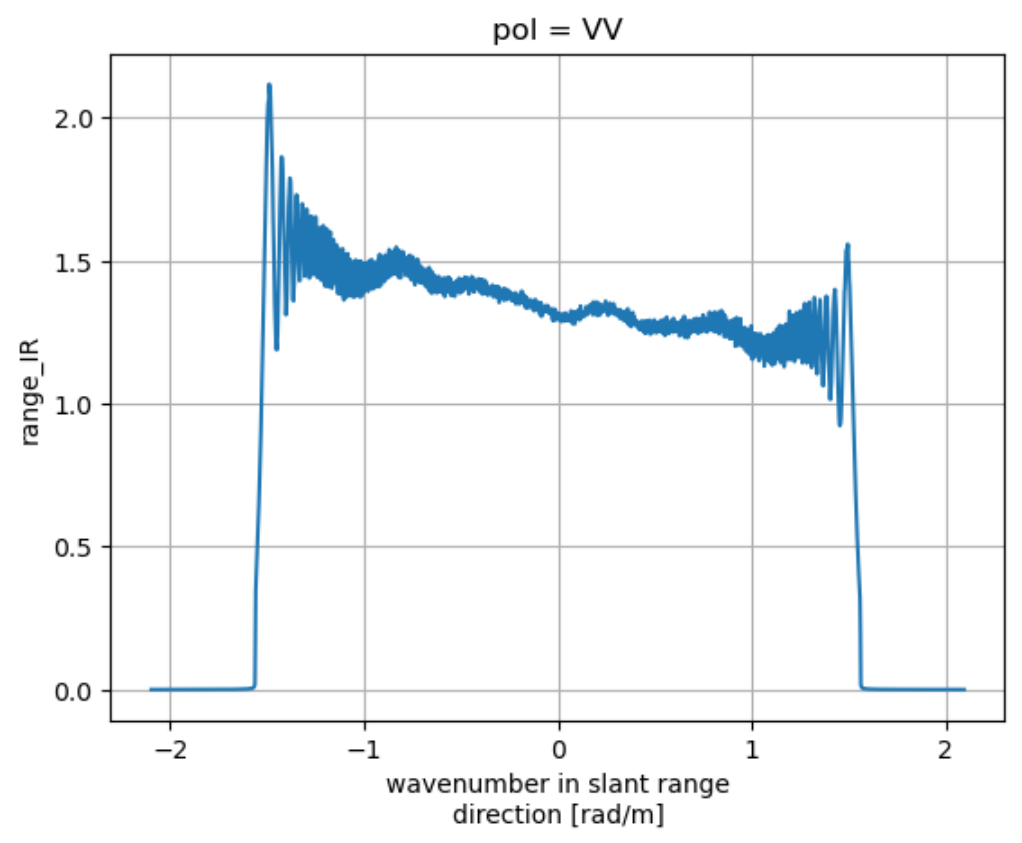


figure (**??**) :Example of Doppler spectrum along range.

The normalization with the instrument Impulse Response is realized in the Fourier domain and writes:

$$FT^{2D}\left[\widetilde{DN_c}\right](f_{rg}, f_{az}) = \frac{FT^{2D}[\widetilde{DN}_c]}{\sqrt{IR_{rg}(f_{rg})}\sqrt{IR_{az}(f_{az})}}$$

with IR_{rg} and IR_{az} being the Impulse Response in range and azimuth direction for the considered acquisition mode.

Note: in *xsarslc* library the methods to estimate the Impulse Response are `xsarslc.processing.impulseResponse.compute_IWS_subswath_Impulse_Response()` and `xsarslc.processing.impulseResponse.compute_WV_Impulse_Response()`

Computation of look cross-spectra (WV and IW products)

The main steps to compute the look cross-spectra are following the paper Engen and Johnsen [1995].

Extraction of azimuthal looks

The extraction of azimuthal looks is computed as follow:

1. Taking the Inverse Fourier Transform of $FT^{2D}\left[\widetilde{\underline{DN_c}}\right]$ in the range direction.
2. Slicing the returned azimuthal Doppler bandwidth into n portions.
3. Taking the Inverse Fourier Transform of each portion in the azimuthal direction.
4. Normalizing each look energy. (if necessary)
5. Detect the look

They are evaluated as follow:

$$FT^{1D}\left[\widetilde{\underline{DN_c}}\right](rg, f_{az}) = \frac{1}{2\pi}\int FT^{2D}\left[\widetilde{\underline{DN_c}}\right] e^{i2\pi f_{rg} rg} df_{rg}$$

The second and third step corresponding to the extraction of look i writes:

$$\widetilde{\underline{DN_c}}^{i}(rg, az) = \frac{1}{2\pi}\int FT^{1D}\left[\widetilde{\underline{DN_c}}\right](rg, f_{az}) W_i(f_{az}) e^{i2\pi f_{az} az} df_{az}$$

where W_i is the weighting function corresponding to slice i in the azimuthal spectrum.

Figure ref{} shows $\left|FT^{1D}\left[\widetilde{\underline{DN_c}}\right](rg, f_{az})\right|^2$ averaged over the range direction and the weighting function of a look.

Detecting look i and normalizing its energy finally writes:

$$look^i(rg, az) = \frac{\left|\widetilde{\underline{DN_c}}^{i}\right|^2}{\sum_{rg,az}\left|\widetilde{\underline{DN_c}}^{i}\right|^2}$$

In practice, the width of the slicing function W_i is defined relatively to the total frequency range of the azimuthal Doppler spectrum. The baseline processing relies on a division into 3 looks and each look contains 25% of the total Doppler frequency range. The remaining 25% are located at the two borders of the frequency axis (12.5% on each side).

Looks cross-spectra

Cross-spectra between look i and look $i + n$ writes:

$$XS^{n\tau}(f_{rg}, f_{az}) = FT^{2D}[look^i] \cdot FT^{2D}[look^{i+n}]^\star$$

where the $\star$ symbol stands for the complex conjugate and where the definition of the 2D Fourier Transform FT^{2D} is

$$F(f_{rg}, f_{az}) \triangleq FT^{2D}[f(rg, az)] = \iint f(rg, az) e^{-i2\pi(f_{az} az + f_{rg} rg)} d_{az}\, d_{rg}$$

The time separation "τ" between two consecutive looks writes:

$$\tau = SaD \times look_{sep}$$

where SaD and $look_{sep}$ are respectively the Synthetic aperture Duration [second] and the look separation.

They writes:

$$SaD = \frac{c \times s}{2 f_r V_{sat} \Delta_{az}}$$
$$look_{sep} = look_{width} \times (1 - look_{overlap})$$

with c, s, f_r, V_{sat}, Δ_{az} being respectively the speed of light, the slant range distance, the radar frequency, the satellite ground velocity and the azimuth spacing. In the baseline processing, $look_{width} = 0.2$ for IW, $look_{width} = 0.25$ for WV and $look_{overlap} = 0$.

Azimuthal cutoff computation

The azimuthal cutoff is a characteristic distance defining the maximum wavelengh that the SAR was able to recover in the azimuth distance. It characterizes its effective azimuthal resolution. The sea surface waves motion is responsible for a strong smearing in the azimuth direction. This smearing effect largely increases with the wave motion and is a good proxy for wind velocity.

The azimuthal cutoff is computed as follow.

1. Computation of covariance function as the Inverse Fourier Transform of the cross-spectrum
2. Averaging the covariance function on the range axis (or taking a transect)
3. Normalize by it maximum
4. Fit a Gaussian function and returns its standard deviation

The covariance function writes:

$$\rho(rg, az) = IFT^{2D}\left[\Re e(XS^{n\tau})\right)$$

where $\Re e$ stands for the real part and where *n=2* in the baseline processing. A Gaussian fit is applied on

$$\underline{\rho}(az) = \frac{\rho(rg = 0, az)}{\rho(rg = 0, az = 0)}$$

over the range span [-500,500] in the baseline processing. In the literature, the Gaussian fit can also be done over the range averaged covariance function $\langle \rho(rg, az) \rangle_{rg}$. The Gaussian fit is realized with a least square difference cost function and a gradient descent methodology. The azimuthal cutoff λ is defined as the standard deviation of the fitted function:

$$\exp\left(\frac{-az^2}{2\lambda^2}\right)$$

Computation of CWAVE parameters

CWAVE parameters is a concept described in the paper Schulz-Stellenfleth *et al.* [2007]. CWAVE parameters are based on the SAR image x-spectrum.

The spectral information of the normalized x-spectrum is decomposed according to orthonormal functions H_{ij} defined as tensor products of Gegenbauer polynomial $G_i(\alpha_k(k_x, k_y))$ and harmonic $F_j(\alpha_\phi(k_x, k_y))$ functions defined in the azimuth k_y and range k_y wave-number space.

This yields to the general formulation of CWAVE parameters C_{ij}:

$$C_{ij} = \sum_{k_x, k_y} \overline{P}(k_x, k_y) H_{ij}(k_x, k_y) dk_x dk_y,$$

with $i \in [1, n_k]$ and $j \in [1, n_\phi]$. In this study $n_k = 4$ and $n_\phi = 5$.

The orthonormal functions are defined such as:

$$H_{ij}(k_x, k_y) = G_i(\alpha_k) F_j(\alpha_\phi) \eta(k_x, k_y),$$

where η writes as:

$$\eta(k_x, k_y) = \left(\frac{2(a_2 k_x^2 + 2a_1 k_x^4 + k_y^2)}{(k_x^2 + k_y^2)(a_2 k_x^2 + a_1 k_x^4 + k_y^2)(\log k_{\text{max}} - \log k_{\text{min}})} \right)^2 , with$$

$$\gamma = 2$$

$$a_1 = \frac{(\gamma^2$$

$$(\gamma^2 * k_{\text{m}}^2$$

$$a_2 = \frac{k_{\text{max}}^2 - }{k_{\text{max}}^2 - }$$

In this study, $k_{\text{min}} = 2\pi/600$ and $k_{\text{max}} = 2\pi/25$ to take benefit of the improved resolution and size of Sentinel-1 SAR images.

$G_i(\alpha_k(k_x, k_y))$ writes :

$$G_i^{(\lambda)}(x) = \frac{1}{i} \left(2x(i + \lambda - 1) G_{i-1}^{(\lambda)}(x) - (i + 2\lambda - 2) G_{i-2}^{(\lambda)}(x) \right), \text{ for } i \geq 2. \tag{14}$$

Otherwise:

$$C_0^{(\lambda)}(x) = 1 \tag{15}$$

$$C_1^{(\lambda)}(x) = 2\lambda x \tag{16}$$

In this study λ is set to $3/2$. $F_j(\alpha_\phi(k_x, k_y))$ writes :

$$F_j(x) = \sqrt{2/\pi} \sin\left(jx\right), \text{ for n>1, when i is even} \tag{17}$$

$$F_j(x) = \sqrt{2/\pi} \sin\left((j-1)x\right), \text{ for n>1, when i is odd} \tag{18}$$

otherwise:

$$F_1(x) = \sqrt{2/\pi} \tag{19}$$

Finally α_k and α_ϕ write:

$$\alpha_k = 2 \frac{\log\left(\sqrt{a_1 k_x^4 + a_2 k_x^2 + k_y^2}\right) - \log(k_{\text{min}})}{\log(k_{\text{max}}) - \log(k_{\text{min}})} - 1 \tag{20}$$

$$\alpha_\phi = \arctan(k_x, k_y) \tag{21}$$

Computation of MACS parameters

In Ifremer SARWAVE Level-1B XSP processor the computation of MACS parameters (real and imaginary part) is a direct implementation of Li *et al.* [2019].

Last documentation build: Feb 16 2024 at 14:59

0.2.3 Level-1B SAR Ifremer Product Description

This page stands as a Product Description document for Sentinel-1 Level-1B Ifremer product.

It describes the format, the files and the content of Level-1B SAR product.

Product philosophy

The Level-1B Sentinel-1 Ifremer product is designed to be a SAR expert product. The rationales behind the dimensions and coordinates that are showing the tiling of the SAR image in slice / sub-swath / burst / tiles are:

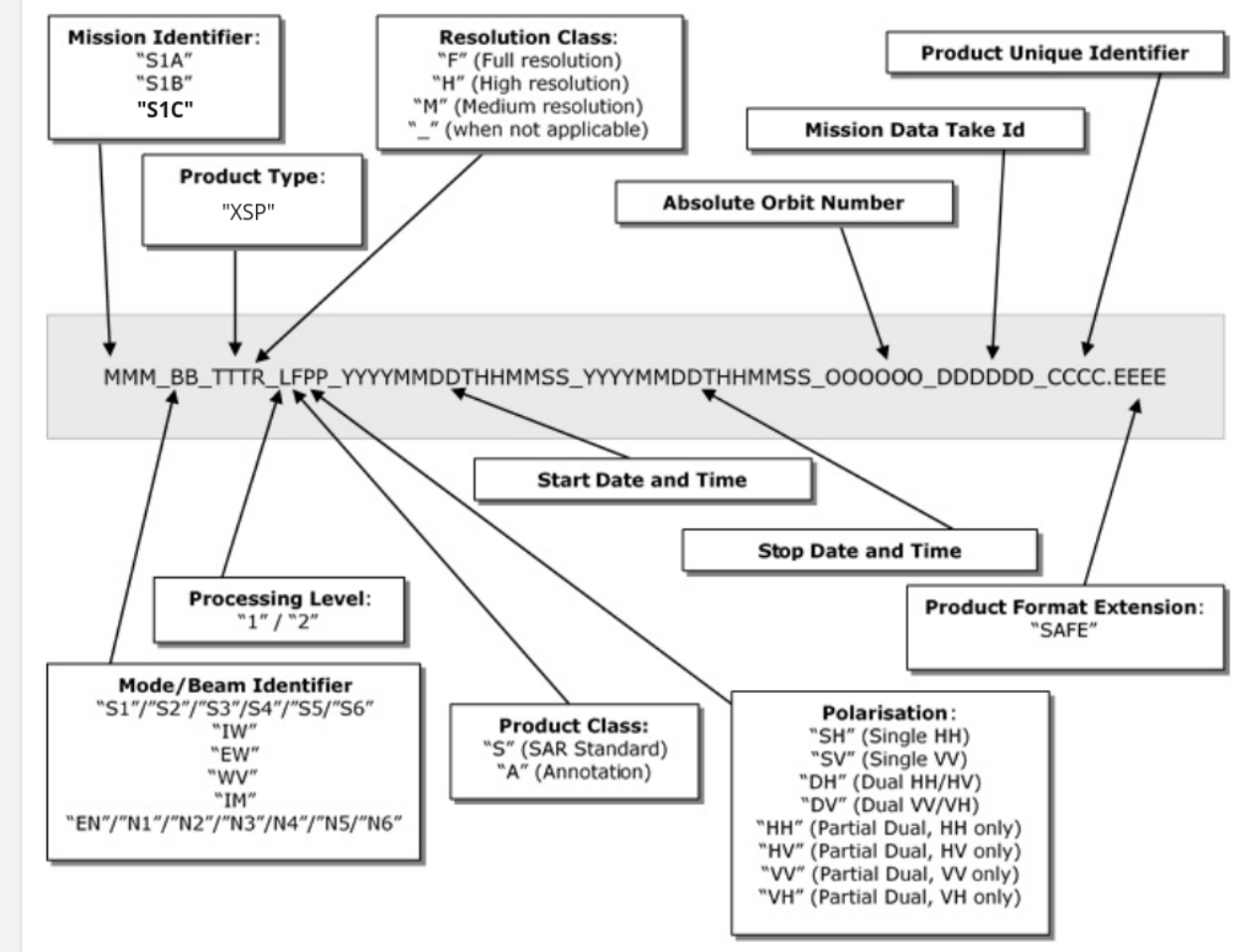
> The willing to stay close to image geometry to be able to easily swap from image domain to geo-referenced spectral domain
>
> The willing to stay close to ESA Sentinel-1 SLC product format.

Level-1B Sentinel-1 Ifremer product has been designed to create an intermediate product in between the Level-1 SLC and a Level-2 OCN. This Level-1B product eases the work on the inversion from SAR image cross-spectrum coming from intra-burst and inter-burst (overlapping) part separating the image processing steps from geophysical inversion. The steps performed to go from Level-1 SLC to Level-1B XSP are time/memory consuming operations such as: High resolution I/O or 2D-FFT.

The current version of the product is still prototype, and future evolution are expected. The configurations of the tiles, periodograms, looks is tuneable and furthers tests will help to choose the proper set of parameters for wind and wave applications.

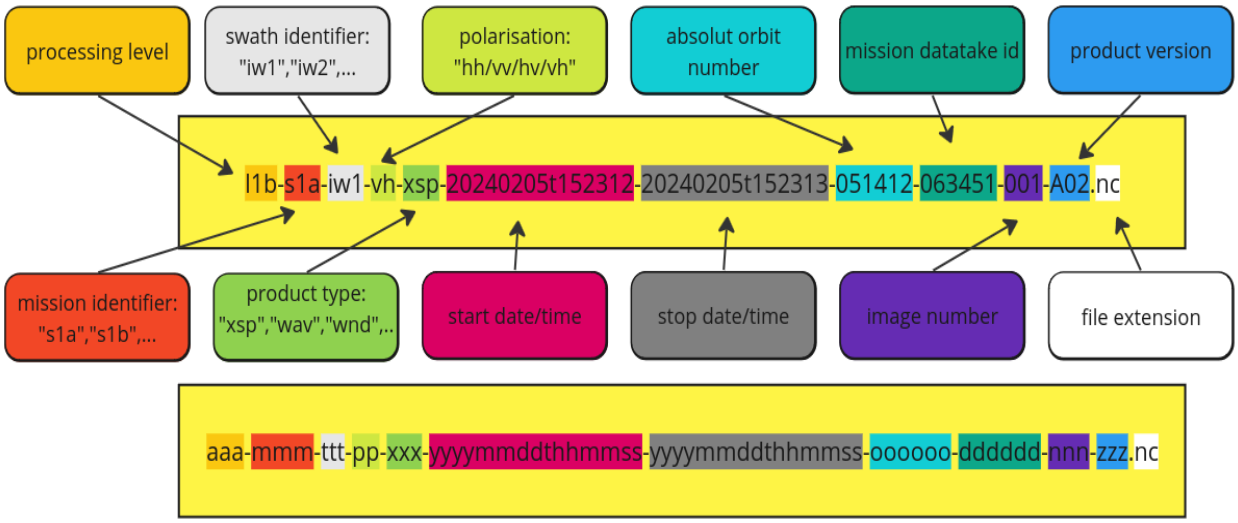
Product structure

Each product is stored in a .SAFE directory. The SAFE convention is inherited from Sentinel-1 mission. Except that the SLC (single look complex) acronym has been replaced by XSP (for cross-spectrum which is the most important variable in this dataset).



Each .SAFE directory contains 3 or 6 netCDF files, one per polarization (could be single polarization HH/VV or dual polarization HH/HV - VV/VH) plus one per sub-swath, following the official ESA SLC products storage convention. All the polarizations are processed from Level-1 to Level-1B even though VV is the classical candidate polarization for wave inversion.

The netCDF files naming convention is adapted from ESA Sentinel-1:



Example of Level-1B measurement filename:

l1b-s1a-iw1-vh-xsp-20240205t152312-20240205t152313-051412-063451-001-A02.nc

where

- *l1b* stands for the processing level
- *s1a* gives the SAR unit name
- *iw1* gives the acquisition mode and the subswath number (up to 3 for IW and up to 5 for EW)
- *vh* is the polarisation of the acquisition: transmit in V and received in H in this example
- *xsp* is the 3-character defining the sub-product that can be found in the file
- *20240205t152312-20240205t152313* are respectively start and stop acquisition dates
- *051412* is the absolute orbit number
- *063451* is the mission datatake ID
- *001* is the image number (ie measurement) in the SAFE directory
- *A02* is a 3-character code that allow to track the processor used to generate the file, its version, and the processing options

Note: Typical size for a IW SLC processed at 17.5 km^2 tile width is about 75 Mb per Level-1B .nc file .

Product variables

Exhaustive content

Each netCDF files is split into 2 groups:

- **intraburst**: region of SLC bursts in which Ocean/Land surface have been seen only one time.
- **interburst**: region of SLC bursts for which the Ocean/Land surface have been observed twice thanks to TOPS antenna steering capability.

Each of the group contains the same set of variables with minor exceptions*.

The variables of **intraburst** group:

```
group: intraburst {
dimensions:
      tile_line = 10 ;
      tile_sample = 4 ;
      bt_thresh = 5 ;
      freq_sample = 403 ;
      freq_line = 50 ;
      \0tau = 3 ;
      \1tau = 2 ;
      \2tau = 1 ;
      c_sample = 2 ;
      c_line = 2 ;
      k_gp = 4 ;
      phi_hf = 5 ;
      lambda_range_max_macs = 10 ;
```

```
variables:
       float incidence(tile_line, tile_sample) ;
               incidence:_FillValue = NaNf ;
               incidence:long_name = "incidence at tile middle" ;
               incidence:units = "degree" ;
               incidence:coordinates = "latitude line longitude pol sample" ;
       float ground_heading(tile_line, tile_sample) ;
               ground_heading:_FillValue = NaNf ;
               ground_heading:long_name = "ground heading" ;
               ground_heading:units = "degree" ;
               ground_heading:convention = "from North clockwise" ;
               ground_heading:coordinates = "latitude line longitude pol sample" ;
       string pol ;
       short burst(tile_line) ;
               burst:coordinates = "line pol" ;
       int64 sensing_time(tile_line, tile_sample) ;
               sensing_time:long_name = "tile sensing time" ;
               sensing_time:coordinates = "latitude line longitude pol sample" ;
               sensing_time:units = "microseconds since 2021-04-18 21:22:25.142172" ;
               sensing_time:calendar = "proleptic_gregorian" ;
       float sigma0(tile_line, tile_sample) ;
               sigma0:_FillValue = NaNf ;
               sigma0:long_name = "RAW calibrated sigma0" ;
               sigma0:units = "linear" ;
               sigma0:coordinates = "latitude line longitude pol sample" ;
       float nesz(tile_line, tile_sample) ;
               nesz:_FillValue = NaNf ;
               nesz:long_name = "RAW noise-equivalent sigma zero" ;
               nesz:units = "linear" ;
               nesz:coordinates = "latitude line longitude pol sample" ;
       short bt_thresh(bt_thresh) ;
               bt_thresh:long_name = "lower edge of bright target to background␣
↪amplitude ratio" ;
       short bright_targets_histogram(tile_line, tile_sample, bt_thresh) ;
               bright_targets_histogram:long_name = "bright targets histogram" ;
               bright_targets_histogram:coordinates = "latitude line longitude pol␣
↪sample" ;
       float sigma0_filt(tile_line, tile_sample) ;
               sigma0_filt:_FillValue = NaNf ;
               sigma0_filt:long_name = "calibrated sigma0 with BT correction" ;
               sigma0_filt:units = "linear" ;
               sigma0_filt:coordinates = "latitude line longitude pol sample" ;
       float normalized_variance_filt(tile_line, tile_sample) ;
               normalized_variance_filt:_FillValue = NaNf ;
               normalized_variance_filt:long_name = "normalized variance with BT␣
↪correction" ;
               normalized_variance_filt:units = "" ;
               normalized_variance_filt:coordinates = "latitude line longitude pol␣
↪sample" ;
       float doppler_centroid(tile_line, tile_sample) ;
               doppler_centroid:_FillValue = NaNf ;
               doppler_centroid:units = "rad/m" ;
               doppler_centroid:averaged_periodograms = 81LL ;
               doppler_centroid:periodo_width_sample = 3540LL ;
               doppler_centroid:periodo_overlap_sample = 1770LL ;
               doppler_centroid:periodo_overlap_line = 1770LL ;
               doppler_centroid:coordinates = "latitude line longitude pol sample" ;
```

(continues on next page)

(continued from previous page)

```
      float k_rg(tile_line, tile_sample, freq_sample) ;
              k_rg:_FillValue = NaNf ;
              k_rg:long_name = "wavenumber in range direction" ;
              k_rg:units = "rad/m" ;
      float k_az(freq_line) ;
              k_az:_FillValue = NaNf ;
              k_az:long_name = "wavenumber in azimuth direction" ;
              k_az:units = "rad/m" ;
              k_az:spacing = 0.00179264256685252 ;
      float var_xspectra_0tau(tile_line, tile_sample, freq_line, freq_sample, \0tau)␣
↪;
              var_xspectra_0tau:_FillValue = NaNf ;
              var_xspectra_0tau:averaged_periodograms = 81LL ;
              var_xspectra_0tau:periodo_width_sample = 3540LL ;
              var_xspectra_0tau:periodo_width_line = 3540LL ;
              var_xspectra_0tau:periodo_overlap_sample = 1770LL ;
              var_xspectra_0tau:periodo_overlap_line = 1770LL ;
              var_xspectra_0tau:coordinates = "k_az k_rg latitude line longitude pol␣
↪sample" ;
      float var_xspectra_1tau(tile_line, tile_sample, freq_line, freq_sample, \1tau)␣
↪;
              var_xspectra_1tau:_FillValue = NaNf ;
              var_xspectra_1tau:averaged_periodograms = 81LL ;
              var_xspectra_1tau:periodo_width_sample = 3540LL ;
              var_xspectra_1tau:periodo_width_line = 3540LL ;
              var_xspectra_1tau:periodo_overlap_sample = 1770LL ;
              var_xspectra_1tau:periodo_overlap_line = 1770LL ;
              var_xspectra_1tau:coordinates = "k_az k_rg latitude line longitude pol␣
↪sample" ;
      float var_xspectra_2tau(tile_line, tile_sample, freq_line, freq_sample, \2tau)␣
↪;
              var_xspectra_2tau:_FillValue = NaNf ;
              var_xspectra_2tau:averaged_periodograms = 81LL ;
              var_xspectra_2tau:periodo_width_sample = 3540LL ;
              var_xspectra_2tau:periodo_width_line = 3540LL ;
              var_xspectra_2tau:periodo_overlap_sample = 1770LL ;
              var_xspectra_2tau:periodo_overlap_line = 1770LL ;
              var_xspectra_2tau:coordinates = "k_az k_rg latitude line longitude pol␣
↪sample" ;
      float tau(tile_line, tile_sample) ;
              tau:_FillValue = NaNf ;
              tau:long_name = "delay between two successive looks" ;
              tau:units = "s" ;
              tau:coordinates = "latitude line longitude pol sample" ;
      float azimuth_cutoff(tile_line, tile_sample) ;
              azimuth_cutoff:_FillValue = NaNf ;
              azimuth_cutoff:long_name = "Azimuthal cut-off (2tau)" ;
              azimuth_cutoff:units = "m" ;
              azimuth_cutoff:coordinates = "latitude line longitude pol sample" ;
      float azimuth_cutoff_error(tile_line, tile_sample) ;
              azimuth_cutoff_error:_FillValue = NaNf ;
              azimuth_cutoff_error:long_name = "normalized azimuthal cut-off error␣
↪std (2tau)" ;
              azimuth_cutoff_error:coordinates = "latitude line longitude pol sample
↪" ;
      short line(tile_line) ;
      short sample(tile_line, tile_sample) ;
```

(continues on next page)

(continued from previous page)

```
      float corner_longitude(tile_line, tile_sample, c_sample, c_line) ;
              corner_longitude:_FillValue = NaNf ;
              corner_longitude:history = "longitude:\n  annotation/s1a.xml:\n  - /
↪product/geolocationGrid/geolocationGridPointList/geolocationGridPoint/line\n  - /
↪product/geolocationGrid/geolocationGridPointList/geolocationGridPoint/pixel\n  - /
↪product/geolocationGrid/geolocationGridPointList/geolocationGridPoint/longitude\n" ;
              corner_longitude:definition = "Geodetic longitude of grid point␣
↪[degrees]." ;
              corner_longitude:coordinates = "latitude line longitude pol sample" ;
      float corner_latitude(tile_line, tile_sample, c_sample, c_line) ;
              corner_latitude:_FillValue = NaNf ;
              corner_latitude:history = "latitude:\n  annotation/s1a.xml:\n  - /
↪product/geolocationGrid/geolocationGridPointList/geolocationGridPoint/line\n  - /
↪product/geolocationGrid/geolocationGridPointList/geolocationGridPoint/pixel\n  - /
↪product/geolocationGrid/geolocationGridPointList/geolocationGridPoint/latitude\n" ;
              corner_latitude:definition = "Geodetic latitude of grid point␣
↪[degrees]." ;
              corner_latitude:coordinates = "latitude line longitude pol sample" ;
      short corner_line(tile_line, c_line) ;
              corner_line:long_name = "line number in original digital number matrix
↪" ;
              corner_line:coordinates = "line pol" ;
      short corner_sample(tile_line, tile_sample, c_sample) ;
              corner_sample:long_name = "sample number in original digital number␣
↪matrix" ;
              corner_sample:coordinates = "latitude line longitude pol sample" ;
      float longitude(tile_line, tile_sample) ;
              longitude:_FillValue = NaNf ;
              longitude:history = "longitude:\n  annotation/s1a.xml:\n  - /product/
↪geolocationGrid/geolocationGridPointList/geolocationGridPoint/line\n  - /product/
↪geolocationGrid/geolocationGridPointList/geolocationGridPoint/pixel\n  - /product/
↪geolocationGrid/geolocationGridPointList/geolocationGridPoint/longitude\n" ;
              longitude:definition = "Geodetic longitude of grid point [degrees]." ;
      float latitude(tile_line, tile_sample) ;
              latitude:_FillValue = NaNf ;
              latitude:history = "latitude:\n  annotation/s1a.xml:\n  - /product/
↪geolocationGrid/geolocationGridPointList/geolocationGridPoint/line\n  - /product/
↪geolocationGrid/geolocationGridPointList/geolocationGridPoint/pixel\n  - /product/
↪geolocationGrid/geolocationGridPointList/geolocationGridPoint/latitude\n" ;
              latitude:definition = "Geodetic latitude of grid point [degrees]." ;
      byte land_flag(tile_line, tile_sample) ;
              land_flag:long_name = "land flag" ;
              land_flag:convention = "True if land is present" ;
              land_flag:coordinates = "latitude line longitude pol sample" ;
              land_flag:dtype = "bool" ;
      float burst_corner_longitude(tile_line, c_sample, c_line) ;
              burst_corner_longitude:_FillValue = NaNf ;
              burst_corner_longitude:long_name = "corner longitude of burst valid␣
↪portion" ;
              burst_corner_longitude:coordinates = "line pol" ;
      float burst_corner_latitude(tile_line, c_sample, c_line) ;
              burst_corner_latitude:_FillValue = NaNf ;
              burst_corner_latitude:long_name = "corner latitude of burst valid␣
↪portion" ;
              burst_corner_latitude:coordinates = "line pol" ;
      short k_gp(k_gp) ;
              k_gp:long_name = "Gegenbauer polynoms dimension" ;
```

(continues on next page)

(continued from previous page)

```
      short phi_hf(phi_hf) ;
              phi_hf:long_name = "Harmonic functions dimension (odd number)" ;
      float cwave_params(tile_line, tile_sample, k_gp, phi_hf, \2tau) ;
              cwave_params:_FillValue = NaNf ;
              cwave_params:long_name = "CWAVE parameters" ;
              cwave_params:coordinates = "latitude line longitude pol sample" ;
      float lambda_range_max_macs(lambda_range_max_macs) ;
              lambda_range_max_macs:_FillValue = NaNf ;
              lambda_range_max_macs:long_name = "maximum wavelength bound for MACS␣
↪estimation " ;
      float xspectra_0tau_Re(tile_line, tile_sample, freq_line, freq_sample, \0tau) ;
              xspectra_0tau_Re:_FillValue = NaNf ;
              xspectra_0tau_Re:averaged_periodograms = 81LL ;
              xspectra_0tau_Re:periodo_width_sample = 3540LL ;
              xspectra_0tau_Re:periodo_width_line = 3540LL ;
              xspectra_0tau_Re:periodo_overlap_sample = 1770LL ;
              xspectra_0tau_Re:periodo_overlap_line = 1770LL ;
              xspectra_0tau_Re:coordinates = "k_az k_rg latitude line longitude pol␣
↪sample" ;
      float xspectra_0tau_Im(tile_line, tile_sample, freq_line, freq_sample, \0tau) ;
              xspectra_0tau_Im:_FillValue = NaNf ;
              xspectra_0tau_Im:averaged_periodograms = 81LL ;
              xspectra_0tau_Im:periodo_width_sample = 3540LL ;
              xspectra_0tau_Im:periodo_width_line = 3540LL ;
              xspectra_0tau_Im:periodo_overlap_sample = 1770LL ;
              xspectra_0tau_Im:periodo_overlap_line = 1770LL ;
              xspectra_0tau_Im:coordinates = "k_az k_rg latitude line longitude pol␣
↪sample" ;
      float xspectra_1tau_Re(tile_line, tile_sample, freq_line, freq_sample, \1tau) ;
              xspectra_1tau_Re:_FillValue = NaNf ;
              xspectra_1tau_Re:averaged_periodograms = 81LL ;
              xspectra_1tau_Re:periodo_width_sample = 3540LL ;
              xspectra_1tau_Re:periodo_width_line = 3540LL ;
              xspectra_1tau_Re:periodo_overlap_sample = 1770LL ;
              xspectra_1tau_Re:periodo_overlap_line = 1770LL ;
              xspectra_1tau_Re:coordinates = "k_az k_rg latitude line longitude pol␣
↪sample" ;
      float xspectra_1tau_Im(tile_line, tile_sample, freq_line, freq_sample, \1tau) ;
              xspectra_1tau_Im:_FillValue = NaNf ;
              xspectra_1tau_Im:averaged_periodograms = 81LL ;
              xspectra_1tau_Im:periodo_width_sample = 3540LL ;
              xspectra_1tau_Im:periodo_width_line = 3540LL ;
              xspectra_1tau_Im:periodo_overlap_sample = 1770LL ;
              xspectra_1tau_Im:periodo_overlap_line = 1770LL ;
              xspectra_1tau_Im:coordinates = "k_az k_rg latitude line longitude pol␣
↪sample" ;
      float xspectra_2tau_Re(tile_line, tile_sample, freq_line, freq_sample, \2tau) ;
              xspectra_2tau_Re:_FillValue = NaNf ;
              xspectra_2tau_Re:averaged_periodograms = 81LL ;
              xspectra_2tau_Re:periodo_width_sample = 3540LL ;
              xspectra_2tau_Re:periodo_width_line = 3540LL ;
              xspectra_2tau_Re:periodo_overlap_sample = 1770LL ;
              xspectra_2tau_Re:periodo_overlap_line = 1770LL ;
              xspectra_2tau_Re:coordinates = "k_az k_rg latitude line longitude pol␣
↪sample" ;
      float xspectra_2tau_Im(tile_line, tile_sample, freq_line, freq_sample, \2tau) ;
              xspectra_2tau_Im:_FillValue = NaNf ;
```

(continues on next page)

(continued from previous page)

```
                xspectra_2tau_Im:averaged_periodograms = 81LL ;
                xspectra_2tau_Im:periodo_width_sample = 3540LL ;
                xspectra_2tau_Im:periodo_width_line = 3540LL ;
                xspectra_2tau_Im:periodo_overlap_sample = 1770LL ;
                xspectra_2tau_Im:periodo_overlap_line = 1770LL ;
                xspectra_2tau_Im:coordinates = "k_az k_rg latitude line longitude pol␣
↪sample" ;
        float macs_Re(tile_line, tile_sample, lambda_range_max_macs, \2tau) ;
                macs_Re:_FillValue = NaNf ;
                macs_Re:kaz_min = -0.010471975511966 ;
                macs_Re:kaz_max = 0.010471975511966 ;
                macs_Re:krg_max = 0.418879020478639 ;
                macs_Re:long_name = "Mean rAnge Cross-Spectrum" ;
                macs_Re:coordinates = "latitude line longitude pol sample" ;
        float macs_Im(tile_line, tile_sample, lambda_range_max_macs, \2tau) ;
                macs_Im:_FillValue = NaNf ;
                macs_Im:kaz_min = -0.010471975511966 ;
                macs_Im:kaz_max = 0.010471975511966 ;
                macs_Im:krg_max = 0.418879020478639 ;
                macs_Im:long_name = "Mean rAnge Cross-Spectrum" ;
                macs_Im:coordinates = "latitude line longitude pol sample" ;
```

Download this notebook from github[10].

Sentinel-1 Level-1B IFREMER variables explanation

Although some variables need extra explanation about why there are present, how they are computed and how to properly use them.

A dedicated Python library is available to use IFREMER Sentinel-1 Level-1B products: https://slcl1butils.readthedocs.io

[1]:
```
from xsarslc.get_test_files import get_test_file
import xarray as xr
# https://cyclobs.ifremer.fr/static/sarwing_datarmor/xsardata/S1B_IW_XSP__1SDV_
↪20200925T173737_20200925T173804_023535_02CB5D_E473.SAFE/s1b-iw1-slc-vv-
↪20200925t173739-20200925t173804-023535-02cb5d-004_L1B_xspec_IFR_1.4k.nc.zip
#localpath = get_test_file('S1B_IW_XSP__1SDV_20200925T173737_20200925T173804_023535_
↪02CB5D_E473.SAFE/s1b-iw1-slc-vv-20200925t173739-20200925t173804-023535-02cb5d-004_
↪L1B_xspec_IFR_1.4k.nc')
localpath = get_test_file('s1b-iw1-slc-vv-20200925t173739-20200925t173804-023535-
↪02cb5d-004_L1B_xspec_IFR_1.4k.nc')
localpath
```

[1]:
```
'/tmp/s1b-iw1-slc-vv-20200925t173739-20200925t173804-023535-02cb5d-004_L1B_xspec_IFR_
↪1.4k.nc'
```

[2]:
```
import os
#localpath_tweaked = os.path.join(os.path.dirname(os.path.dirname(localpath)),os.path.
↪basename(localpath))
#print(localpath_tweaked,os.path.exists(localpath_tweaked))
print(os.path.exists(localpath))
ds = xr.open_dataset(localpath,group='intraburst')
ds
```

[10] https://github.com/umr-lops/xsar_slc/tree/main/docs/examples/L1B_Sentinel1_variables_explanation.ipynb

```
True
```

[2]:
```
<xarray.Dataset>
Dimensions:                  (burst: 9, tile_line: 1, tile_sample: 4,
                              freq_sample: 401, freq_line: 51, 0tau: 3, 1tau: 2,
                              2tau: 1, c_sample: 2, c_line: 2)
Coordinates:
    pol                      <U2 …
  * burst                    (burst) int16 0 1 2 3 4 5 6 7 8
    k_rg                     (burst, tile_sample, freq_sample) float32 …
    k_az                     (freq_line) float32 …
    line                     (burst, tile_line) int16 …
    sample                   (burst, tile_sample) int16 …
    longitude                (burst, tile_line, tile_sample) float32 …
    latitude                 (burst, tile_line, tile_sample) float32 …
Dimensions without coordinates: tile_line, tile_sample, freq_sample, freq_line,
                                0tau, 1tau, 2tau, c_sample, c_line
Data variables: (12/25)
    incidence                (burst, tile_line, tile_sample) float32 …
    normalized_variance      (burst, tile_line, tile_sample) float32 …
    sigma0                   (burst, tile_line, tile_sample) float32 …
    nesz                     (burst, tile_line, tile_sample) float32 …
    ground_heading           (burst, tile_line, tile_sample) float32 …
    doppler_centroid         (burst, tile_line, tile_sample) float32 …
    …                       …
    xspectra_0tau_Re         (burst, tile_line, tile_sample, freq_line, freq_sample,␣
↪0tau) float32 …
    xspectra_0tau_Im         (burst, tile_line, tile_sample, freq_line, freq_sample,␣
↪0tau) float32 …
    xspectra_1tau_Re         (burst, tile_line, tile_sample, freq_line, freq_sample,␣
↪1tau) float32 …
    xspectra_1tau_Im         (burst, tile_line, tile_sample, freq_line, freq_sample,␣
↪1tau) float32 …
    xspectra_2tau_Re         (burst, tile_line, tile_sample, freq_line, freq_sample,␣
↪2tau) float32 …
    xspectra_2tau_Im         (burst, tile_line, tile_sample, freq_line, freq_sample,␣
↪2tau) float32 …
Attributes: (12/27)
    name:                     SENTINEL1_DS:/home/datawork-cersat-public/projec…
    short_name:               SENTINEL1_DS:S1B_IW_SLC__1SDV_20200925T173737_20…
    product:                  SLC
    safe:                     S1B_IW_SLC__1SDV_20200925T173737_20200925T173804…
    swath:                    IW
    multidataset:             False
    …                        …
    tile_overlap_sample:      0
    tile_overlap_line:        0
    periodo_width_sample:     3540
    periodo_width_line:       3540
    periodo_overlap_sample:   1770
    periodo_overlap_line:     1770
```

wavenumbers in range axis: “k_rg”

“k_rg” variable slightly depends on each tile because the pixel spacing is not the same in the IW sub-swath.

We made the choice to have a constant dk (in range and azimuth) for all the tiles/cross-spectrum, and we achieved a 10e-3 consistency as a maximum point-to-point difference wavenumbers .

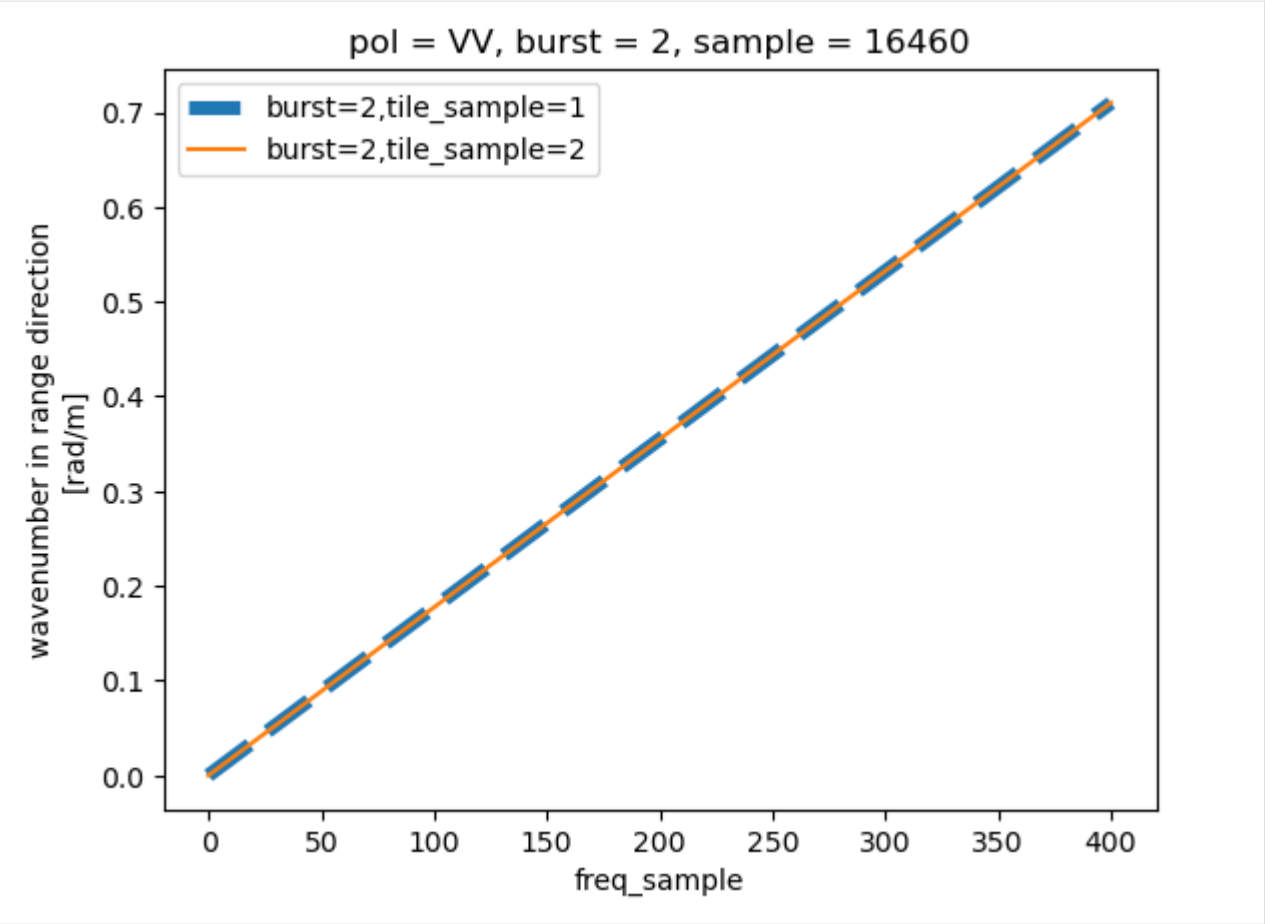
This strategy allows to compare a spectra to another.

[3]:
```
k_rg1 = ds['k_rg'].isel(burst=2,tile_sample=1)
k_rg2 = ds['k_rg'].isel(burst=2,tile_sample=3)
k_rg1
```

[3]:
```
<xarray.DataArray 'k_rg' (freq_sample: 401)>
[401 values with dtype=float32]
Coordinates:
    pol      <U2 …
    burst    int16 2
    k_rg     (freq_sample) float32 …
    sample   int16 …
Dimensions without coordinates: freq_sample
Attributes:
    long_name:  wavenumber in range direction
    units:      rad/m
```

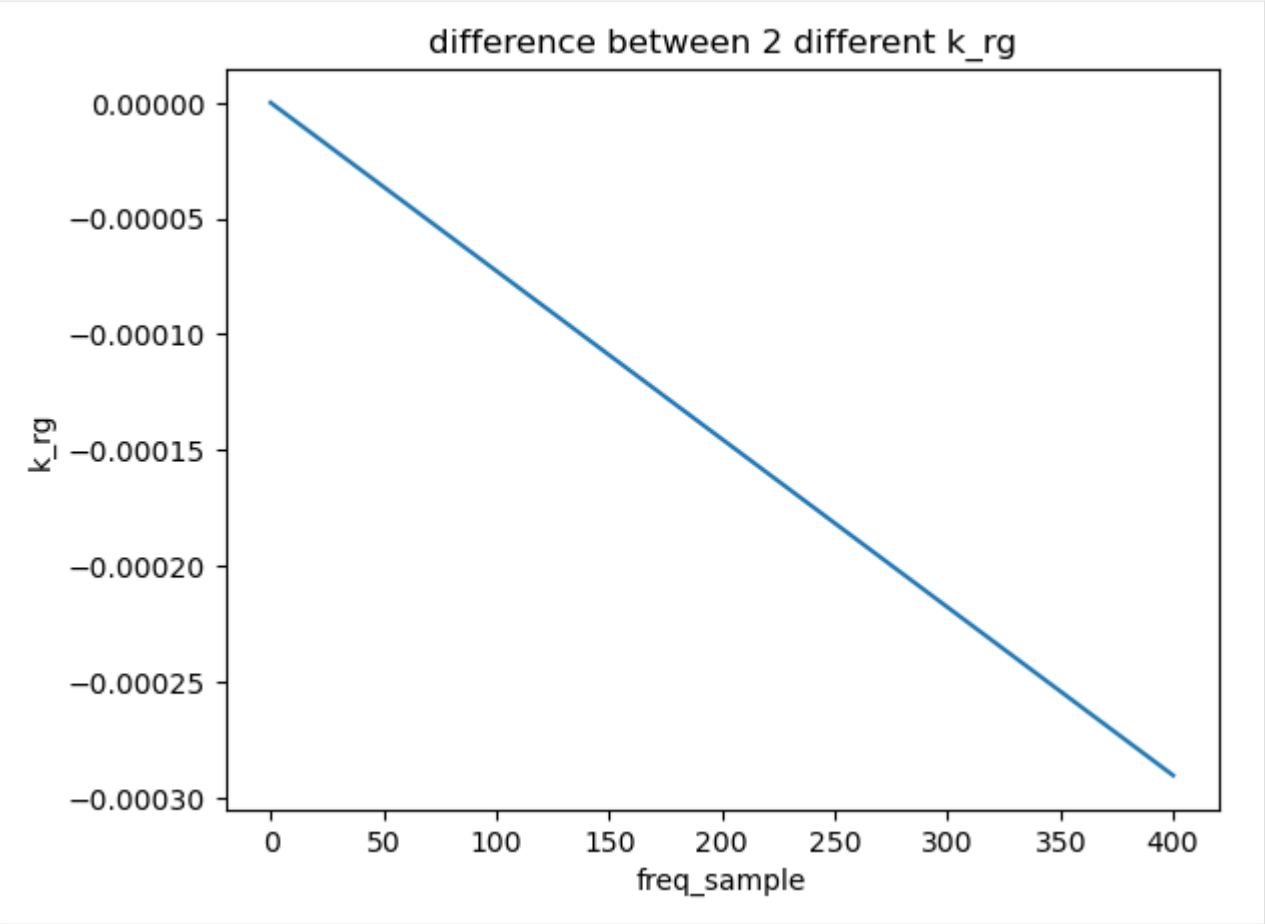
[4]:
```
from matplotlib import pyplot as plt
k_rg1.plot(label="burst=2,tile_sample=1",linestyle='--',lw=5)
k_rg2.plot(label="burst=2,tile_sample=2")
plt.legend()
```

[4]:
```
<matplotlib.legend.Legend at 0x7f0ceb09a950>
```



[5]:
```
(k_rg1-k_rg2).plot()
plt.title('difference between 2 different k_rg')
```

[5]:
```
Text(0.5, 1.0, 'difference between 2 different k_rg')
```



The difference between the 2 vectors of wave numbers is super small but for users who might want to stack the different cross spectrum, a first step would be to assign a single k_rg to all the cross spectrum.

SAR image complexe cartesian cross spectrum

The "xpsectra_{0-1-2}tau_*" is split in real and imaginary part because netCDF4 file format does not allow to store complex values.

Also only half of the spectrum along the range wavenumbers is stored in the netCDF files to save space.

[6]:
```
# recompose complexe SAr image cross spectrum from real and imaginary part
for tautau in range(3):
    ds['xspectra_%stau'%tautau] = ds['xspectra_%stau_Re'%tautau] + 1j*ds['xspectra_
↪%stau_Im'%tautau]
    #ds = ds.drop(['xspectra_%stau_Re'%tautau,'xspectra_%stau_Im'%tautau])
one_cartesian_xsp = ds['xspectra_2tau'].isel({'burst':6,'tile_line':0,'tile_sample':2,
↪'2tau':0})
one_cartesian_xsp = one_cartesian_xsp.assign_coords({'freq_line':one_cartesian_xsp.k_
↪az,'freq_sample':one_cartesian_xsp.k_rg})
one_cartesian_xsp
```

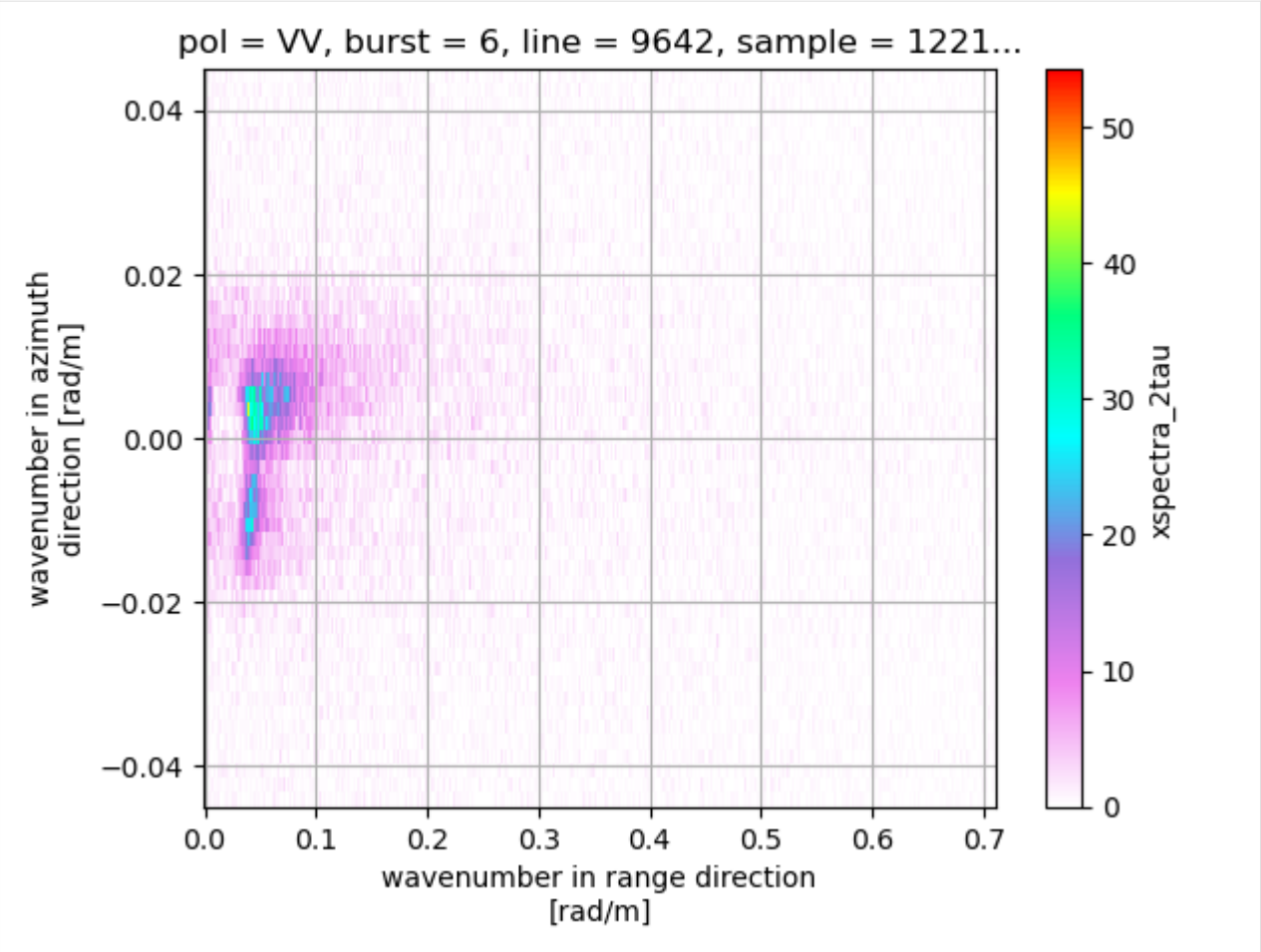
[6]:
```
<xarray.DataArray 'xspectra_2tau' (freq_line: 51, freq_sample: 401)>
array([[-1.2477797 +0.5443525j ,  1.5725    -0.01272768j,
         0.4407456 -0.2429612j , …,  0.00830453-0.3919062j ,
```

(continues on next page)

(continued from previous page)

```
         0.1216751 +0.09071319j, -0.05311767+0.49395174j],
       [ 2.060735  +0.5237646j ,  0.2005456 +0.29838213j,
        -0.02560596-0.18570837j, …,  0.05239271+0.30355382j,
         0.54826015-0.08569147j,  0.02144801-0.19576392j],
       [-1.4342278 +1.4327244j , -0.30036288+0.18255554j,
         0.6784638 -0.17505352j, …, -0.04385818+0.16508722j,
        -0.375506  -0.16829075j, -0.24914488+0.21455552j],
       …,
       [-1.4342278 -1.4327244j , -0.53528905+0.7241161j ,
        -1.539189  +0.64669746j, …,  0.03286539+0.10321924j,
        -0.2948199 -0.2361605j , -0.47778353-0.54428077j],
       [ 2.060735  -0.5237646j , -0.2049799 -1.0952276j ,
         0.42247522+1.1954173j , …,  0.15609382-0.17570984j,
         0.36818713-0.3684804j , -0.01811259-0.32531738j],
       [-1.2477797 -0.5443525j , -0.39482182+1.2361256j ,
        -0.9189708 +0.479853j  , …,  0.360627  +0.5266922j ,
         0.07324556+0.2173959j ,  0.4194936 +0.03121182j]],
      dtype=complex64)
Coordinates:
    pol          <U2 'VV'
    burst        int16 6
    k_rg         (freq_sample) float32 …
    k_az         (freq_line) float32 …
    line         int16 9642
    sample       int16 12215
    longitude    float32 3.652
    latitude     float32 40.42
  * freq_line    (freq_line) float32 -0.04414 -0.04238 … 0.04238 0.04414
  * freq_sample  (freq_sample) float32 0.0 0.001775 0.00355 … 0.7082 0.71
```

[7]:
```
from matplotlib import colors as mcolors
cmap = mcolors.LinearSegmentedColormap.from_list("", ["white","violet","mediumpurple",
↪"cyan","springgreen","yellow","red"])

abs(one_cartesian_xsp.real).plot(cmap=cmap)
plt.grid(True)
```



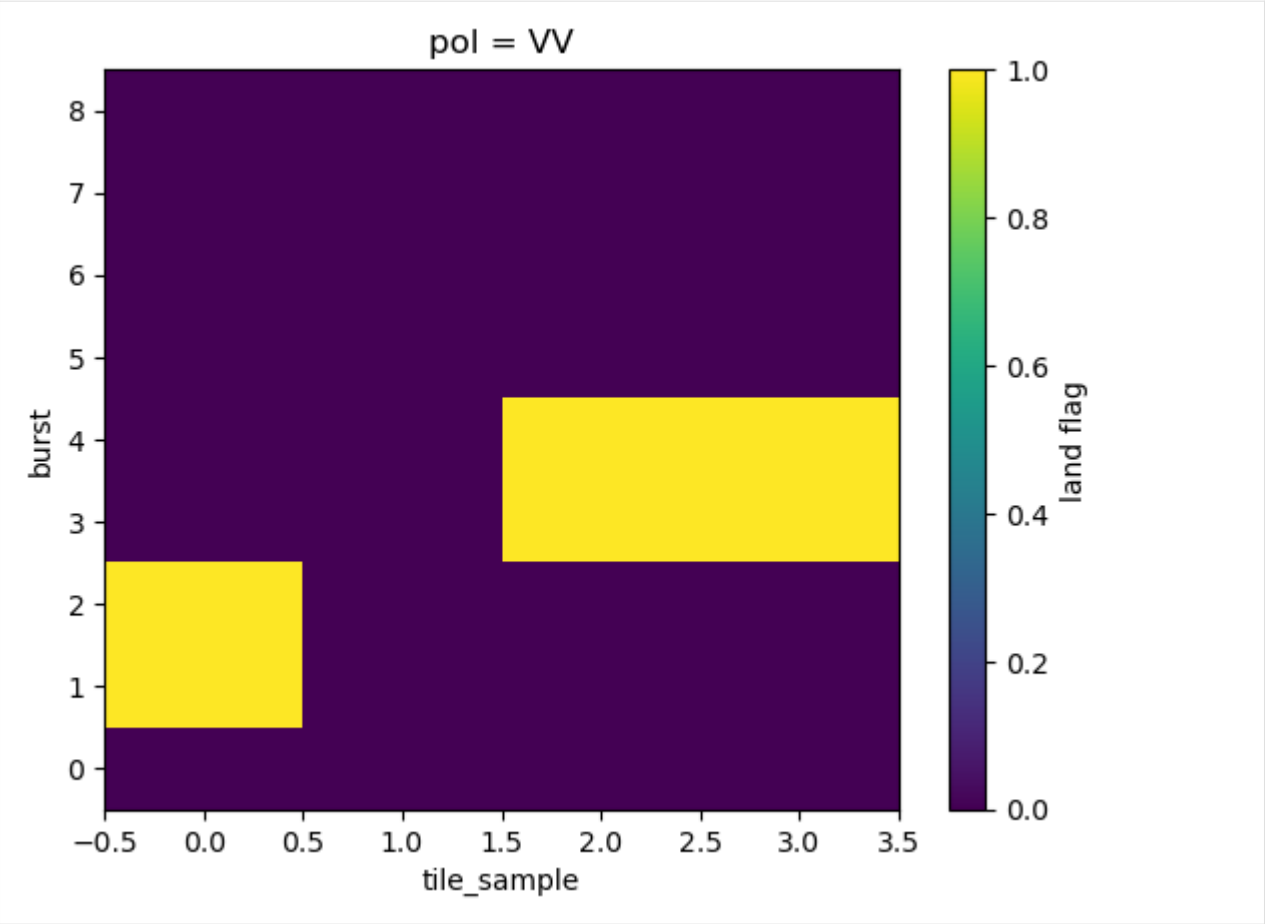
landmask

The “land_flag” annotated in the sample of L1B product comes from `cartopy` 10 m resolution polygons.

cartopy/shapefiles/natural_earth/physical/: https://www.naturalearthdata.com/

[8]:
```
ds['land_flag'].plot()
```

[8]:
```
<matplotlib.collections.QuadMesh at 0x7f0c30becbb0>
```



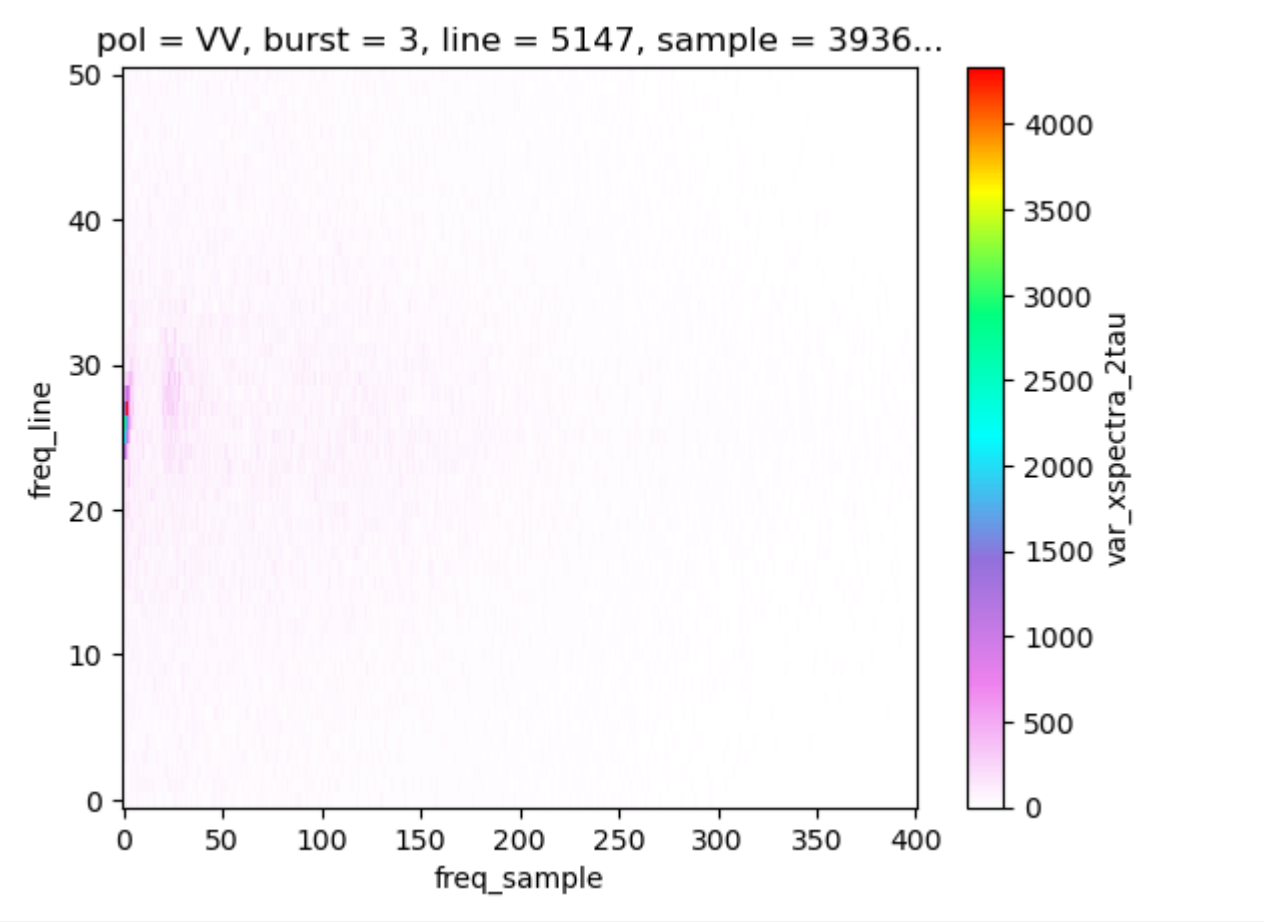
variance of cross spectrum

"var_xspectra_{0-1-2}tau" is the variance of the different periodograms while "xpsectra_{0-1-2}tau" is the mean of the periodograms computed within a given tile.

These variables **var_xspectra_{0-1-2}tau** are defined on the same grid than the cross spectrum **xpsectra_{0-1-2}tau**. They are also splitted in Real part and Imaginary part in the netCDF files, and they are also stored on the half part of the wavenumer along range axis since they are symmetric.

[9]:
```
ds['var_xspectra_2tau'].isel(tile_line=0,tile_sample=0,burst=3).squeeze().
↪plot(cmap=cmap)
```

[9]:
```
<matplotlib.collections.QuadMesh at 0x7f0c30ac9570>
```



Explanation on specific variables

This section gives illustrated details on some of the variables:

- *Sentinel-1 Level-1B IFREMER variables explanation*

Product attributes

global attributes of the sub-swath file:

```
// global attributes:
                :version_xsar = "1.0.2" ;
                :version_xsarslc = "2024.1.31" ;
                :processor = "./lib/python3.9/site-packages/xsarslc/scripts/L1B_
↪xspectra_IW_SLC_IFR.py" ;
                :generation_date = "2024-Jan-31" ;
```

global attributes of the intra-burst group:

```
// group attributes:
:name = "SENTINEL1_DS:/data/esa/sentinel-1a/L1/IW/S1A_IW_SLC__1S/2018/065/S1A_IW_SLC__
↪1SDV_20180306T230337_20180306T230405_020901_023DBA_73A0.SAFE:IW1" ;
                :short_name = "SENTINEL1_DS:S1A_IW_SLC__1SDV_20180306T230337_
```

(continues on next page)

(continued from previous page)

```
↪20180306T230405_020901_023DBA_73A0.SAFE:IW1" ;
                :product = "SLC" ;
                :safe = "S1A_IW_SLC__1SDV_20180306T230337_20180306T230405_020901_
↪023DBA_73A0.SAFE" ;
                :swath = "IW" ;
                :multidataset = "False" ;
                :ipf = 2.84 ;
                :platform = "SENTINEL-1A" ;
                :pols = "VV VH" ;
                :start_date = "2018-03-06 23:03:38.051625" ;
                :stop_date = "2018-03-06 23:04:03.182857" ;
                :footprint = "POLYGON ((-76.08430839864033 26.63756326996933, -75.
↪18793064940225 26.79052036139598, -75.50267764839376 28.28730160398311, -76.
↪41199086776611 28.13488378485153, -76.08430839864033 26.63756326996933))" ;
                :coverage = "169km * 90km (line * sample )" ;
                :orbit_pass = "Ascending" ;
                :platform_heading = -12.5313253576975 ;
                :comment = "denoised digital number, read at full resolution" ;
                :history = "digital_number: measurement/s1a-iw1-slc-v*-20180306t230338-
↪20180306t230403-020901-023dba-00*.tiff\n" ;
                :radar_frequency = 5405000454.33435 ;
                :azimuth_time_interval = 0.0020555563 ;
                :tile_width_sample = 17700LL ;
                :tile_width_line = 17700LL ;
                :tile_overlap_sample = 0LL ;
                :tile_overlap_line = 0LL ;
```

global attributes of the inter-burst group:

```
// group attributes:
                :name = "SENTINEL1_DS:/data/esa/sentinel-1a/L1/IW/S1A_IW_SLC__1S/2021/
↪108/S1A_IW_SLC__1SDV_20210418T212223_20210418T212253_037510_046C42_6560.SAFE:IW1" ;
                :short_name = "SENTINEL1_DS:S1A_IW_SLC__1SDV_20210418T212223_
↪20210418T212253_037510_046C42_6560.SAFE:IW1" ;
                :product = "SLC" ;
                :safe = "S1A_IW_SLC__1SDV_20210418T212223_20210418T212253_037510_
↪046C42_6560.SAFE" ;
                :swath = "IW" ;
                :multidataset = "False" ;
                :ipf = 3.31 ;
                :platform = "SENTINEL-1A" ;
                :pols = "VV VH" ;
                :start_date = "2021-04-18 21:22:23.644969" ;
                :stop_date = "2021-04-18 21:22:51.563535" ;
                :footprint = "POLYGON ((128.3637024723631 14.57604630647329, 127.
↪5496545856357 14.73346785356862, 127.2081937934356 13.04621122868775, 128.
↪0161923665588 12.88763160287155, 128.3637024723631 14.57604630647329))" ;
                :coverage = "191km * 89km (line * sample )" ;
                :orbit_pass = "Descending" ;
                :platform_heading = -167.922014873722 ;
                :azimuth_steering_rate = 1.590368784 ;
                :mean_incidence = 34.0561728242343 ;
                :azimuth_time_interval = 0.0020555563 ;
                :tile_width_sample = 17700LL ;
                :tile_width_line = 17700LL ;
                :tile_overlap_sample = 0LL ;
                :tile_overlap_line = 0LL ;
```

Product access

Currently the Level-1B SAR Sentinel-1 Ifremer/SARWAVE product is disseminated through this URL:

https://cerweb.ifremer.fr/datarmor/sarwave/diffusion/sar/iw/slc/l1b/experimental_product_collection/

Acknowledgment

The Sentinel-1 Level-1B SAR IFREMER Product has been co-funded by ESA through the SARWAVE project (https://www.sarwave.org/). The processor development benefits from support and contribution from/to Sentinel-1 Mission Performance Cluster (https://sar-mpc.eu/about/activities-and-team/).

Download this notebook from github[11].

0.2.4 SAR Sentinel-1 cross spectrum computation from large image Interferometric Wide-swath SLC

[1]:
```
import os
import xsarslc.get_test_files
import logging
import xsar
from importlib import reload
reload(logging)
logging.basicConfig(level=logging.INFO)
logging.info('start notebook')
import xsarslc.processing.xspectra as proc
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.filterwarnings("ignore")
```

```
INFO:root:start notebook
```

get datatree with digital numbers and annotations from IW SLC product

[2]:
```
import xsarslc
import xsar
import os
#one_tiff = '/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/
↪sentinel-1a/L1/IW/S1A_IW_SLC__1S/2022/127/S1A_IW_SLC__1SDV_20220507T162411_
↪20220507T162439_043107_0525DE_734D.SAFE/measurement/s1a-iw1-slc-vv-20220507t162411-
↪20220507t162439-043107-0525de-004.tiff'
#tmp = xsar.get_test_file('S1A_IW_SLC__1SDV_20170907T103019_20170907T103047_018268_
↪01EB76_5F55.SAFE')
fullpathsafeSLC = xsar.get_test_file('S1A_IW_SLC__1SDV_20181008T134439_
↪20181008T134506_024045_02A0A6_65C8.SAFE') # only one subswath IW1 zipped

#fullpathsafeSLC = os.path.dirname(os.path.dirname(one_tiff))
#imagette_number = os.path.basename(one_tiff).split('-')[1].replace('iw','')
imagette_number = 1
```

(continues on next page)

[11] https://github.com/umr-lops/xsar_slc/tree/main/docs/examples/xspec_IW_intra_and_inter_burst.ipynb

(continued from previous page)

```
str_gdal = 'SENTINEL1_DS:%s:IW%s'%(fullpathsafeSLC,imagette_number)
print('str_gdal',str_gdal)
xsarobj = xsar.Sentinel1Dataset(str_gdal) #,resolution='30m'
dt = xsarobj.datatree
dt.load()
dt
```

```
str_gdal SENTINEL1_DS:/home1/scratch/agrouaze/xsardatasync/xsardata/S1A_IW_SLC__1SDV_
↪20181008T134439_20181008T134506_024045_02A0A6_65C8.SAFE:IW1
```

[2]:
```
DataTree('None', parent=None)
│   Dimensions:  ()
│   Data variables:
│       *empty*
│   Attributes: (12/15)
│       name:               SENTINEL1_DS:/home1/scratch/agrouaze/xsardatasync/xsar…
│       short_name:         SENTINEL1_DS:S1A_IW_SLC__1SDV_20181008T134439_20181008…
│       product:            SLC
│       safe:               S1A_IW_SLC__1SDV_20181008T134439_20181008T134506_02404…
│       swath:              IW
│       multidataset:       False
│       …                   …
│       start_date:         2018-10-08 13:44:40.984312
│       stop_date:          2018-10-08 13:45:06.117599
│       footprint:          POLYGON ((-114.6305305568416 32.60521729566573, -115.5…
│       coverage:           171km * 88km (line * sample )
│       orbit_pass:         Descending
│       platform_heading:   -167.2011573025802
├── DataTree('measurement')
│       Dimensions:          (line: 13455, sample: 21135, pol: 2)
│       Coordinates:
│         * line             (line) int64 0 1 2 3 4 5 … 13450 13451 13452 13453 13454
│         * sample           (sample) int64 0 1 2 3 4 5 … 21130 21131 21132 21133 21134
│         * pol              (pol) object 'VV' 'VH'
│       Data variables:
│           digital_number   (pol, line, sample) complex64 0j 0j 0j 0j 0j … 0j 0j 0j 0j
│           time             (line) datetime64[ns] 2018-10-08T13:44:40.984142 … 2018…
│           sampleSpacing    float64 2.33
│           lineSpacing      float64 13.98
│       Attributes: (12/15)
│           name:               SENTINEL1_DS:/home1/scratch/agrouaze/xsardatasync/xsar…
│           short_name:         SENTINEL1_DS:S1A_IW_SLC__1SDV_20181008T134439_20181008…
│           product:            SLC
│           safe:               S1A_IW_SLC__1SDV_20181008T134439_20181008T134506_02404…
│           swath:              IW
│           multidataset:       False
│           …                   …
│           start_date:         2018-10-08 13:44:40.984312
│           stop_date:          2018-10-08 13:45:06.117599
│           footprint:          POLYGON ((-114.6305305568416 32.60521729566573, -115.5…
│           coverage:           171km * 88km (line * sample )
│           orbit_pass:         Descending
│           platform_heading:   -167.2011573025802
├── DataTree('geolocation_annotation')
│       Dimensions:          (line: 10, sample: 21)
│       Coordinates:
│         * line             (line) int64 0 1495 2990 4485 … 8970 10465 11960 13454
│         * sample           (sample) int64 0 1057 2114 3171 … 17969 19026 20083 21134
```

(continues on next page)

(continued from previous page)

```
|       Data variables:
|           longitude       (line, sample) float64 -114.6 -114.7 … -115.9 -115.9
|           latitude        (line, sample) float64 32.61 32.61 32.62 … 31.23 31.24
|           height          (line, sample) float64 22.2 22.2 22.2 … 781.9 781.9 781.9
|           azimuthTime     (line, sample) datetime64[ns] 2018-10-08T13:44:40.984060 …
|           slantRangeTime  (line, sample) float64 0.005334 0.00535 … 0.005662
|           incidenceAngle  (line, sample) float64 30.65 30.99 31.32 … 36.44 36.7
|           elevationAngle  (line, sample) float64 27.34 27.64 27.93 … 32.36 32.58
|       Attributes:
|           footprint:         POLYGON ((-114.6305305568416 32.60521729566573, -115.5…
|           coverage:          171km * 88km (line * sample )
|           approx_transform:  |-0.00,-0.00,-114.64|\n|-0.00, 0.00, 32.61|\n| 0.00, 0…
|           history:           annotations
├── DataTree('bursts')
|       Dimensions:           (burst: 9, line: 1495)
|       Dimensions without coordinates: burst, line
|       Data variables:
|           azimuthTime       (burst) datetime64[ns] 2018-10-08T13:44:40.984312 … 2…
|           azimuthAnxTime    (burst) float64 2.43e+03 2.432e+03 … 2.449e+03 2.452e+03
|           sensingTime       (burst) datetime64[ns] 2018-10-08T13:44:42.100540 … 2…
|           byteOffset        (burst) int64 107955 126495255 … 884819055 1011206355
|           firstValidSample  (burst, line) float64 nan nan nan nan … nan nan nan nan
|           lastValidSample   (burst, line) float64 nan nan nan nan … nan nan nan nan
|           linesPerBurst     int64 1495
|           samplesPerBurst   int64 21135
|       Attributes:
|           history:  annotations
├── DataTree('FMrate')
|       Dimensions:                  (azimuthTime: 11)
|       Coordinates:
|         * azimuthTime              (azimuthTime) datetime64[ns] 2018-10-08T13:44:36…
|       Data variables:
|           t0                       (azimuthTime) float64 0.005334 … 0.005334
|           azimuthFmRatePolynomial  (azimuthTime) object -2330.76015196 + 450153.757…
|       Attributes:
|           history:  annotations
├── DataTree('doppler_estimate')
|       Dimensions:                       (azimuthTime: 11, nb_fine_dce: 20)
|       Coordinates:
|         * azimuthTime                   (azimuthTime) datetime64[ns] 2018-10-08T13:…
|         * nb_fine_dce                   (nb_fine_dce) int64 0 1 2 3 4 … 16 17 18 19
|       Data variables:
|           t0                            (azimuthTime) float64 0.005338 … 0.005337
|           geometryDcPolynomial          (azimuthTime) object -0.6347436 + 462.2098·…
|           dataDcPolynomial              (azimuthTime) object 12.95983 + 51953.91·x …
|           fineDceAzimuthStartTime       (azimuthTime) datetime64[ns] 2018-10-08T13:…
|           fineDceAzimuthStopTime        (azimuthTime) datetime64[ns] 2018-10-08T13:…
|           dataDcRmsError                (azimuthTime) float64 7.29 6.658 … 12.9
|           slantRangeTime                (azimuthTime, nb_fine_dce) float64 0.005344…
|           frequency                     (azimuthTime, nb_fine_dce) float64 29.09 …
|           dataDcRmsErrorAboveThreshold  (azimuthTime) bool False False … False False
|       Attributes:
|           history:  annotations
├── DataTree('orbit')
|       Dimensions:     (time: 17)
|       Coordinates:
|         * time        (time) datetime64[ns] 2018-10-08T13:43:31.294060 … 2018-10-…
```

(continues on next page)

(continued from previous page)

```
│       Data variables:
│           velocity_x  (time) float64 -3.097e+03 -3.081e+03 … -2.801e+03 -2.778e+03
│           velocity_y  (time) float64 -3.51e+03 -3.445e+03 … -2.498e+03 -2.429e+03
│           velocity_z  (time) float64 -5.98e+03 -6.026e+03 … -6.604e+03 -6.64e+03
│           position_x  (time) float64 -1.925e+06 -1.956e+06 … -2.368e+06 -2.396e+06
│           position_y  (time) float64 -5.392e+06 -5.427e+06 … -5.843e+06 -5.868e+06
│           position_z  (time) float64 4.15e+06 4.09e+06 … 3.204e+06 3.138e+06
│       Attributes:
│           orbit_pass:        Descending
│           platform_heading:  -167.2011573025802
│           frame:             Earth Fixed
│           history:           orbit:\n  annotation/s1a.xml:\n  - //product/generalAn…
├── DataTree('image')
│       Dimensions:                  (dim_0: 2)
│       Dimensions without coordinates: dim_0
│       Data variables: (12/14)
│           LineUtcTime              (dim_0) datetime64[ns] 2018-10-08T13:44:40.98431…
│           numberOfLines            int64 13455
│           numberOfSamples          int64 21135
│           azimuthPixelSpacing      float64 13.98
│           slantRangePixelSpacing   float64 2.33
│           groundRangePixelSpacing  float64 4.185
│           …                        …
│           slantRangeTime           float64 0.005334
│           swath_subswath           <U3 'IW1'
│           radarFrequency           float64 5.405e+09
│           rangeSamplingRate        float64 6.435e+07
│           azimuthSteeringRate      float64 1.59
│           history                  <U824 'image:\n  annotation/s1a.xml:\n  - /produ…
├── DataTree('calibration')
│       Dimensions:      (line: 29, sample: 530, pol: 2)
│       Coordinates:
│         * line         (line) int64 -1038 -552 90 576 1063 … 12506 12993 13628 14115
│         * sample       (sample) int64 0 40 80 120 160 … 21040 21080 21120 21134
│         * pol          (pol) object 'VV' 'VH'
│       Data variables:
│           sigma0_lut   (pol, line, sample) float64 331.9 331.8 331.7 … 306.4 306.4
│           gamma0_lut   (pol, line, sample) float64 307.8 307.7 307.6 … 274.3 274.3
│           azimuthTime  (pol, line) datetime64[ns] 2018-10-08T13:44:39.169256 … 20…
│       Attributes:
│           history:  calibration
├── DataTree('noise_range')
│       Dimensions:      (line: 11, sample: 530, pol: 2)
│       Coordinates:
│         * line         (line) int64 -1495 0 1495 2990 4485 … 8970 10465 11960 13454
│         * sample       (sample) int64 0 40 80 120 160 … 21040 21080 21120 21134
│         * pol          (pol) object 'VV' 'VH'
│       Data variables:
│           noise_lut    (pol, line, sample) float64 547.5 544.0 540.6 … 512.3 513.3
│           azimuthTime  (pol, line) datetime64[ns] 2018-10-08T13:44:38.229867 … 20…
│       Attributes:
│           history:  noise
├── DataTree('noise_azimuth')
│       Dimensions:       (line: 1359, pol: 2)
│       Coordinates:
│         * line          (line) int64 0 10 20 30 40 … 13420 13430 13440 13450 13454
│         * pol           (pol) object 'VV' 'VH'
```

(continues on next page)

(continued from previous page)

```
│       Data variables:
│           noise_lut     (pol, line) float64 1.171 1.166 1.161 … 1.141 1.145 1.146
│           line_start    (pol) int64 0 0
│           line_stop     (pol) int64 13454 13454
│           sample_start  (pol) int64 0 0
│           sample_stop   (pol) int64 21134 21134
│       Attributes:
│           history:  noise
├── DataTree('antenna_pattern')
│       Dimensions:         (swath_nb: 1, dim_slantRangeTime: 663, dim_azimuthTime:␣
↪10)
│       Coordinates:
│         * swath_nb        (swath_nb) int64 1
│       Dimensions without coordinates: dim_slantRangeTime, dim_azimuthTime
│       Data variables:
│           slantRangeTime  (swath_nb, dim_slantRangeTime) float64 0.005334 … 0.005663
│           swath           (swath_nb, dim_azimuthTime) float64 1.0 1.0 1.0 … 1.0 1.0
│           roll            (swath_nb, dim_azimuthTime) float64 30.15 30.15 … 30.16
│           azimuthTime     (swath_nb, dim_azimuthTime) datetime64[ns] 2018-10-08T13:…
│           terrainHeight   (swath_nb, dim_azimuthTime) float64 20.73 74.82 … 709.7
│           elevationAngle  (swath_nb, dim_azimuthTime, dim_slantRangeTime) float64 2…
│           incidenceAngle  (swath_nb, dim_azimuthTime, dim_slantRangeTime) float64 3…
│           gain            (swath_nb, dim_azimuthTime, dim_slantRangeTime) float64 1…
│       Attributes:
│           dim_azimuthTime:     max dimension of azimuthTime for a swath
│           dim_slantRangeTime:  max dimension of slantRangeTime for a swath
│           comment:             The antenna pattern data set record contains a list …
│           example:             for example, if swath Y is smaller than swath X, use…
│           source:              Sentinel-1 Product Specification
│           history:             antenna_pattern:\n  annotation/s1a.xml:\n  - /produc…
├── DataTree('swath_merging')
└── DataTree('recalibration')
```

cross spectrum in inter bursts parts of the IW product

[3]:
```
inter_xs = proc.compute_IW_subswath_interburst_xspectra(dt,
                                        tile_width={'sample': 20.e3, 'line': 20.e3},
                                        tile_overlap={'sample': 0, 'line': 0},
                                        periodo_width = {'sample': 2000, 'line': 1200}
↪,
                                        periodo_overlap = {'sample': 1000, 'line':␣
↪600},
                                        dev=True,polarization='VV')
inter_xs
```

```
INFO:root:reduce number of burst -> 1
0it [00:00, ?it/s]
```

[3]:
```
<xarray.Dataset>
Dimensions:  ()
Data variables:
    *empty*
```

0.2.5 cross spectrum in intra bursts parts of the IW product

[4]:
```
intra_xs = proc.compute_IW_subswath_intraburst_xspectra(dt,
                                          tile_width={'sample': 20.e3, 'line': 20.e3},
                                         tile_overlap={'sample': 0, 'line': 0},
                                         periodo_width = {'sample': 4000, 'line': 4000}
↪,
                                         periodo_overlap = {'sample': 2000, 'line':␣
↪2000},
                                         polarization='VV',dev=True)
intra_xs
```

```
INFO:root:reduce number of burst -> 1
intrabursts: 100
↪%|████████████████████████████████████████████████████████████████████████████████
↪1/1 [01:13<00:00, 73.74s/it]
```

[4]:
```
<xarray.Dataset>
Dimensions:                    (tile_sample: 4, tile_line: 1, bt_thresh: 5,
                                0tau: 3, freq_line: 57, freq_sample: 451,
                                1tau: 2, 2tau: 1, c_sample: 2, c_line: 2,
                                k_gp: 4, phi_hf: 5, lambda_range_max_macs: 10)
Coordinates:
    pol                        <U2 'VV'
  * bt_thresh                  (bt_thresh) int64 5 50 100 150 200
    k_rg                       (tile_sample, freq_sample) float64 0.0 … 0.7068
    k_az                       (freq_line) float64 -0.04417 -0.04259 … 0.04417
    line                       (tile_line) int64 747
    sample                     (tile_line, tile_sample) int64 3204 7796 … 17483
    longitude                  (tile_line, tile_sample) float64 -114.8 … -115.4
    latitude                   (tile_line, tile_sample) float64 32.54 … 32.64
  * k_gp                       (k_gp) int64 1 2 3 4
  * phi_hf                     (phi_hf) int64 1 2 3 4 5
  * lambda_range_max_macs      (lambda_range_max_macs) float64 50.0 … 275.0
Dimensions without coordinates: tile_sample, tile_line, 0tau, freq_line,
                                freq_sample, 1tau, 2tau, c_sample, c_line
Data variables: (12/27)
    incidence                  (tile_line, tile_sample) float64 31.65 … 35.66
    ground_heading             (tile_line, tile_sample) float64 -169.1 … -169.4
    burst                      (tile_line) int64 0
    sensing_time               (tile_line, tile_sample) datetime64[ns] 2018-10…
    sigma0                     (tile_line, tile_sample) float64 0.1124 … 0.1928
    nesz                       (tile_line, tile_sample) float64 0.004084 … 0…
    …                          …
    corner_line                (tile_line, c_line) int64 32 1462
    corner_sample              (tile_line, tile_sample, c_sample) int64 951 …
    burst_corner_longitude     (tile_line, c_sample, c_line) float64 -114.6 …
    burst_corner_latitude      (tile_line, c_sample, c_line) float64 32.61 …
    cwave_params               (tile_line, tile_sample, k_gp, phi_hf, 2tau) float64 …
    macs                       (tile_line, tile_sample, lambda_range_max_macs, 2tau)␣
↪complex128 …
Attributes: (12/24)
    name:                      SENTINEL1_DS:/home1/scratch/agrouaze/xsardatasync…
    short_name:                SENTINEL1_DS:S1A_IW_SLC__1SDV_20181008T134439_201…
    product:                   SLC
    safe:                      S1A_IW_SLC__1SDV_20181008T134439_20181008T134506_…
    swath:                     IW
```

(continues on next page)

(continued from previous page)

```
    multidataset:           False
    …                       …
    radar_frequency:        5405000454.33435
    azimuth_time_interval:  0.002055556299999998
    tile_width_sample:      20000.0
    tile_width_line:        20000.0
    tile_overlap_sample:    0
    tile_overlap_line:      0
```

Download this notebook from github[12].

0.2.6 Example to compute image cross spectrum between different looks within a WV Sentinel-1 SLC product

in WV images (20 km X 20 km) there is only one burst (thus no overlapping area).

[1]:
```
import xsar
import warnings
import logging
from importlib import reload
reload(logging)
logging.basicConfig(level=logging.INFO)
logging.info('go')
# warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.filterwarnings("ignore")
#tmp = xsar.get_test_file('S1A_WV_SLC__1SSV_20160510T101603_20160510T102143_011195_
↪010EA1_Z010.SAFE')
tmp = xsar.get_test_file('S1A_WV_SLC__1SSV_20191005T100804_20191005T101314_029322_
↪035533_0597.SAFE') # very small product
# print(tmp)
# tmp = '/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/sentinel-
↪1a/L2/WV/S1A_WV_OCN__2S/2016/131/S1A_WV_OCN__2SSV_20160510T101603_20160510T102143_
↪011195_010EA1_F972.SAFE'
# tmp = '/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/sentinel-
↪1a/L1/WV/S1A_WV_SLC__1S/2016/131/S1A_WV_SLC__1SSV_20160510T101603_20160510T102143_
↪011195_010EA1_7038.SAFE'
# tmp = '/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/sentinel-
↪1a/L1/WV/S1A_WV_SLC__1S/2023/014/S1A_WV_SLC__1SSV_20230114T130811_20230114T133351_
↪046780_059BC5_FED5.SAFE'

strwv = 'SENTINEL1_DS:'+tmp+':WV_017'
print(strwv)
xsar_instance = xsar.Sentinel1Dataset(strwv)
xsar_instance
```

```
INFO:root:go
```

```
SENTINEL1_DS:/home1/scratch/agrouaze/xsardatasync/xsardata/S1A_WV_SLC__1SSV_
↪20191005T100804_20191005T101314_029322_035533_0597.SAFE:WV_017
```

[1]:
```
<Sentinel1Dataset full coverage object>
```

[12] https://github.com/umr-lops/xsar_slc/tree/main/docs/examples/xspec_WV_example.ipynb

[2]:
```
xsar_instance.dataset
```

[2]:
```
<xarray.Dataset>
Dimensions:          (line: 4923, sample: 5681, pol: 1)
Coordinates:
  * line             (line) int64 0 1 2 3 4 5 6 … 4917 4918 4919 4920 4921 4922
  * sample           (sample) int64 0 1 2 3 4 5 … 5675 5676 5677 5678 5679 5680
  * pol              (pol) object 'VV'
Data variables:
    digital_number   (pol, line, sample) complex64 dask.array<chunksize=(1, 4923,␣
↪5000), meta=np.ndarray>
    time             (line) datetime64[ns] 2019-10-05T10:11:58.450274048 … 2…
    sampleSpacing    float64 1.498
    lineSpacing      float64 4.133
Attributes: (12/15)
    name:               SENTINEL1_DS:/home1/scratch/agrouaze/xsardatasync/xsar…
    short_name:         SENTINEL1_DS:S1A_WV_SLC__1SSV_20191005T100804_20191005…
    product:            SLC
    safe:               S1A_WV_SLC__1SSV_20191005T100804_20191005T101314_02932…
    swath:              WV
    multidataset:       False
    …                   …
    start_date:         2019-10-05 10:11:58.450260
    stop_date:          2019-10-05 10:12:01.432986
    footprint:          POLYGON ((-61.07570128124852 28.34141558287481, -61.28…
    coverage:           20km * 20km (line * sample )
    orbit_pass:         Descending
    platform_heading:   -167.4153352100693
```

[3]:
```
xsar_instance.dataset.digital_number.values[0:10,0:10]
```

[3]:
```
array([[[   0.  +0.j,    0.  +0.j,    0.  +0.j, …,    0.  +0.j,
           0.  +0.j,    0.  +0.j],
        [   0.  +0.j,    0.  +0.j,    0.  +0.j, …,    0.  +0.j,
           0.  +0.j,    0.  +0.j],
        [   0.  +0.j,    0.  +0.j,    0.  +0.j, …,   14. +15.j,
          64. -30.j,   43. -46.j],
        …,
        [ -12.+243.j,  -69. -81.j,   13. -53.j, …,   82. +53.j,
         190.  +4.j,  108.-126.j],
        [  87. -29.j, -120. -67.j,   83. +82.j, …,  123.+140.j,
         232.  -8.j, -123.-134.j],
        [ -29. +21.j,  -34.  -6.j,  -44. +99.j, …,  213. +40.j,
         325.+204.j,   99. -20.j]]], dtype=complex64)
```

0.2.7 display digital numbers

[4]:
```
import holoviews as hv
import numpy as np
from holoviews.operation.datashader import datashade,rasterize
hv.extension('bokeh')
```

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

Data type cannot be displayed: application/vnd.holoviews_load.v0+json, application/javascript

Data type cannot be displayed: text/html, application/vnd.holoviews_exec.v0+json

[5]:
```
tmp = abs(xsar_instance.dataset['digital_number'].sel(pol='VV'))
DN = tmp.rolling({"sample":10,"line":10}).mean()
DN
```

[5]:
```
<xarray.DataArray 'digital_number' (line: 4923, sample: 5681)>
dask.array<truediv, shape=(4923, 5681), dtype=float32, chunksize=(4923, 5009),␣
↪chunktype=numpy.ndarray>
Coordinates:
  * line      (line) int64 0 1 2 3 4 5 6 7 … 4916 4917 4918 4919 4920 4921 4922
  * sample    (sample) int64 0 1 2 3 4 5 6 … 5674 5675 5676 5677 5678 5679 5680
    pol       <U2 'VV'
Attributes:
    comment:  denoised digital number, read at full resolution
    history:  digital_number: measurement/s1a-wv1-slc-vv-20191005t101158-2019…
```

[6]:
```
rasterize(hv.Image(DN.values).opts(width=800,height=700,cmap='Greys',colorbar=True))#␣
↪clim=(0,40)
```

[6]:
```
:DynamicMap   []
   :Image   [x,y]   (z)
```

0.2.8 compute the periodograms and the looks inside the periodograms to get a cross spectra

[7]:
```
from xsarslc.processing.xspectra import compute_WV_intraburst_xspectra
import time
t0 = time.time()
xs0 = compute_WV_intraburst_xspectra(dt=xsar_instance.datatree,
                                     polarization='VV',periodo_width={"line":
↪2000,"sample":2000},
                                     periodo_overlap={"line":1000,"sample":1000})
print('time to compute x-spectra on WV :%1.1f sec'%(time.time()-t0))
xs0
```

```
time to compute x-spectra on WV :70.5 sec
```

[7]:
```
<xarray.Dataset>
Dimensions:                    (bt_thresh: 5, 0tau: 3, freq_line: 97,
                                freq_sample: 274, 1tau: 2, 2tau: 1, c_sample: 2,
                                c_line: 2, k_gp: 4, phi_hf: 5,
                                lambda_range_max_macs: 10)
Coordinates:
    pol                        <U2 'VV'
  * bt_thresh                  (bt_thresh) int64 5 50 100 150 200
    k_rg                       (freq_sample) float64 0.0 0.003143 … 0.858
    k_az                       (freq_line) float64 -0.1505 -0.1473 … 0.1505
```

(continues on next page)

(continued from previous page)

```
    line                        int64 2461
    sample                      int64 2840
    longitude                   float64 -61.2
    latitude                    float64 28.27
  * k_gp                        (k_gp) int64 1 2 3 4
  * phi_hf                      (phi_hf) int64 1 2 3 4 5
  * lambda_range_max_macs       (lambda_range_max_macs) float64 50.0 … 275.0
Dimensions without coordinates: 0tau, freq_line, freq_sample, 1tau, 2tau,
                                c_sample, c_line
Data variables: (12/26)
    incidence                   float64 24.19
    ground_heading              float64 -168.5
    sensing_time                datetime64[ns] 2019-10-05T10:11:59.941637632
    sigma0                      float64 0.1942
    nesz                        float64 0.001677
    bright_targets_histogram    (bt_thresh) int64 54 0 0 0 0
    …                          …
    corner_line                 (c_line) int64 25 4897
    corner_sample               (c_sample) int64 25 5655
    burst_corner_longitude      (c_sample, c_line) float64 -61.08 … -61.32
    burst_corner_latitude       (c_sample, c_line) float64 28.34 28.16 28.38 28.2
    cwave_params                (k_gp, phi_hf, 2tau) float64 4.855 … -0.334
    macs                        (lambda_range_max_macs, 2tau) complex128 (0.167…
Attributes: (12/21)
    name:                       SENTINEL1_DS:/home1/scratch/agrouaze/xsardatasync…
    short_name:                 SENTINEL1_DS:S1A_WV_SLC__1SSV_20191005T100804_201…
    product:                    SLC
    safe:                       S1A_WV_SLC__1SSV_20191005T100804_20191005T101314_…
    swath:                      WV
    multidataset:               False
    …                          …
    radar_frequency:            5405000704.0
    azimuth_time_interval:      0.0006059988896297211
    tile_width_line:            20138.934607
    tile_width_sample:          <xarray.DataArray 'cumulative ground length' ()>\…
    tile_overlap_sample:        0.0
    tile_overlap_line:          0.0
```

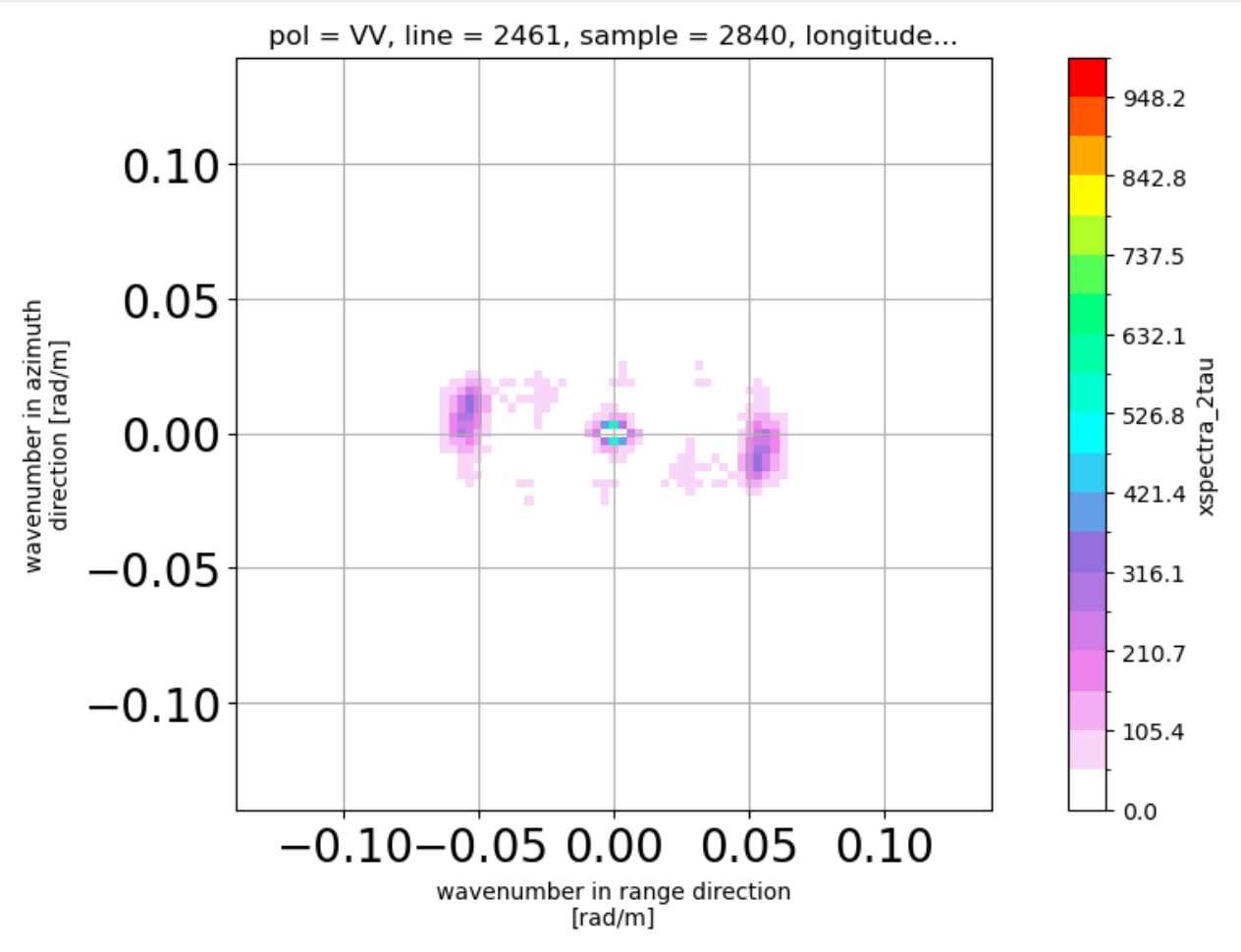
[8]:
```
import numpy as np
from xsarslc.processing import xspectra
xs = xs0.swap_dims({'freq_line': 'k_az', 'freq_sample': 'k_rg'})
xs = xspectra.symmetrize_xspectrum(xs['xspectra_2tau'], dim_range='k_rg', dim_azimuth=
↪'k_az')

########################################## real part ############################
coS=np.abs(xs.mean(dim='2tau').real)
coS_reduced = coS.where(np.logical_and(np.abs(coS.k_rg)<=0.14, np.abs(coS.k_az)<=0.
↪14), drop=True)
from matplotlib import pyplot as plt
%matplotlib inline
from matplotlib import colors as mcolors
cmap = mcolors.LinearSegmentedColormap.from_list("", ["white","violet","mediumpurple",
↪"cyan","springgreen","yellow","red"])
PuOr = plt.get_cmap('PuOr')
plt.figure(figsize=(8,6))
#coS_reduced_av=coS_reduced.rolling(k_rg=3, center=True).mean().rolling(k_az=3,␣
```

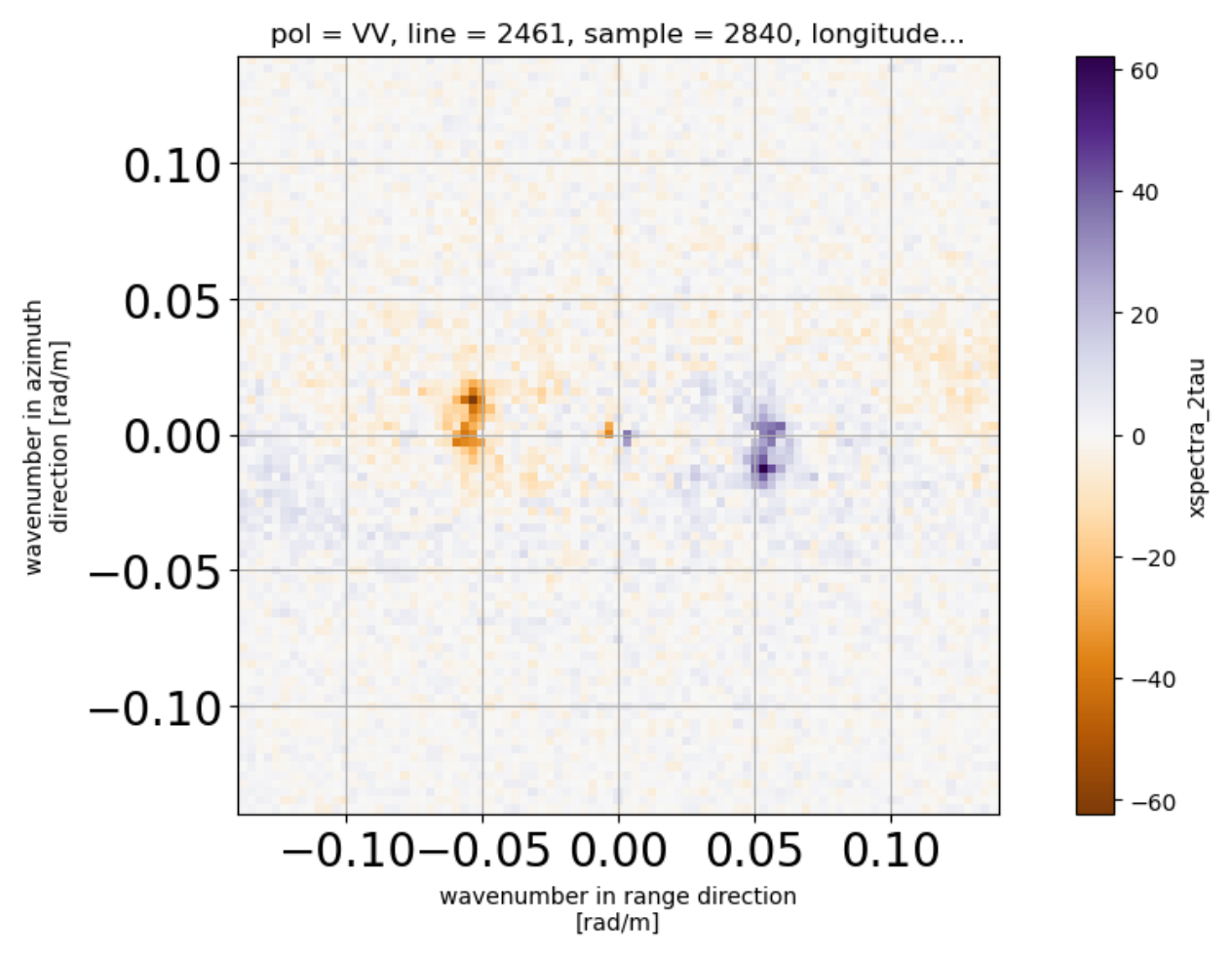
(continues on next page)

(continued from previous page)

```
↪center=True).mean()
coS_reduced.plot(cmap=cmap, levels=20, vmin=0)

plt.grid()
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.axis('scaled')

########################################## imag.part ############################
ims = xs.mean(dim='2tau').imag.squeeze()
xS_red = ims.where(np.logical_and(np.abs(ims.k_rg)<=0.14, np.abs(ims.k_az)<=0.14),␣
↪drop=True)
#xS_av=xS_red.rolling(k_rg=3, center=True).mean().rolling(k_az=3, center=True).mean()

plt.figure(figsize=(8,6))
xS_red.plot(x='k_rg',cmap=PuOr)
plt.grid()
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.axis('scaled')
```

[8]:
```
(-0.13984955506418678,
 0.13984955506418678,
 -0.13949481344309983,
 0.13949481344309983)
```


pol = VV, line = 2461, sample = 2840, longitude...
wavenumber in azimuth direction [rad/m]
0.10
0.05
0.00
−0.05
−0.10
−0.10 −0.05 0.00 0.05 0.10
wavenumber in range direction [rad/m]
948.2
842.8
737.5
632.1
526.8
421.4
316.1
210.7
105.4
0.0
xspectra_2tau



Download this notebook from github[13].

0.2.9 A notebook to illustrate how to compute and correct Impulse Response in IW SLC product

compute IW1 impulse response

[1]:
```
import warnings
warnings.filterwarnings("ignore")
from xsarslc.processing.impulseResponse import generate_IWS_AUX_file_ImpulseReponse
# SENTINEL1_DS:/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/
↪sentinel-1a/L1/WV/S1A\_WV\_SLC\_\_1S/2018/050/S1A\_WV\_SLC\_\_1SSV\_20180219T221522\
↪_20180219T222851\_020681\_0236CE\_8F55.SAFE:WV_051
subswathes = {'/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/
↪sentinel-1a/L1/IW/S1A_IW_SLC__1S/2022/271/S1A_IW_SLC__1SDV_20220928T095555_
↪20220928T095622_045203_056722_FB4F.SAFE/':[1],
             }
```

(continues on next page)

[13] https://github.com/umr-lops/xsar_slc/tree/main/docs/examples/example_IW_compute_and_correct_from_impulse_response.ipynb

(continued from previous page)

```
subswath_number = 1

IRs_IW1_VV = generate_IWS_AUX_file_ImpulseReponse(subswathes, subswath_number ,␣
↪polarization='VV')
IRs_IW1_VV
#IRs_IW1_VH = generate_IWS_AUX_file_ImpulseReponse(subswathes, subswath_number ,␣
↪polarization='VH')
#IRs_IW1_VH
```

```
path: /home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/sentinel-1a/
↪L1/IW/S1A_IW_SLC__1S/2022/271/S1A_IW_SLC__1SDV_20220928T095555_20220928T095622_
↪045203_056722_FB4F.SAFE/
```

[1]:
```
<xarray.Dataset>
Dimensions:          (k_srg: 2048, k_az: 256)
Coordinates:
    pol              <U2 'VV'
    burst            int64 1
  * k_srg            (k_srg) float64 -1.349 -1.347 -1.346 … 1.345 1.346 1.347
  * k_az             (k_az) float64 -0.2236 -0.2219 -0.2201 … 0.2201 0.2219
Data variables:
    range_IR         (k_srg) float64 0.0005885 0.0005878 … 0.0005874 0.0005885
    mean_incidence   float64 33.5
    azimuth_IR       (k_az) float64 0.002508 0.002509 … 0.00251 0.002507
Attributes: (12/21)
    long_name:               Impulse response in range direction
    name:                    SENTINEL1_DS:/home/datawork-cersat-public/cache/p…
    short_name:              SENTINEL1_DS:S1A_IW_SLC__1SDV_20220928T095555_202…
    product:                 SLC
    safe:                    S1A_IW_SLC__1SDV_20220928T095555_20220928T095622_…
    swath:                   IW
    …                      …
    platform_heading:        -168.0316299165244
    comment:                 denoised digital number, read at full resolution
    history:                 digital_number: /home/datawork-cersat-public/cach…
    radar_frequency:         5405000454.33435
    azimuth_time_interval:   0.002055556299999998
    subswath:                IW1
```

save IR file to netCDF

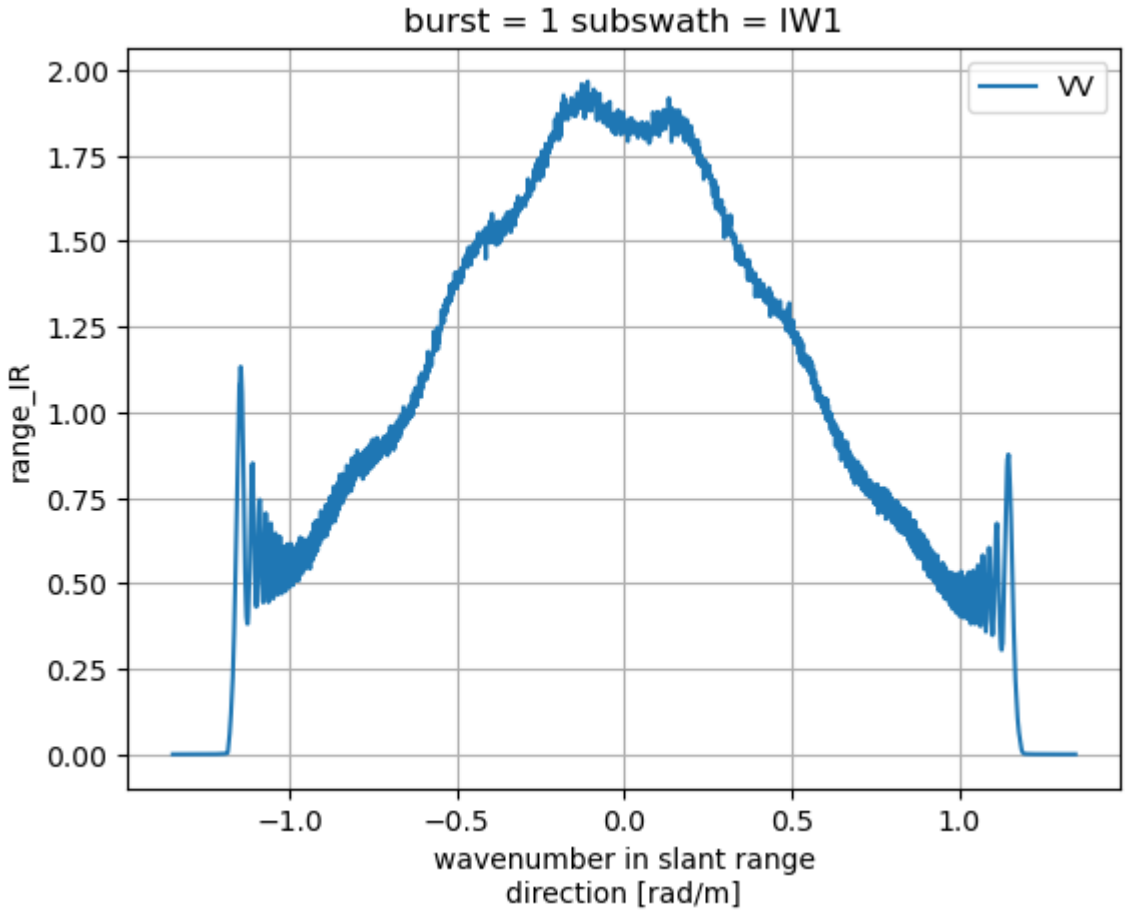
[2]:
```
out_file = "/tmp/S1_IR_IW1_VV_AUX_FILE.nc"
if 'footprint' in IRs_IW1_VV.attrs:
    IRs_IW1_VV.attrs.update({'footprint': str(IRs_IW1_VV.footprint)})
if 'multidataset' in IRs_IW1_VV.attrs:
    IRs_IW1_VV.attrs.update({'multidataset': str(IRs_IW1_VV.multidataset)})
IRs_IW1_VV.to_netcdf(out_file)
print(out_file)
```

```
/tmp/S1_IR_IW1_VV_AUX_FILE.nc
```

display IW1 range impulse response for VV and VH

[3]:
```
from matplotlib import pyplot as plt
IRs_IW1_VV['range_IR'].plot(label='VV')
#IRs_IW1_VH['range_IR'].plot(label='VH',alpha=0.7)
plt.legend()
plt.grid(True)
plt.title('burst = %s subswath = %s'%(IRs_IW1_VV.burst.values,IRs_IW1_VV.attrs[
↪'subswath']))
```

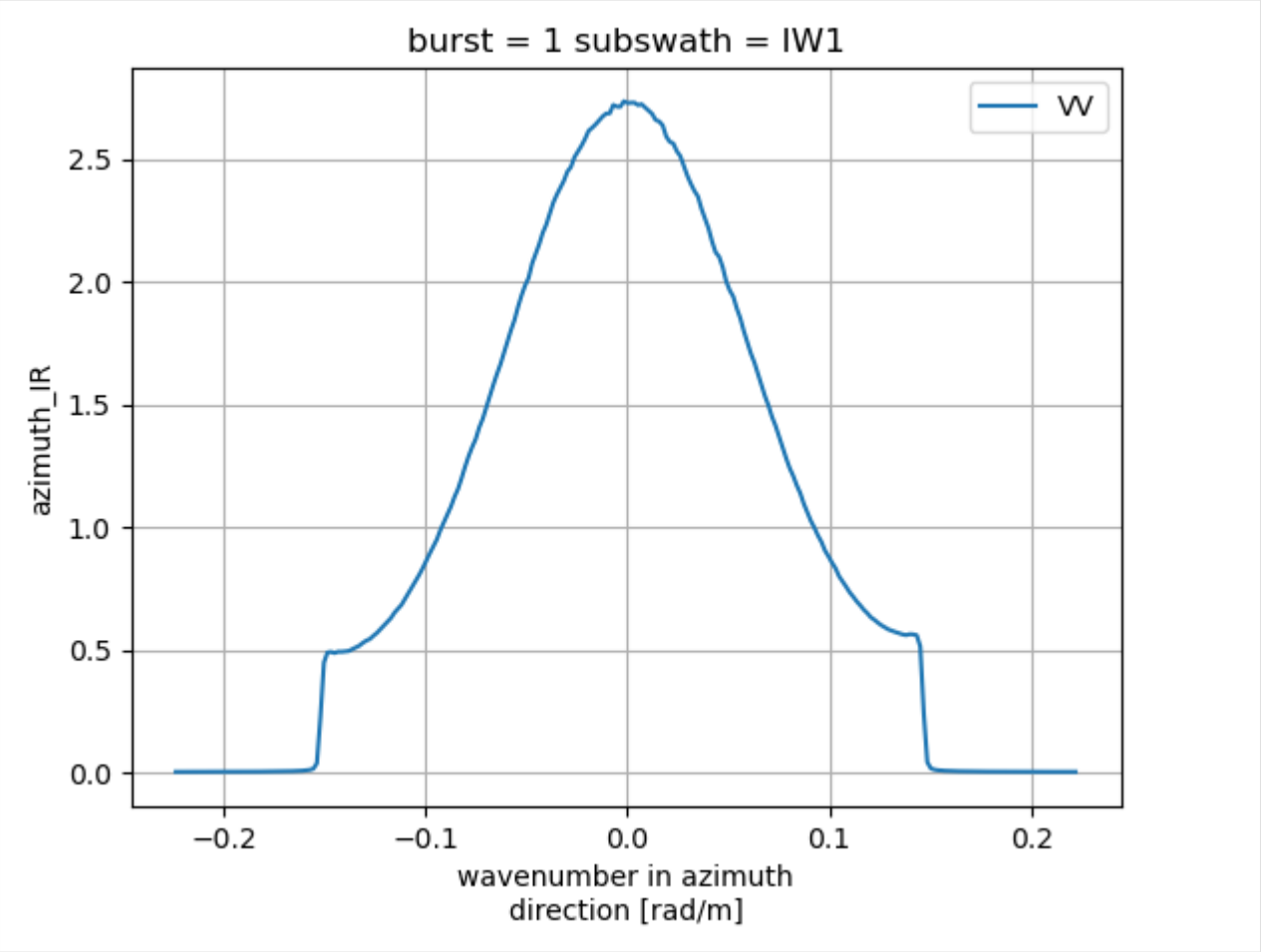
[3]:
```
Text(0.5, 1.0, 'burst = 1 subswath = IW1')
```



display IW1 azimuth impulse response for VV and VH

[4]:
```
from matplotlib import pyplot as plt
IRs_IW1_VV['azimuth_IR'].plot(label='VV')
#IRs_IW1_VH['azimuth_IR'].plot(label='VH',alpha=0.7)
plt.legend()
plt.grid(True)
plt.title('burst = %s subswath = %s'%(IRs_IW1_VV.burst.values,IRs_IW1_VV.attrs[
↪'subswath']))
```

[4]:
```
Text(0.5, 1.0, 'burst = 1 subswath = IW1')
```



apply IR correction when computing a cross spectra

[5]:
```
import xsarslc
import xsar
import os
one_tiff = '/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/
↪sentinel-1a/L1/IW/S1A_IW_SLC__1S/2022/127/S1A_IW_SLC__1SDV_20220507T162411_
↪20220507T162439_043107_0525DE_734D.SAFE/measurement/s1a-iw1-slc-vv-20220507t162411-
↪20220507t162439-043107-0525de-004.tiff'
fullpathsafeSLC = os.path.dirname(os.path.dirname(one_tiff))
imagette_number = os.path.basename(one_tiff).split('-')[1].replace('iw','')
str_gdal = 'SENTINEL1_DS:%s:IW%s'%(fullpathsafeSLC,imagette_number)
print('str_gdal',str_gdal)
xsarobj = xsar.Sentinel1Dataset(str_gdal) #,resolution='30m'
xsarobj
```

```
str_gdal SENTINEL1_DS:/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/
↪esa/sentinel-1a/L1/IW/S1A_IW_SLC__1S/2022/127/S1A_IW_SLC__1SDV_20220507T162411_
↪20220507T162439_043107_0525DE_734D.SAFE:IW1
```

[5]:
```
<Sentinel1Dataset full coverage object>
```

[6]:
```
dt = xsarobj.datatree
dt.load()
dt
```

[6]:
```
DataTree('None', parent=None)
|   Dimensions:  ()
|   Data variables:
|       *empty*
|   Attributes: (12/15)
|       name:                 SENTINEL1_DS:/home/datawork-cersat-public/cache/projec…
|       short_name:           SENTINEL1_DS:S1A_IW_SLC__1SDV_20220507T162411_20220507…
|       product:              SLC
|       safe:                 S1A_IW_SLC__1SDV_20220507T162411_20220507T162439_04310…
|       swath:                IW
|       multidataset:         False
|       …                   …
|       start_date:           2022-05-07 16:24:11.657511
|       stop_date:            2022-05-07 16:24:39.551410
|       footprint:            POLYGON ((-156.007554522451 22.11305598776757, -156.82…
|       coverage:             190km * 86km (line * sample )
|       orbit_pass:           Descending
|       platform_heading:     -167.7204378813183
├── DataTree('measurement')
|       Dimensions:            (line: 14900, sample: 20566, pol: 2)
|       Coordinates:
|         * line               (line) int64 0 1 2 3 4 5 … 14895 14896 14897 14898 14899
|         * sample             (sample) int64 0 1 2 3 4 5 … 20561 20562 20563 20564 20565
|         * pol                (pol) object 'VV' 'VH'
|       Data variables:
|           digital_number     (pol, line, sample) complex64 0j 0j 0j 0j 0j … 0j 0j 0j 0j
|           time               (line) datetime64[ns] 2022-05-07T16:24:11.657340 … 2022…
|           sampleSpacing      float64 2.33
|           lineSpacing        float64 14.0
|       Attributes: (12/15)
|           name:                 SENTINEL1_DS:/home/datawork-cersat-public/cache/projec…
|           short_name:           SENTINEL1_DS:S1A_IW_SLC__1SDV_20220507T162411_20220507…
|           product:              SLC
|           safe:                 S1A_IW_SLC__1SDV_20220507T162411_20220507T162439_04310…
|           swath:                IW
|           multidataset:         False
|           …                   …
|           start_date:           2022-05-07 16:24:11.657511
|           stop_date:            2022-05-07 16:24:39.551410
|           footprint:            POLYGON ((-156.007554522451 22.11305598776757, -156.82…
|           coverage:             190km * 86km (line * sample )
|           orbit_pass:           Descending
|           platform_heading:     -167.7204378813183
├── DataTree('geolocation_annotation')
|       Dimensions:            (line: 11, sample: 21)
|       Coordinates:
|         * line               (line) int64 0 1490 2980 4470 … 10430 11920 13410 14899
|         * sample             (sample) int64 0 1029 2058 3087 … 17493 18522 19551 20565
|       Data variables:
|           longitude          (line, sample) float64 -156.0 -156.1 … -157.1 -157.2
|           latitude           (line, sample) float64 22.11 22.12 22.13 … 20.57 20.58
|           height             (line, sample) float64 9.791e-05 9.635e-05 … 6.201e-05
|           azimuthTime        (line, sample) datetime64[ns] 2022-05-07T16:24:11.657260 …
|           slantRangeTime     (line, sample) float64 0.005331 0.005347 … 0.005651
```

(continues on next page)

(continued from previous page)

```
│           incidenceAngle  (line, sample) float64 30.8 31.13 31.45 … 36.34 36.59
│           elevationAngle  (line, sample) float64 27.49 27.77 28.05 … 32.28 32.5
│       Attributes:
│           footprint:         POLYGON ((-156.007554522451 22.11305598776757, -156.82…
│           coverage:          190km * 86km (line * sample )
│           approx_transform:  |-0.00,-0.00,-156.02|\n|-0.00, 0.00, 22.12|\n| 0.00, 0…
│           history:           annotations
├── DataTree('bursts')
│       Dimensions:           (burst: 10, line: 1490)
│       Dimensions without coordinates: burst, line
│       Data variables:
│           azimuthTime       (burst) datetime64[ns] 2022-05-07T16:24:11.657511 … 2…
│           azimuthAnxTime    (burst) float64 2.603e+03 2.606e+03 … 2.628e+03
│           sensingTime       (burst) datetime64[ns] 2022-05-07T16:24:12.795957 … 2…
│           byteOffset        (burst) int64 119515 122692875 … 980706395 1103279755
│           firstValidSample  (burst, line) float64 nan nan nan nan … nan nan nan nan
│           lastValidSample   (burst, line) float64 nan nan nan nan … nan nan nan nan
│           linesPerBurst     int64 1490
│           samplesPerBurst   int64 20566
│       Attributes:
│           history:  annotations
├── DataTree('FMrate')
│       Dimensions:                  (azimuthTime: 11)
│       Coordinates:
│         * azimuthTime              (azimuthTime) datetime64[ns] 2022-05-07T16:24:10…
│       Data variables:
│           t0                       (azimuthTime) float64 0.005331 … 0.005331
│           azimuthFmRatePolynomial  (azimuthTime) object -2336.74129411 + 449131.687…
│       Attributes:
│           history:  annotations
├── DataTree('doppler_estimate')
│       Dimensions:                        (azimuthTime: 11, nb_fine_dce: 20)
│       Coordinates:
│         * azimuthTime                    (azimuthTime) datetime64[ns] 2022-05-07T16:…
│         * nb_fine_dce                    (nb_fine_dce) int64 0 1 2 3 4 … 16 17 18 19
│       Data variables:
│           t0                             (azimuthTime) float64 0.005332 … 0.005332
│           geometryDcPolynomial           (azimuthTime) object -1.940496 + 203.4306·x…
│           dataDcPolynomial               (azimuthTime) object -29.09492 - 13406.99·x…
│           fineDceAzimuthStartTime        (azimuthTime) datetime64[ns] 2022-05-07T16:…
│           fineDceAzimuthStopTime         (azimuthTime) datetime64[ns] 2022-05-07T16:…
│           dataDcRmsError                 (azimuthTime) float64 1.826 2.185 … 10.56
│           slantRangeTime                 (azimuthTime, nb_fine_dce) float64 0.005338…
│           frequency                      (azimuthTime, nb_fine_dce) float64 -27.52 …
│           dataDcRmsErrorAboveThreshold   (azimuthTime) bool False False … False False
│       Attributes:
│           history:  annotations
├── DataTree('orbit')
│       Dimensions:     (time: 17)
│       Coordinates:
│         * time        (time) datetime64[ns] 2022-05-07T16:23:08.178010 … 2022-05-…
│       Data variables:
│           velocity_x  (time) float64 -3.562e+03 -3.499e+03 … -2.568e+03 -2.499e+03
│           velocity_y  (time) float64 -53.72 -14.41 24.81 63.93 … 486.6 524.1 561.6
│           velocity_z  (time) float64 -6.71e+03 -6.743e+03 … -7.131e+03 -7.153e+03
│           position_x  (time) float64 -5.632e+06 -5.668e+06 … -6.093e+06 -6.119e+06
│           position_y  (time) float64 -3.046e+06 -3.046e+06 … -3.01e+06 -3.005e+06
```

(continues on next page)

(continued from previous page)

```
│           position_z  (time) float64 3.003e+06 2.936e+06 … 1.963e+06 1.892e+06
│       Attributes:
│           orbit_pass:        Descending
│           platform_heading:  -167.7204378813183
│           frame:             Earth Fixed
│           history:           orbit:\n  annotation/s1a.xml:\n  - //product/generalAn…
├── DataTree('image')
│       Dimensions:                  (dim_0: 2)
│       Dimensions without coordinates: dim_0
│       Data variables: (12/14)
│           LineUtcTime              (dim_0) datetime64[ns] 2022-05-07T16:24:11.65751…
│           numberOfLines            int64 14900
│           numberOfSamples          int64 20566
│           azimuthPixelSpacing      float64 14.0
│           slantRangePixelSpacing   float64 2.33
│           groundRangePixelSpacing  float64 4.178
│           …                       …
│           slantRangeTime           float64 0.005331
│           swath_subswath           <U3 'IW1'
│           radarFrequency           float64 5.405e+09
│           rangeSamplingRate        float64 6.435e+07
│           azimuthSteeringRate      float64 1.59
│           history                  <U824 'image:\n  annotation/s1a.xml:\n  - /produ…
├── DataTree('calibration')
│       Dimensions:      (line: 30, sample: 516, pol: 2)
│       Coordinates:
│         * line         (line) int64 0 486 973 1608 2095 … 13978 14464 15099 15585
│         * sample       (sample) int64 0 40 80 120 160 … 20480 20520 20560 20565
│         * pol          (pol) object 'VV' 'VH'
│       Data variables:
│           sigma0_lut   (pol, line, sample) float64 331.1 331.0 330.9 … 306.9 306.9
│           gamma0_lut   (pol, line, sample) float64 306.8 306.7 306.6 … 274.9 274.9
│           azimuthTime  (pol, line) datetime64[ns] 2022-05-07T16:24:11.657511 … 20…
│       Attributes:
│           history:  calibration
├── DataTree('noise_range')
│       Dimensions:      (line: 11, sample: 516, pol: 2)
│       Coordinates:
│         * line         (line) int64 -1490 0 1490 2980 4470 … 8940 10430 11920 13558
│         * sample       (sample) int64 0 40 80 120 160 … 20480 20520 20560 20565
│         * pol          (pol) object 'VV' 'VH'
│       Data variables:
│           noise_lut    (pol, line, sample) float64 461.8 459.1 456.4 … 404.7 404.9
│           azimuthTime  (pol, line) datetime64[ns] 2022-05-07T16:24:11.657511 … 20…
│       Attributes:
│           history:  noise
├── DataTree('noise_azimuth')
│       Dimensions:       (line: 1500, pol: 2)
│       Coordinates:
│         * line          (line) int64 0 10 20 30 40 … 14860 14870 14880 14890 14899
│         * pol           (pol) object 'VV' 'VH'
│       Data variables:
│           noise_lut     (pol, line) float64 1.17 1.165 1.161 … 1.138 1.143 1.146
│           line_start    (pol) int64 0 0
│           line_stop     (pol) int64 14899 14899
│           sample_start  (pol) int64 0 0
│           sample_stop   (pol) int64 20565 20565
```

(continues on next page)

(continued from previous page)

```
│       Attributes:
│           history:  noise
├── DataTree('antenna_pattern')
│       Dimensions:          (swath_nb: 1, dim_slantRangeTime: 645, dim_azimuthTime:␣
↪11)
│       Coordinates:
│         * swath_nb         (swath_nb) int64 1
│       Dimensions without coordinates: dim_slantRangeTime, dim_azimuthTime
│       Data variables:
│           slantRangeTime   (swath_nb, dim_slantRangeTime) float64 0.005331 … 0.005651
│           swath            (swath_nb, dim_azimuthTime) float64 1.0 1.0 1.0 … 1.0 1.0
│           roll             (swath_nb, dim_azimuthTime) float64 30.22 30.22 … 30.23
│           azimuthTime      (swath_nb, dim_azimuthTime) datetime64[ns] 2022-05-07T16:…
│           terrainHeight    (swath_nb, dim_azimuthTime) float64 0.0 0.0 … 2.023 0.0
│           elevationAngle   (swath_nb, dim_azimuthTime, dim_slantRangeTime) float64 2…
│           incidenceAngle   (swath_nb, dim_azimuthTime, dim_slantRangeTime) float64 3…
│           gain             (swath_nb, dim_azimuthTime, dim_slantRangeTime) float64 1…
│       Attributes:
│           dim_azimuthTime:     max dimension of azimuthTime for a swath
│           dim_slantRangeTime:  max dimension of slantRangeTime for a swath
│           comment:             The antenna pattern data set record contains a list …
│           example:             for example, if swath Y is smaller than swath X, use…
│           source:              Sentinel-1 Product Specification
│           history:             antenna_pattern:\n  annotation/s1a.xml:\n  - /produc…
├── DataTree('swath_merging')
└── DataTree('recalibration')
```

[7]:
```
import xsarslc.processing.xspectra
intra_xs_without_IR = xsarslc.processing.xspectra.compute_IW_subswath_intraburst_
↪xspectra(dt,
                                        tile_width={'sample': 5.e3, 'line': 5.e3},
                                        tile_overlap={'sample': -10.e3, 'line': -10.
↪e3},
                                        periodo_width = {'sample': 4000, 'line':␣
↪4000},
                                       periodo_overlap = {'sample': 2000, 'line':␣
↪2000},
                                       polarization='VV',dev=True)
intra_xs_without_IR
```

```
intrabursts: 100
↪%|████████████████████████████████████████████████████████████████████████████████
↪1/1 [00:06<00:00,  6.49s/it]
```

[7]:
```
<xarray.Dataset>
Dimensions:                    (tile_sample: 6, tile_line: 2, bt_thresh: 5,
                                0tau: 3, freq_line: 57, freq_sample: 446,
                                1tau: 2, 2tau: 1, c_sample: 2, c_line: 2,
                                k_gp: 4, phi_hf: 5, lambda_range_max_macs: 10)
Coordinates:
    pol                        <U2 'VV'
  * bt_thresh                  (bt_thresh) int64 5 50 100 150 200
    k_rg                       (tile_line, tile_sample, freq_sample) float64 0…
    k_az                       (freq_line) float64 -0.04409 -0.04251 … 0.04409
    line                       (tile_line) int64 209 1280
    sample                     (tile_line, tile_sample) int64 1274 4661 … 19158
    longitude                  (tile_line, tile_sample) float64 -156.1 … -156.8
```

(continues on next page)

(continued from previous page)

```
    latitude                   (tile_line, tile_sample) float64 22.1 … 22.09
  * k_gp                       (k_gp) int64 1 2 3 4
  * phi_hf                     (phi_hf) int64 1 2 3 4 5
  * lambda_range_max_macs      (lambda_range_max_macs) float64 50.0 … 275.0
Dimensions without coordinates: tile_sample, tile_line, 0tau, freq_line,
                                freq_sample, 1tau, 2tau, c_sample, c_line
Data variables: (12/27)
    incidence                  (tile_line, tile_sample) float64 31.21 … 36.22
    ground_heading             (tile_line, tile_sample) float64 -168.9 … -169.2
    burst                      (tile_line) int64 0 0
    sensing_time               (tile_line, tile_sample) datetime64[ns] 2022-05…
    sigma0                     (tile_line, tile_sample) float64 0.1109 … 0.0…
    nesz                       (tile_line, tile_sample) float64 0.003995 … 0…
    …                         …
    corner_line                (tile_line, c_line) int64 31 387 1102 1458
    corner_sample              (tile_line, tile_sample, c_sample) int64 718 …
    burst_corner_longitude     (tile_line, c_sample, c_line) float64 -156.0 …
    burst_corner_latitude      (tile_line, c_sample, c_line) float64 22.11 …
    cwave_params               (tile_line, tile_sample, k_gp, phi_hf, 2tau) float64 …
    macs                       (tile_line, tile_sample, lambda_range_max_macs, 2tau)␣
↪complex128 …
Attributes: (12/24)
    name:                     SENTINEL1_DS:/home/datawork-cersat-public/cache/p…
    short_name:               SENTINEL1_DS:S1A_IW_SLC__1SDV_20220507T162411_202…
    product:                  SLC
    safe:                     S1A_IW_SLC__1SDV_20220507T162411_20220507T162439_…
    swath:                    IW
    multidataset:             False
    …                        …
    radar_frequency:          5405000454.33435
    azimuth_time_interval:    0.002055556299999998
    tile_width_sample:        5000.0
    tile_width_line:          5000.0
    tile_overlap_sample:      -10000.0
    tile_overlap_line:        -10000.0
```

[8]:
```
IR_path = out_file
intra_xs_with_IR = xsarslc.processing.xspectra.compute_IW_subswath_intraburst_
↪xspectra(dt,
                                        tile_width={'sample': 5.e3, 'line': 5.e3},
                                        tile_overlap={'sample': -10.e3, 'line': -10.
↪e3},
                                        periodo_width = {'sample': 4000, 'line':␣
↪4000},
                                       periodo_overlap = {'sample': 2000, 'line':␣
↪2000},
                                       polarization='VV',dev=True,IR_path=IR_path)
intra_xs_with_IR
```

```
intrabursts: 100
↪%|████████████████████████████████████████████████████████████████████████████
↪1/1 [00:06<00:00,  6.92s/it]
```

[8]:
```
<xarray.Dataset>
Dimensions:                    (tile_sample: 6, tile_line: 2, bt_thresh: 5,
                                0tau: 3, freq_line: 57, freq_sample: 446,
                                1tau: 2, 2tau: 1, c_sample: 2, c_line: 2,
```

(continues on next page)

(continued from previous page)

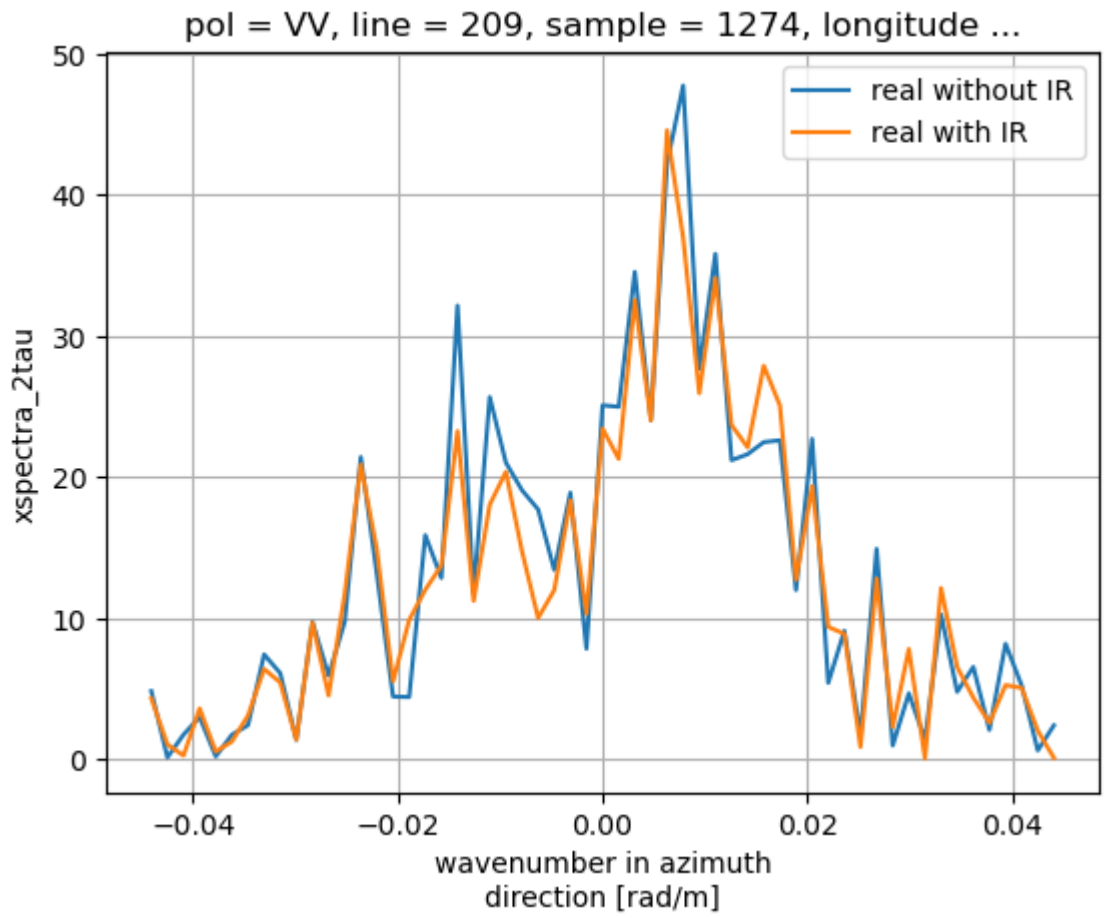
```
                                k_gp: 4, phi_hf: 5, lambda_range_max_macs: 10)
Coordinates:
    pol                        <U2 'VV'
  * bt_thresh                  (bt_thresh) int64 5 50 100 150 200
    k_rg                       (tile_line, tile_sample, freq_sample) float64 0…
    k_az                       (freq_line) float64 -0.04409 -0.04251 … 0.04409
    line                       (tile_line) int64 209 1280
    sample                     (tile_line, tile_sample) int64 1274 4661 … 19158
    longitude                  (tile_line, tile_sample) float64 -156.1 … -156.8
    latitude                   (tile_line, tile_sample) float64 22.1 … 22.09
  * k_gp                       (k_gp) int64 1 2 3 4
  * phi_hf                     (phi_hf) int64 1 2 3 4 5
  * lambda_range_max_macs      (lambda_range_max_macs) float64 50.0 … 275.0
Dimensions without coordinates: tile_sample, tile_line, 0tau, freq_line,
                                freq_sample, 1tau, 2tau, c_sample, c_line
Data variables: (12/27)
    incidence                  (tile_line, tile_sample) float64 31.21 … 36.22
    ground_heading             (tile_line, tile_sample) float64 -168.9 … -169.2
    burst                      (tile_line) int64 0 0
    sensing_time               (tile_line, tile_sample) datetime64[ns] 2022-05…
    sigma0                     (tile_line, tile_sample) float64 0.1109 … 0.0…
    nesz                       (tile_line, tile_sample) float64 0.003995 … 0…
    …                         …
    corner_line                (tile_line, c_line) int64 31 387 1102 1458
    corner_sample              (tile_line, tile_sample, c_sample) int64 718 …
    burst_corner_longitude     (tile_line, c_sample, c_line) float64 -156.0 …
    burst_corner_latitude      (tile_line, c_sample, c_line) float64 22.11 …
    cwave_params               (tile_line, tile_sample, k_gp, phi_hf, 2tau) float64 …
    macs                       (tile_line, tile_sample, lambda_range_max_macs, 2tau)␣
↪complex128 …
Attributes: (12/24)
    name:                    SENTINEL1_DS:/home/datawork-cersat-public/cache/p…
    short_name:              SENTINEL1_DS:S1A_IW_SLC__1SDV_20220507T162411_202…
    product:                 SLC
    safe:                    S1A_IW_SLC__1SDV_20220507T162411_20220507T162439_…
    swath:                   IW
    multidataset:            False
    …                       …
    radar_frequency:         5405000454.33435
    azimuth_time_interval:   0.002055556299999998
    tile_width_sample:       5000.0
    tile_width_line:         5000.0
    tile_overlap_sample:     -10000.0
    tile_overlap_line:       -10000.0
```

[9]:
```
tile_line_i = 0
tile_sample_i = 0
onespec_without_IR = intra_xs_without_IR['xspectra_2tau'].isel({'tile_sample':tile_
↪sample_i,'tile_line':tile_line_i}).squeeze()
onespec_without_IR = onespec_without_IR.swap_dims({'freq_line': 'k_az', 'freq_sample':
↪ 'k_rg'})
abs(onespec_without_IR.real.mean(dim='k_rg')).plot(label='real without IR')

onespec_with_IR = intra_xs_with_IR['xspectra_2tau'].isel({'tile_sample':tile_sample_i,
↪'tile_line':tile_line_i}).squeeze()
onespec_with_IR = onespec_with_IR.swap_dims({'freq_line': 'k_az', 'freq_sample': 'k_rg
```

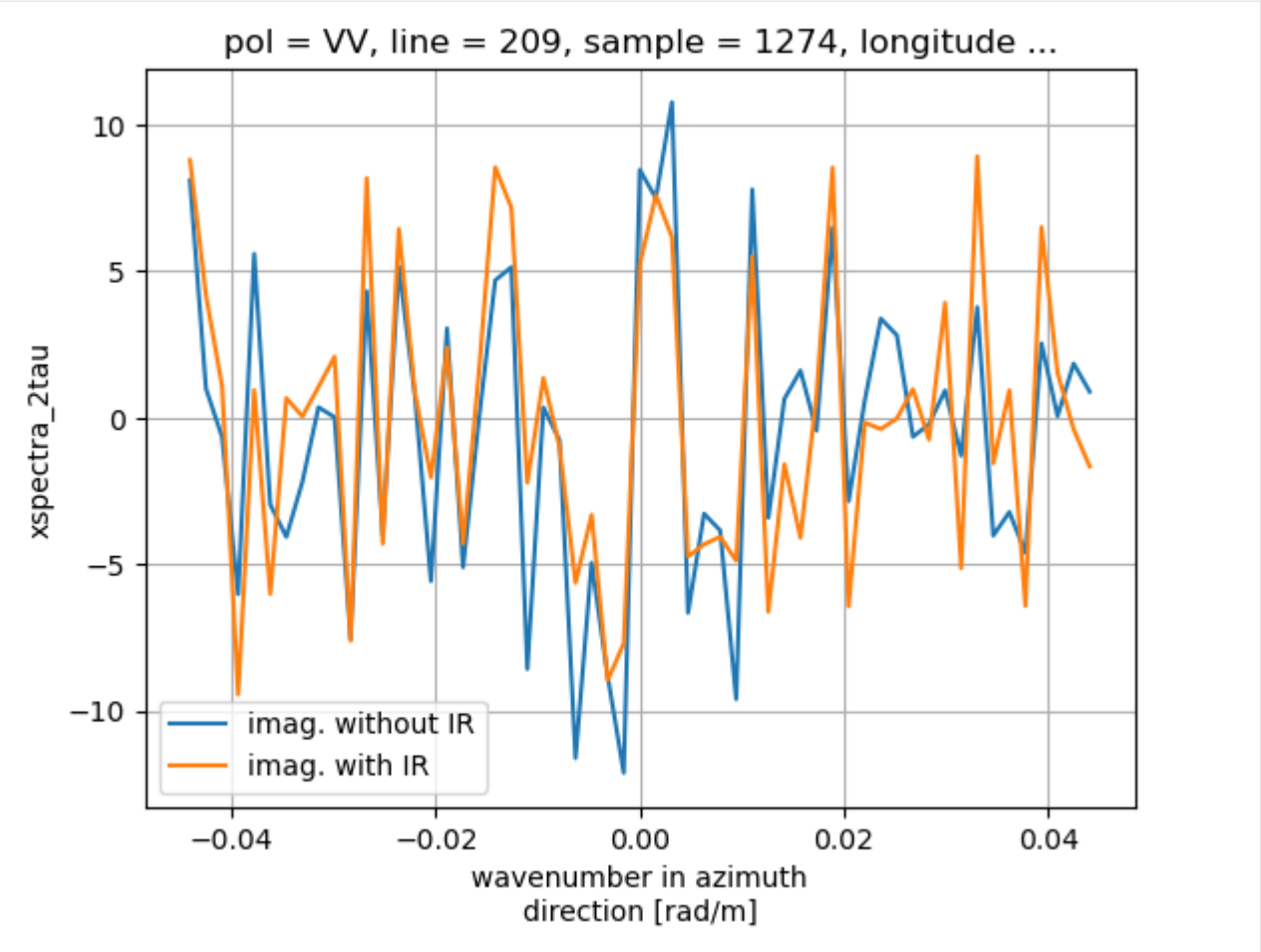
(continues on next page)

(continued from previous page)

```
↪'})
abs(onespec_with_IR.real.mean(dim='k_rg')).plot(label='real with IR')
plt.legend()
plt.grid(True)
```



[10]:
```
onespec_without_IR.imag.mean(dim='k_rg').plot(label='imag. without IR')
onespec_with_IR.imag.mean(dim='k_rg').plot(label='imag. with IR')
plt.legend()
plt.grid(True)
```



Download this notebook from github[14].

0.2.10 A notebook to illustrate how to compute and correct Impulse Response in WV SLC product

[1]:
```
"""
This is an example to show how to compute sensor Impulse Response (RI) on WV SLC data
and then correct it.
"""
```

[1]:
```
'\nThis is an example to show how to compute sensor Impulse Response (RI) on WV SLC␣
↪data\nand then correct it.\n'
```

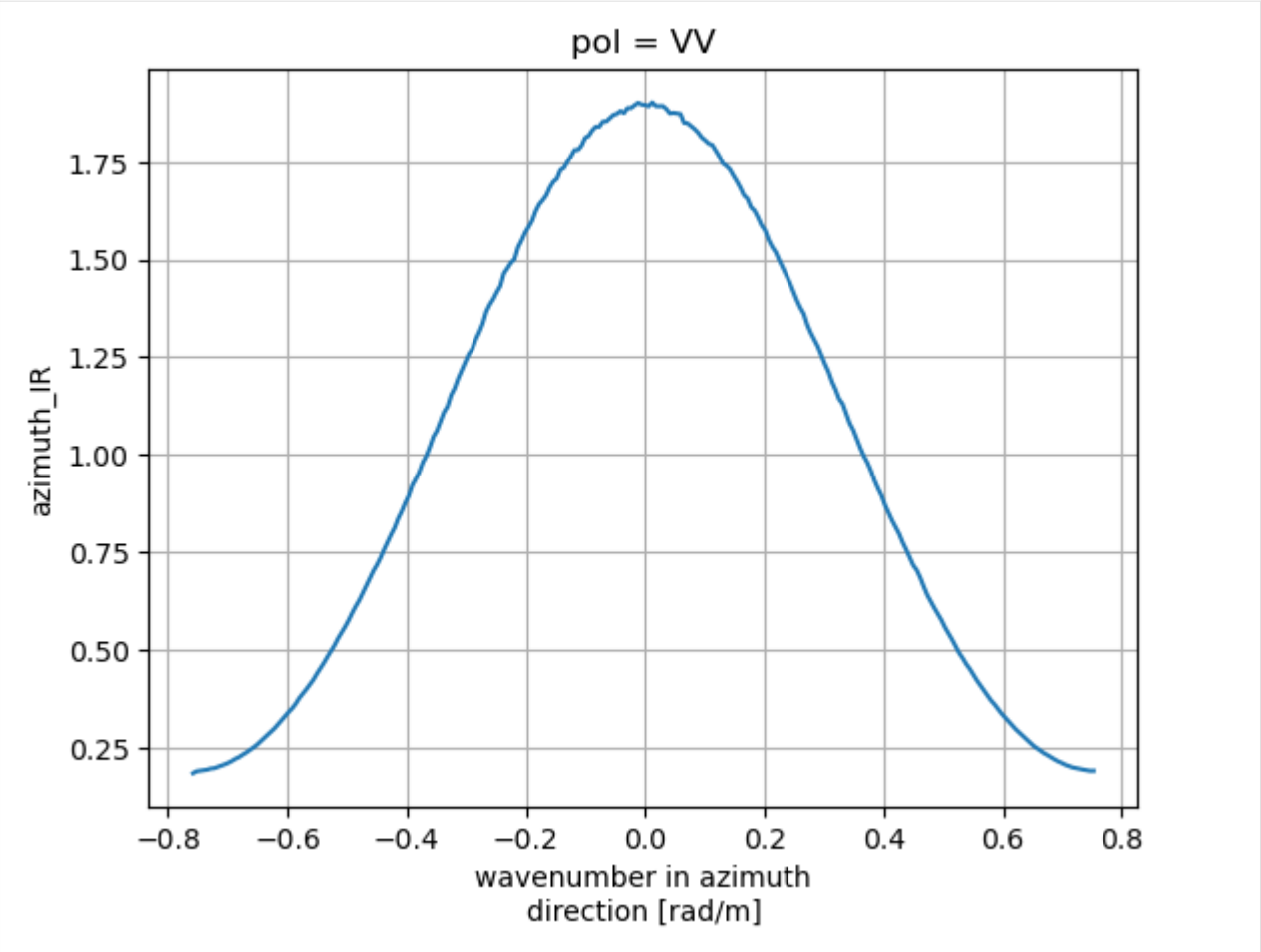
[14] https://github.com/umr-lops/xsar_slc/tree/main/docs/examples/example_WV_compute_and_correct_from_impulse_response.ipynb

read a WV SLC over homogeneous region in the rain forest

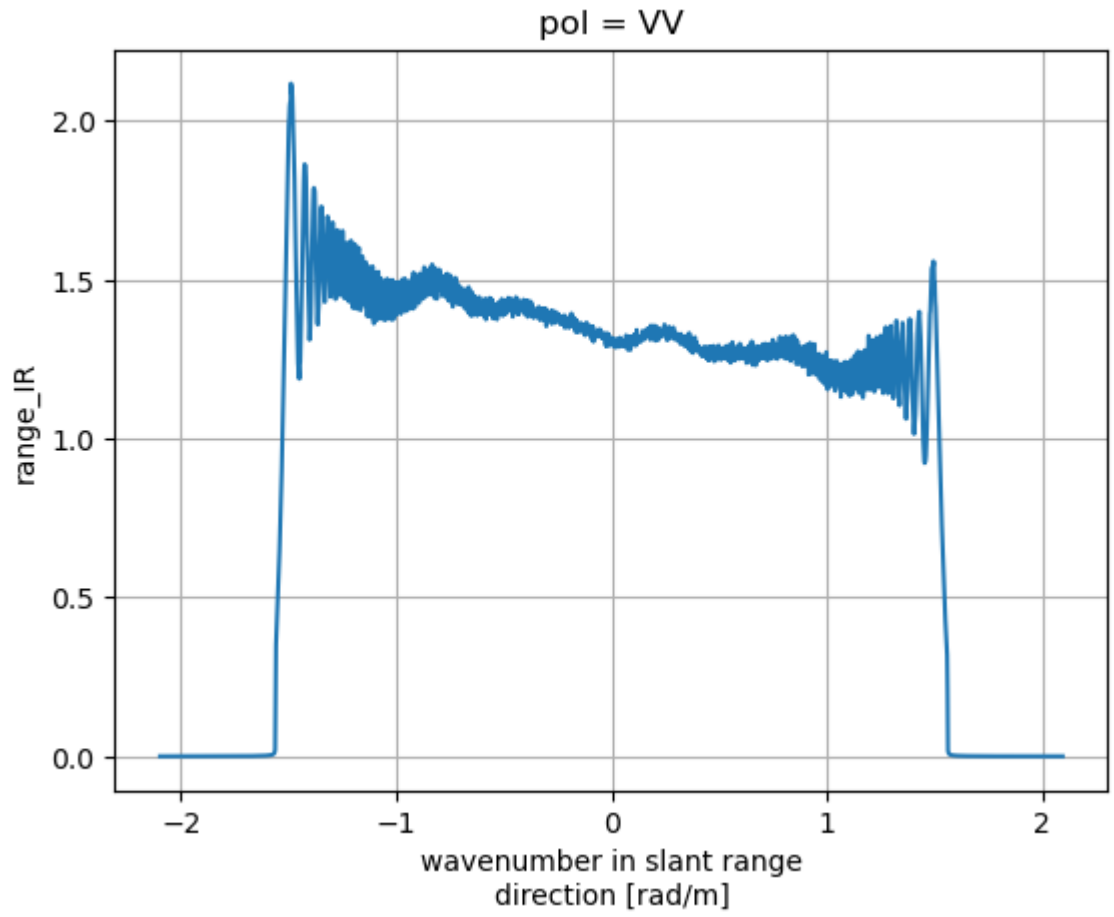
[2]:
```
import warnings
warnings.filterwarnings("ignore")
from xsarslc.processing.impulseResponse import generate_WV_AUX_file_ImpulseReponse
# SENTINEL1_DS:/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/
↪sentinel-1a/L1/WV/S1A\_WV\_SLC\_\_1S/2018/050/S1A\_WV\_SLC\_\_1SSV\_20180219T221522\
↪_20180219T222851\_020681\_0236CE\_8F55.SAFE:WV_051
subswathes = {'/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/
↪sentinel-1a/L1/WV/S1A_WV_SLC__1S/2018/050/S1A_WV_SLC__1SSV_20180219T221522_
↪20180219T222851_020681_0236CE_8F55.SAFE':['051'],
              '/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/
↪sentinel-1a/L1/WV/S1A_WV_SLC__1S/2018/062/S1A_WV_SLC__1SSV_20180303T221522_
↪20180303T222851_020856_023C59_7FBF.SAFE':['051']}

IRs = generate_WV_AUX_file_ImpulseReponse(subswathes)
IRs
```

[2]:
```
<xarray.Dataset>
Dimensions:          (k_srg: 2048, k_az: 256)
Coordinates:
    pol              <U2 'VV'
  * k_srg            (k_srg) float64 -2.098 -2.096 -2.094 … 2.092 2.094 2.096
  * k_az             (k_az) float64 -0.7575 -0.7516 -0.7457 … 0.7457 0.7516
Data variables:
    range_IR         (k_srg) float64 0.0004371 0.0004377 … 0.0004373 0.0004375
    mean_incidence   float64 23.96
    azimuth_IR       (k_az) float64 0.1835 0.1885 0.1899 … 0.191 0.1896 0.1897
Attributes:
    long_name:               Impulse response in range direction
    product:                 SLC
    swath:                   WV
    multidataset:            False
    ipf:                     2.84
    platform:                SENTINEL-1A
    pols:                    VV
    coverage:                20km * 20km (line * sample )
    orbit_pass:              Ascending
    radar_frequency:         5405000704.0
    azimuth_time_interval:   0.0006059988896297211
```

[3]:
```
from matplotlib import pyplot as plt
IRs['azimuth_IR'].plot()
plt.grid(True)
```



[4]:
```
IRs['range_IR'].plot()
plt.grid(True)
```



write a netCDF AUX IR file

[5]:
```
out_file = "/tmp/S1_IR_WV_AUX_FILE.nc"
if 'footprint' in IRs.attrs:
    IRs.attrs.update({'footprint': str(IRs.footprint)})
if 'multidataset' in IRs.attrs:
    IRs.attrs.update({'multidataset': str(IRs.multidataset)})
IRs.to_netcdf(out_file)
print(out_file)
IRs.to_netcdf(out_file)
print(out_file)
```

```
/tmp/S1_IR_WV_AUX_FILE.nc
/tmp/S1_IR_WV_AUX_FILE.nc
```

apply IR correction when computing cross spectra on a WV image

[6]:
```
out_file = "/tmp/S1_IR_WV_AUX_FILE.nc"
import os
import xsar
import warnings
warnings.filterwarnings("ignore")
#a_wv_safe = '/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/
↪sentinel-1a/L1/WV/S1A_WV_SLC__1S/2023/030/S1A_WV_SLC__1SSV_20230130T095609_
↪20230130T100149_047011_05A391_E93C.SAFE'
a_wv_tiff = '/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/
↪sentinel-1a/L1/WV/S1A_WV_SLC__1S/2023/030/S1A_WV_SLC__1SSV_20230130T095609_
↪20230130T100149_047011_05A391_E93C.SAFE/measurement/s1a-wv1-slc-vv-20230130t100004-
↪20230130t100007-047011-05a391-017.tiff'
fullpathsafeSLC = os.path.dirname(os.path.dirname(a_wv_tiff))
print(fullpathsafeSLC,os.path.exists(fullpathsafeSLC))

imagette_number = os.path.basename(a_wv_tiff).split('-')[-1].replace('.tiff','')
str_gdal = 'SENTINEL1_DS:%s:WV_%s'%(fullpathsafeSLC,imagette_number)
print('str_gdal',str_gdal)
s1ds = xsar.Sentinel1Dataset(str_gdal)
s1ds
```

```
/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/esa/sentinel-1a/L1/WV/
↪S1A_WV_SLC__1S/2023/030/S1A_WV_SLC__1SSV_20230130T095609_20230130T100149_047011_
↪05A391_E93C.SAFE True
str_gdal SENTINEL1_DS:/home/datawork-cersat-public/cache/project/mpc-sentinel1/data/
↪esa/sentinel-1a/L1/WV/S1A_WV_SLC__1S/2023/030/S1A_WV_SLC__1SSV_20230130T095609_
↪20230130T100149_047011_05A391_E93C.SAFE:WV_017
```

[6]:
```
<Sentinel1Dataset full coverage object>
```

[7]:
```
slc = s1ds.datatree['measurement'].to_dataset()['digital_number']
slc
```

[7]:
```
<xarray.DataArray 'digital_number' (pol: 1, line: 4907, sample: 5571)>
dask.array<open_rasterio-034951840d7c4223938c19b8c7bd4376<this-array>, shape=(1, 4907,
↪ 5571), dtype=complex64, chunksize=(1, 4907, 5000), chunktype=numpy.ndarray>
Coordinates:
  * line     (line) int64 0 1 2 3 4 5 6 7 … 4900 4901 4902 4903 4904 4905 4906
  * sample   (sample) int64 0 1 2 3 4 5 6 … 5564 5565 5566 5567 5568 5569 5570
  * pol      (pol) object 'VV'
Attributes:
    comment:  denoised digital number, read at full resolution
    history:  digital_number: measurement/s1a-wv1-slc-vv-20230130t100004-2023…
```

[8]:
```
s1ds.dataset
```

[8]:
```
<xarray.Dataset>
Dimensions:         (line: 4907, sample: 5571, pol: 1)
Coordinates:
  * line            (line) int64 0 1 2 3 4 5 6 … 4901 4902 4903 4904 4905 4906
  * sample          (sample) int64 0 1 2 3 4 5 … 5565 5566 5567 5568 5569 5570
  * pol             (pol) object 'VV'
Data variables:
    digital_number  (pol, line, sample) complex64 dask.array<chunksize=(1, 4907,␣
↪5000), meta=np.ndarray>
    time            (line) datetime64[ns] 2023-01-30T10:00:04.057309952 … 2…
    sampleSpacing   float64 1.498
```

(continues on next page)

(continued from previous page)

```
    lineSpacing     float64 4.144
Attributes: (12/15)
    name:               SENTINEL1_DS:/home/datawork-cersat-public/cache/projec…
    short_name:         SENTINEL1_DS:S1A_WV_SLC__1SSV_20230130T095609_20230130…
    product:            SLC
    safe:               S1A_WV_SLC__1SSV_20230130T095609_20230130T100149_04701…
    swath:              WV
    multidataset:       False
    …                  …
    start_date:         2023-01-30 10:00:04.057288
    stop_date:          2023-01-30 10:00:07.030318
    footprint:          POLYGON ((130.822283515509 -41.56791070616844, 131.065…
    coverage:           20km * 21km (line * sample )
    orbit_pass:         Ascending
    platform_heading:   -13.8309692949357
```

[9]:
```
# from xsarslc.processing.xspectra.compute_subswath_intraburst_xspectra import␣
↪compute_intraburst_xspectrum
# compute_intraburst_xspectrum(slc, mean_incidence, slant_spacing, azimuth_spacing,␣
↪synthetic_duration,
#                              azimuth_dim='line', nperseg={'sample': 512, 'line':
↪ 512},
#                              noverlap={'sample': 256, 'line': 256}, IR_path=out_
↪file)
```

compute xspec without IR correction

[10]:
```
from xsarslc.processing.xspectra import compute_WV_intraburst_xspectra
import time
t0 = time.time()
xs0 = compute_WV_intraburst_xspectra(dt=s1ds.datatree,
                                     polarization='VV',periodo_width={"line":
↪3000,"sample":3000},
                                     periodo_overlap={"line":-3000,"sample":-
↪3000})
print('time to compute x-spectra on WV :%1.1f sec'%(time.time()-t0))
xs0
```

```
time to compute x-spectra on WV :23.6 sec
```

[10]:
```
<xarray.Dataset>
Dimensions:                   (bt_thresh: 5, 0tau: 3, freq_line: 145,
                               freq_sample: 397, 1tau: 2, 2tau: 1, c_sample: 2,
                               c_line: 2, k_gp: 4, phi_hf: 5,
                               lambda_range_max_macs: 10)
Coordinates:
    pol                       <U2 'VV'
  * bt_thresh                 (bt_thresh) int64 5 50 100 150 200
    k_rg                      (freq_sample) float64 0.0 0.002094 … 0.829
    k_az                      (freq_line) float64 -0.1506 -0.1485 … 0.1506
    line                      int64 2453
    sample                    int64 2785
    longitude                 float64 130.9
    latitude                  float64 -41.45
```

(continues on next page)

(continued from previous page)

```
  * k_gp                       (k_gp) int64 1 2 3 4
  * phi_hf                     (phi_hf) int64 1 2 3 4 5
  * lambda_range_max_macs      (lambda_range_max_macs) float64 50.0 … 275.0
Dimensions without coordinates: 0tau, freq_line, freq_sample, 1tau, 2tau,
                                c_sample, c_line
Data variables: (12/26)
    incidence                  float64 23.31
    ground_heading             float64 -15.36
    sensing_time               datetime64[ns] 2023-01-30T10:00:05.543825664
    sigma0                     float64 0.368
    nesz                       float64 0.001408
    bright_targets_histogram   (bt_thresh) int64 539 0 0 0 0
    …                         …
    corner_line                (c_line) int64 25 4881
    corner_sample              (c_sample) int64 25 5545
    burst_corner_longitude     (c_sample, c_line) float64 130.8 130.8 131.1 131.0
    burst_corner_latitude      (c_sample, c_line) float64 -41.57 … -41.34
    cwave_params               (k_gp, phi_hf, 2tau) float64 7.757 … -0.1919
    macs                       (lambda_range_max_macs, 2tau) complex128 (0.329…
Attributes: (12/21)
    name:                      SENTINEL1_DS:/home/datawork-cersat-public/cache/p…
    short_name:                SENTINEL1_DS:S1A_WV_SLC__1SSV_20230130T095609_202…
    product:                   SLC
    safe:                      S1A_WV_SLC__1SSV_20230130T095609_20230130T100149_…
    swath:                     WV
    multidataset:              False
    …                         …
    radar_frequency:           5405000704.0
    azimuth_time_interval:     0.0006059988896297211
    tile_width_line:           20125.523484
    tile_width_sample:         <xarray.DataArray 'cumulative ground length' ()>\…
    tile_overlap_sample:       0.0
    tile_overlap_line:         0.0
```

compute xspec with IR corretion

[11]:
```
import time
import xsarslc.processing.xspectra
from importlib import reload
reload(xsarslc.processing.xspectra)
t0 = time.time()
xs1 = xsarslc.processing.xspectra.compute_WV_intraburst_xspectra(dt=s1ds.datatree,
                                        polarization='VV',periodo_width={"line":
↪3000,"sample":3000},
                                        periodo_overlap={"line":-3000,"sample":-
↪3000},IR_path=out_file)
print('time to compute x-spectra on WV :%1.1f sec'%(time.time()-t0))
xs1
```

```
time to compute x-spectra on WV :23.9 sec
```

[11]:
```
<xarray.Dataset>
Dimensions:                    (bt_thresh: 5, 0tau: 3, freq_line: 145,
                                freq_sample: 397, 1tau: 2, 2tau: 1, c_sample: 2,
                                c_line: 2, k_gp: 4, phi_hf: 5,
```

(continues on next page)

(continued from previous page)

```
                                lambda_range_max_macs: 10)
Coordinates:
    pol                        <U2 'VV'
  * bt_thresh                  (bt_thresh) int64 5 50 100 150 200
    k_rg                       (freq_sample) float64 0.0 0.002094 … 0.829
    k_az                       (freq_line) float64 -0.1506 -0.1485 … 0.1506
    line                       int64 2453
    sample                     int64 2785
    longitude                  float64 130.9
    latitude                   float64 -41.45
  * k_gp                       (k_gp) int64 1 2 3 4
  * phi_hf                     (phi_hf) int64 1 2 3 4 5
  * lambda_range_max_macs      (lambda_range_max_macs) float64 50.0 … 275.0
Dimensions without coordinates: 0tau, freq_line, freq_sample, 1tau, 2tau,
                                c_sample, c_line
Data variables: (12/26)
    incidence                  float64 23.31
    ground_heading             float64 -15.36
    sensing_time               datetime64[ns] 2023-01-30T10:00:05.543825664
    sigma0                     float64 0.368
    nesz                       float64 0.001408
    bright_targets_histogram   (bt_thresh) int64 539 0 0 0 0
    …                          …
    corner_line                (c_line) int64 25 4881
    corner_sample              (c_sample) int64 25 5545
    burst_corner_longitude     (c_sample, c_line) float64 130.8 130.8 131.1 131.0
    burst_corner_latitude      (c_sample, c_line) float64 -41.57 … -41.34
    cwave_params               (k_gp, phi_hf, 2tau) float64 7.733 … -0.1993
    macs                       (lambda_range_max_macs, 2tau) complex128 (0.327…
Attributes: (12/21)
    name:                      SENTINEL1_DS:/home/datawork-cersat-public/cache/p…
    short_name:                SENTINEL1_DS:S1A_WV_SLC__1SSV_20230130T095609_202…
    product:                   SLC
    safe:                      S1A_WV_SLC__1SSV_20230130T095609_20230130T100149_…
    swath:                     WV
    multidataset:              False
    …                          …
    radar_frequency:           5405000704.0
    azimuth_time_interval:     0.0006059988896297211
    tile_width_line:           20125.523484
    tile_width_sample:         <xarray.DataArray 'cumulative ground length' ()>\…
    tile_overlap_sample:       0.0
    tile_overlap_line:         0.0
```

compare with/without

[12]:
```
import numpy as np
from xsarslc.processing import xspectra
xs = xs0.swap_dims({'freq_line': 'k_az', 'freq_sample': 'k_rg'})
xs = xspectra.symmetrize_xspectrum(xs['xspectra_2tau'], dim_range='k_rg', dim_azimuth=
↪'k_az')

xs_ir = xs1.swap_dims({'freq_line': 'k_az', 'freq_sample': 'k_rg'})
xs_ir = xspectra.symmetrize_xspectrum(xs_ir['xspectra_2tau'], dim_range='k_rg', dim_
↪azimuth='k_az')
```

(continues on next page)

(continued from previous page)

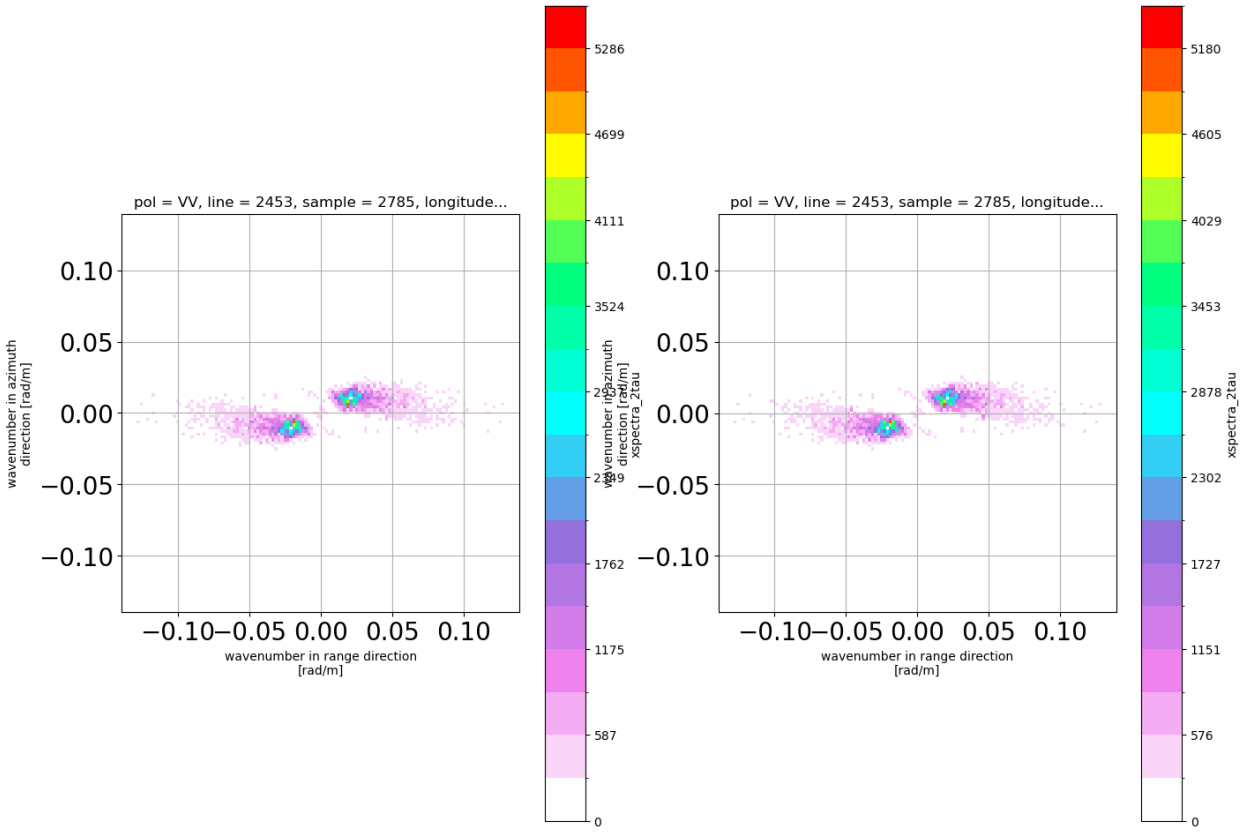
```
########################################## real part ##########################
coS=np.abs(xs.mean(dim='2tau').real) # co spectrum = real part of cross-spectrum
coS_reduced = coS.where(np.logical_and(np.abs(coS.k_rg)<=0.14, np.abs(coS.k_az)<=0.
↪14), drop=True)

coS_IRcor=np.abs(xs_ir.mean(dim='2tau').real) # co spectrum = real part of cross-
↪spectrum
coS_reduced_IRcor = coS_IRcor.where(np.logical_and(np.abs(coS_IRcor.k_rg)<=0.14, np.
↪abs(coS_IRcor.k_az)<=0.14), drop=True)
from matplotlib import pyplot as plt
%matplotlib inline
from matplotlib import colors as mcolors
cmap = mcolors.LinearSegmentedColormap.from_list("", ["white","violet","mediumpurple",
↪"cyan","springgreen","yellow","red"])
PuOr = plt.get_cmap('PuOr')
plt.figure(figsize=(16,12))
plt.subplot(1,2,1)
#coS_reduced_av=coS_reduced.rolling(k_rg=3, center=True).mean().rolling(k_az=3,␣
↪center=True).mean()
coS_reduced.plot(cmap=cmap, levels=20, vmin=0)

plt.grid()
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.axis('scaled')

plt.subplot(1,2,2)
#coS_reduced_av=coS_reduced.rolling(k_rg=3, center=True).mean().rolling(k_az=3,␣
↪center=True).mean()
plt.title('with IR correction')
coS_reduced_IRcor.plot(cmap=cmap, levels=20, vmin=0)

plt.grid()
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.axis('scaled')

########################################## imag.part ##########################
ims = xs.mean(dim='2tau').imag.squeeze()
xS_red = ims.where(np.logical_and(np.abs(ims.k_rg)<=0.14, np.abs(ims.k_az)<=0.14),␣
↪drop=True)
ims_irCor = xs_ir.mean(dim='2tau').imag.squeeze()
xS_red_IRCor = ims_irCor.where(np.logical_and(np.abs(ims_irCor.k_rg)<=0.14, np.
↪abs(ims_irCor.k_az)<=0.14), drop=True)
#xS_av=xS_red.rolling(k_rg=3, center=True).mean().rolling(k_az=3, center=True).mean()

plt.figure(figsize=(16,12))
plt.subplot(1,2,1)
xS_red.plot(x='k_rg',cmap=PuOr)
plt.grid()
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.axis('scaled')

plt.subplot(1,2,2)
xS_red_IRCor.plot(x='k_rg',cmap=PuOr)
```

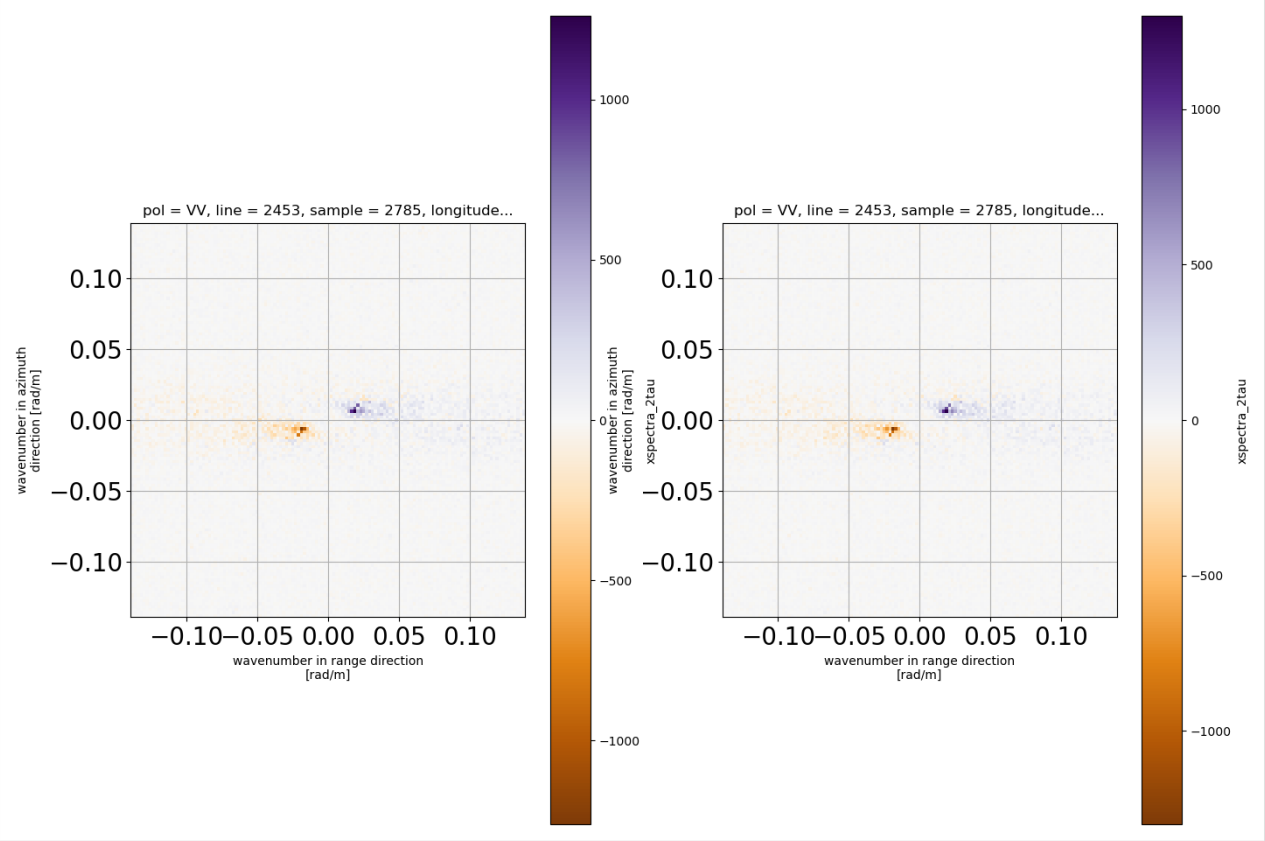
(continues on next page)

(continued from previous page)

```
plt.grid()
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.axis('scaled')
```

[12]:
```
(-0.13921871536677022,
 0.13921871536677022,
 -0.1390863226967172,
 0.1390863226967172)
```





display difference with / without IR correction

[13]:
```
import numpy as np
from xsarslc.processing import xspectra
xs = xs0.swap_dims({'freq_line': 'k_az', 'freq_sample': 'k_rg'})
xs = xspectra.symmetrize_xspectrum(xs['xspectra_2tau'], dim_range='k_rg', dim_azimuth=
↪'k_az')

xs_ir = xs1.swap_dims({'freq_line': 'k_az', 'freq_sample': 'k_rg'})
xs_ir = xspectra.symmetrize_xspectrum(xs_ir['xspectra_2tau'], dim_range='k_rg', dim_
↪azimuth='k_az')

########################################## real part ############################
coS=np.abs(xs.mean(dim='2tau').real) # co spectrum = real part of cross-spectrum
coS_reduced = coS.where(np.logical_and(np.abs(coS.k_rg)<=0.14, np.abs(coS.k_az)<=0.
↪14), drop=True)

coS_IRcor=np.abs(xs_ir.mean(dim='2tau').real) # co spectrum = real part of cross-
↪spectrum
coS_reduced_IRcor = coS_IRcor.where(np.logical_and(np.abs(coS_IRcor.k_rg)<=0.14, np.
↪abs(coS_IRcor.k_az)<=0.14), drop=True)
from matplotlib import pyplot as plt
%matplotlib inline
from matplotlib import colors as mcolors
cmap = mcolors.LinearSegmentedColormap.from_list("", ["white","violet","mediumpurple",
↪"cyan","springgreen","yellow","red"])
```

(continues on next page)

(continued from previous page)

```
PuOr = plt.get_cmap('PuOr')
plt.figure(figsize=(8,6))
plt.subplot(1,1,1)
#coS_reduced_av=coS_reduced.rolling(k_rg=3, center=True).mean().rolling(k_az=3,␣
↪center=True).mean()
di = coS_reduced-coS_reduced_IRcor
print(di)
di.plot(cmap=cmap)

plt.grid()
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.axis('scaled')


########################################## imag.part ############################
ims = xs.mean(dim='2tau').imag.squeeze()
xS_red = ims.where(np.logical_and(np.abs(ims.k_rg)<=0.14, np.abs(ims.k_az)<=0.14),␣
↪drop=True)
ims_irCor = xs_ir.mean(dim='2tau').imag.squeeze()
xS_red_IRCor = ims_irCor.where(np.logical_and(np.abs(ims_irCor.k_rg)<=0.14, np.
↪abs(ims_irCor.k_az)<=0.14), drop=True)
#xS_av=xS_red.rolling(k_rg=3, center=True).mean().rolling(k_az=3, center=True).mean()

plt.figure(figsize=(8,6))
plt.subplot(1,1,1)
(xS_red-xS_red_IRCor).plot(x='k_rg',cmap=PuOr)
plt.grid()
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.axis('scaled')
```

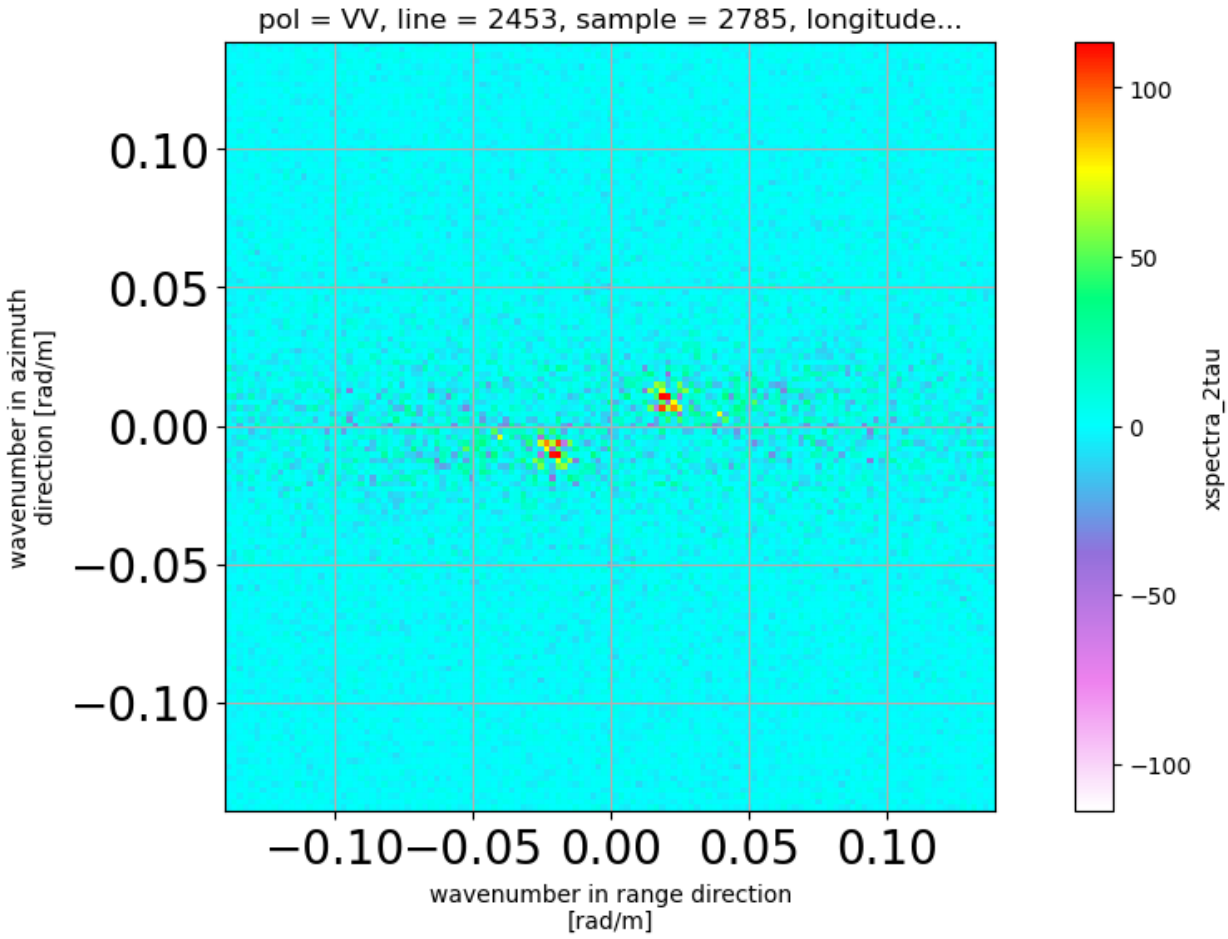
```
<xarray.DataArray 'xspectra_2tau' (k_az: 133, k_rg: 133)>
array([[-3.30231023,  0.20230055,  3.24634354, …, -0.28420991,
        -5.80216624, -0.82587412],
       [-1.03039672,  4.10552075,  0.85178918, …, -1.66087963,
        -2.70627495,  3.32467915],
       [ 0.68772213,  0.37675369, -4.81396219, …, -1.86558608,
         0.48049893, -1.52556207],
       …,
       [-1.52556207,  0.48049893, -1.86558608, …, -4.81396219,
         0.37675369,  0.68772213],
       [ 3.32467915, -2.70627495, -1.66087963, …,  0.85178918,
         4.10552075, -1.03039672],
       [-0.82587412, -5.80216624, -0.28420991, …,  3.24634354,
         0.20230055, -3.30231023]])
Coordinates:
  * k_az       (k_az) float64 -0.138 -0.1359 -0.1339 … 0.1339 0.1359 0.138
  * k_rg       (k_rg) float64 -0.1382 -0.1361 -0.134 … 0.134 0.1361 0.1382
    pol        <U2 'VV'
    line       int64 2453
    sample     int64 2785
    longitude  float64 130.9
    latitude   float64 -41.45
```

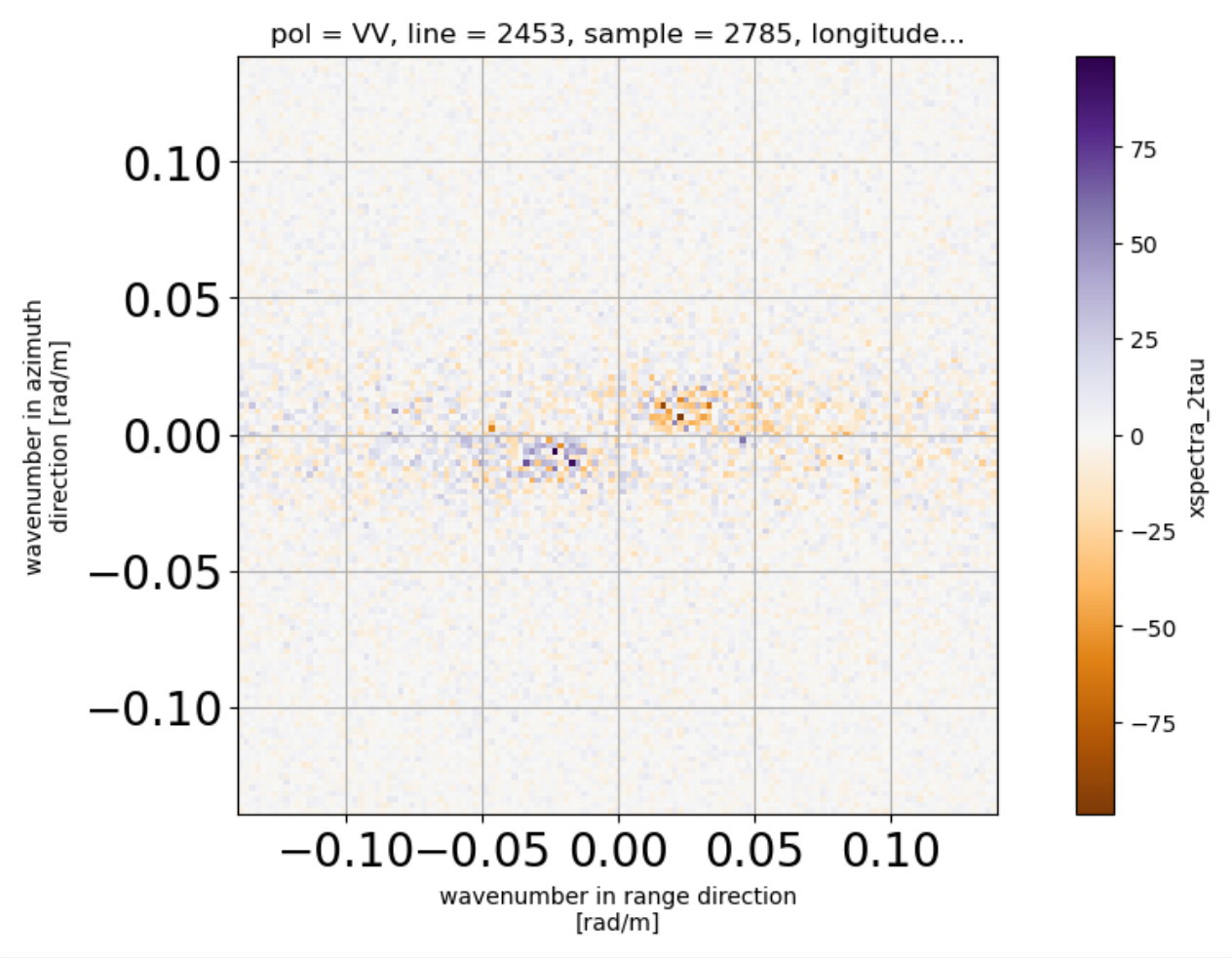
[13]:
```
(-0.13921871536677022,
 0.13921871536677022,
```

(continues on next page)

(continued from previous page)

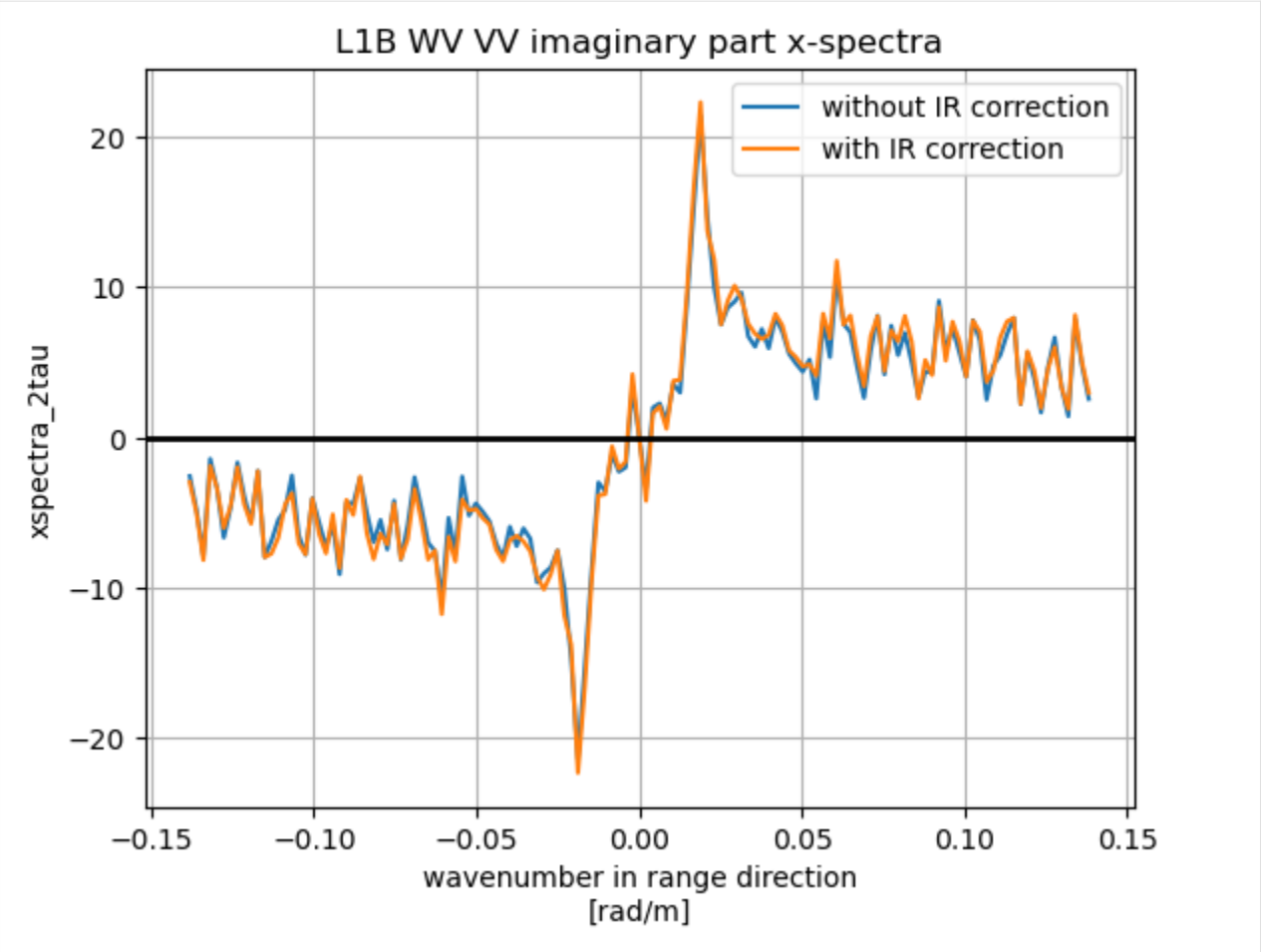
```
-0.1390863226967172,
0.1390863226967172)
```





[14]:
```
xS_red.mean(dim='k_az').plot(label='without IR correction')
xS_red_IRCor.mean(dim='k_az').plot(label='with IR correction')
plt.grid(True)
plt.legend()
plt.title('L1B WV VV imaginary part x-spectra')
plt.axhline(y=0,c='k',lw=2)
```

[14]:
```
<matplotlib.lines.Line2D at 0x7f30d6202530>
```



Download this notebook from github[15].

0.2.11 illustration of the default Impulse Response (IR) files from `xsarslc`

[1]:
```
import xsarslc.get_config_infos
from importlib import reload
reload(xsarslc.get_config_infos)
import holoviews as hv
hv.extension('bokeh')
import xarray as xr
import os
```

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

Data type cannot be displayed: application/vnd.holoviews_load.v0+json, application/javascript

[15] https://github.com/umr-lops/xsar_slc/tree/main/docs/examples/default_impulseResponse_files_IW.ipynb

Data type cannot be displayed: text/html, application/vnd.holoviews_exec.v0+json

[2]:
```
lst_nc = []
for unit in ['S1A','S1B']:
    for sw in ['IW1','IW2','IW3']:
        for pol in ['VV']:
            fileir = xsarslc.get_config_infos.get_IR_file(unit=unit, subswath=sw,␣
↪polarization=pol)
            lst_nc.append(fileir)
            print(fileir)
```

```
/home1/datahome/agrouaze/sources/git/xsar_slc/auxdata/S1A_IRs_IW1_VV.nc
/home1/datahome/agrouaze/sources/git/xsar_slc/auxdata/S1A_IRs_IW2_VV.nc
/home1/datahome/agrouaze/sources/git/xsar_slc/auxdata/S1A_IRs_IW3_VV.nc
/home1/datahome/agrouaze/sources/git/xsar_slc/auxdata/S1B_IRs_IW1_VV.nc
/home1/datahome/agrouaze/sources/git/xsar_slc/auxdata/S1B_IRs_IW2_VV.nc
/home1/datahome/agrouaze/sources/git/xsar_slc/auxdata/S1B_IRs_IW3_VV.nc
```

display IR in range dimension

[3]:
```
vari =  'range_IR'
all_cu = []
filter_out = ['HH','HV','VH']
for ff in lst_nc:
    go = True
    for fi in filter_out:
        if fi in ff:
            print(ff,'filtered out with',fi)
            go = False
        else:
            pass
    if go:
        ds = xr.open_dataset(ff)

        tmpcu = hv.Curve(ds[vari],label=os.path.basename(ff)).opts(tools=['hover'],
↪show_legend=True)
        all_cu.append(tmpcu)
print(len(all_cu))
hv.Overlay(all_cu).opts(width=1000,height=750,tools=['hover'],show_grid=True,show_
↪legend=True)
```

```
6
```

[3]:
```
:Overlay
   .Curve.S1A_IRs_IW1_VV_full_stop_nc :Curve   [k_srg]   (range_IR)
   .Curve.S1A_IRs_IW2_VV_full_stop_nc :Curve   [k_srg]   (range_IR)
   .Curve.S1A_IRs_IW3_VV_full_stop_nc :Curve   [k_srg]   (range_IR)
   .Curve.S1B_IRs_IW1_VV_full_stop_nc :Curve   [k_srg]   (range_IR)
   .Curve.S1B_IRs_IW2_VV_full_stop_nc :Curve   [k_srg]   (range_IR)
   .Curve.S1B_IRs_IW3_VV_full_stop_nc :Curve   [k_srg]   (range_IR)
```

display IR in azimuth dimension

[4]:
```
vari = 'azimuth_IR'
all_cu = []
filter_out = ['HH','HV','VH']
for ff in lst_nc:
    go = True
    for fi in filter_out:
        if fi in ff:
            print(ff,'filtered out with',fi)
            go = False
        else:
            pass
    if go:
        ds = xr.open_dataset(ff)

        tmpcu = hv.Curve(ds[vari],label=os.path.basename(ff)).opts(tools=['hover'],
↪show_legend=True)
        all_cu.append(tmpcu)
print(len(all_cu))
hv.Overlay(all_cu).opts(width=1000,height=750,tools=['hover'],show_grid=True,show_
↪legend=True)
```

```
6
```

[4]:
```
:Overlay
   .Curve.S1A_IRs_IW1_VV_full_stop_nc :Curve   [k_az]   (azimuth_IR)
   .Curve.S1A_IRs_IW2_VV_full_stop_nc :Curve   [k_az]   (azimuth_IR)
   .Curve.S1A_IRs_IW3_VV_full_stop_nc :Curve   [k_az]   (azimuth_IR)
   .Curve.S1B_IRs_IW1_VV_full_stop_nc :Curve   [k_az]   (azimuth_IR)
   .Curve.S1B_IRs_IW2_VV_full_stop_nc :Curve   [k_az]   (azimuth_IR)
   .Curve.S1B_IRs_IW3_VV_full_stop_nc :Curve   [k_az]   (azimuth_IR)
```

0.2.12 API reference

processing

xsarslc.processing.xspectra.**compute_subswath_xspectra**(*dt*, *polarization*, *tile_width_intra*, *tile_width_inter*, *tile_overlap_intra*, *tile_overlap_inter*, *periodo_width_intra*, *periodo_width_inter*, *periodo_overlap_intra*, *periodo_overlap_inter*, ***kwargs*)

Main function to compute IW inter and intra burst spectra. It has to be modified to be able to change Xspectra options

Parameters

- **dt** (*xarray.Datatree*) – datatree contraining subswath information
- **tile_width_intra** (*dict, optional*) – approximate sizes of tiles in meters. Dict of shape {dim_name (str): width of tile [m](float)} for intra burst.
- **tile_width_inter** (*dict, optional*) – approximate sizes of tiles in meters. Dict of shape {dim_name (str): width of tile [m](float)} for inter burst.

- **tile_overlap_intra** (*dict, optional*) – approximate sizes of tiles overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)} for intra burst.
- **tile_overlap_inter** (*dict, optional*) – approximate sizes of tiles overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)} for inter burst.
- **periodo_width_intra** (*dict*) – approximate sizes of periodogram in meters. Dict of shape {dim_name (str): width of tile [m](float)} for intra burst.
- **periodo_width_inter** (*dict*) – approximate sizes of periodogram in meters. Dict of shape {dim_name (str): width of tile [m](float)} for inter burst.
- **periodo_overlap_intra** (*dict*) – approximate sizes of periodogram overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)} for intra burst.
- **periodo_overlap_inter** (*dict*) – approximate sizes of periodogram overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)} for inter burst.
- **polarization** (*str, optional*) – polarization to be selected for xspectra computation

Keyword Arguments

kwargs (*dict*) – keyword arguments passed to called functions. landmask, IR_path …

xsarslc.processing.xspectra.**compute_IW_subswath_intraburst_xspectra**(*dt*, *polarization*, *periodo_width={'line': 2000, 'sample': 2000}*, *periodo_overlap={'line': 1000, 'sample': 1000}*, *tile_width={'line': 20000.0, 'sample': 20000.0}*, *tile_overlap={'line': 10000.0, 'sample': 10000.0}*, *landmask=None*, *IR_path=None*, ***kwargs*)

Compute IW subswath intra-burst xspectra per tile Note: If requested tile is larger than the size of availabe data. tile will be set to maximum available size

Parameters

- **dt** (*xarray.Datatree*) – datatree contraining subswath information
- **polarization** (*str, optional*) – polarization to be selected for xspectra computation
- **tile_width** (*dict*) – approximative sizes of tiles in meters. Dict of shape {dim_name (str): width of tile [m](float)}
- **tile_overlap** (*dict*) – approximative sizes of tiles overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)}
- **polarization** – polarization to be selected for xspectra computation

- **periodo_width** (*dict*) – approximate sizes of periodogram in meters. Dict of shape {dim_name (str): width of tile [m](float)}
- **periodo_overlap** (*dict*) – approximate sizes of periodogram overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)}
- **landmask** (*cartopy, optional*) – a landmask to be used for land discrimination
- **IR_path** (*str, optional*) – a path to the Impulse Response file

Keyword Arguments
kwargs (*dict*) – optional keyword arguments : burst_list (list), dev (bool) are valid entries

Returns
xspectra.

Return type
(xarray)

xsarslc.processing.xspectra.**compute_IW_subswath_interburst_xspectra**(*dt*, *polarization*, *periodo_width={'line': 1200, 'sample': 2000}*, *periodo_overlap={'line': 600, 'sample': 1000}*, *tile_width={'line': 1500.0, 'sample': 20000.0}*, *tile_overlap={'line': 750.0, 'sample': 10000.0}*, *landmask=None*, *IR_path=None*, ***kwargs*)

Compute IW subswath inter-burst xspectra. No deramping is applied since only magnitude is used.

Note: If requested tile is larger than the size of availabe data. tile will be set to maximum available size Note: The overlap is short in azimuth (line) direction. Keeping nperseg = {'line':None} in Xspectra computation keeps maximum number of point in azimuth but is not ensuring the same number of overlapping point for all burst

Parameters

- **dt** (*xarray.Datatree*) – datatree containing sub-swath information
- **polarization** (*str, optional*) – polarization to be selected for xspectra computation
- **tile_width** (*dict*) – approximate sizes of tiles in meters. Dict of shape {dim_name (str): width of tile [m](float)}
- **tile_overlap** (*dict*) – approximate sizes of tiles overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)}
- **polarization** – polarization to be selected for xspectra computation
- **periodo_width** (*dict*) – approximate sizes of periodogram in meters. Dict of shape {dim_name (str): width of tile [m](float)}

- **periodo_overlap** (*dict*) – approximate sizes of periodogram overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)}
- **landmask** (*cartopy, optional*) – a landmask to be used for land discrimination
- **IR_path** (*str, optional*) – a path to the Impulse Response file

Keyword Arguments
kwargs (*dict*) – optional keyword arguments : burst_list (list), dev (bool) are valid entries

Returns
xspectra.

Return type
(xarray)

xsarslc.processing.xspectra.**compute_modulation**(*ds*, *lowpass_width*, *spacing*)

Compute modulation map (sig0/low_pass_filtered_sig0)

Parameters

- **ds** (*xarray*) – array of (deramped) digital number
- **lowpass_width** (*dict*) – form {name of dimension (str): width in [m] (float)}. width for low pass filtering [m]
- **spacing** (*dict*) – form {name of dimension (str): spacing in [m] (float)}. spacing for each filtered dimension

xsarslc.processing.xspectra.**compute_azimuth_cutoff**(*spectrum*, *definition='drfab'*)

compute azimuth cutoff :param spectrum: Xspectrum with coordinates k_rg and k_az :type spectrum: xarray :param definition: ipf (covariance is averaged over range) or drfab (covariance taken at range = 0) :type definition: str, optional

Returns
azimuth cutoff [m]

Return type
(float)

xsarslc.processing.intraburst.**tile_burst_to_xspectra**(*burst*, *geolocation_annotation*, *orbit*, *calibration*, *noise_range*, *noise_azimuth*, *tile_width*, *tile_overlap*, *lowpass_width={'line': 4750.0, 'sample': 4750.0}*, *periodo_width={'line': 4000.0, 'sample': 4000.0}*, *periodo_overlap={'line': 2000.0, 'sample': 2000.0}*, *landmask=None*, *IR_path=None*, ***kwargs*)

Divide burst in tiles and compute intra-burst cross-spectra using compute_intraburst_xspectrum() function.

Parameters

- **burst** (*xarray.Dataset*) – dataset with deramped digital number variable
- **geolocation_annotation** (*xarray.Dataset*) – dataset of geolocation annotation
- **orbit** (*xarray.Dataset*) – dataset of orbit annotation
- **tile_width** (*dict*) – approximative sizes of tiles in meters. Dict of shape {dim_name (str): width of tile [m](float)}

- **tile_overlap** (*dict*) – approximative sizes of tiles overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)}
- **periodo_width** (*dict*) – approximative sizes of periodogram in meters. Dict of shape {dim_name (str): width of tile [m](float)}
- **periodo_overlap** (*dict*) – approximative sizes of periodogram overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)}
- **lowpass_width** (*dict*) – width for low pass filtering [m]. Dict of form {dim_name (str): width (float)}
- **landmask** (*optional*) – If provided, land mask passed to is_ocean(). Otherwise xspectra are calculated by default
- **IR_path** (*str, optional*) – a path to the Impulse Response file

Keyword Arguments

- **landmask** (*optional*) – If provided, land mask passed to is_ocean(). Otherwise xspectra are calculated by default
- **kwargs** – keyword arguments passed to compute_intraburst_xspectrum()

xsarslc.processing.intraburst.**compute_intraburst_xspectrum**(*slc*, *mean_incidence*, *slant_spacing*, *azimuth_spacing*, *synthetic_duration*, *azimuth_dim='line'*, *nperseg={'line': 512, 'sample': 512}*, *noverlap={'line': 256, 'sample': 256}*, *IR_path=None*, ***kwargs*)

Compute SAR cross spectrum using a 2D Welch method. Looks are centered on the mean Doppler frequency If ds contains only one cycle, spectrum wavenumbers are added as coordinates in returned DataSet, otherwise, they are passed as variables (k_range, k_azimuth).

Parameters

- **slc** (*xarray*) – digital numbers of Single Look Complex image.
- **mean_incidence** (*float*) – mean incidence on slc
- **slant_spacing** (*float*) – slant spacing
- **azimuth_spacing** (*float*) – azimuth spacing
- **synthetic_duration** (*float*) – synthetic aperture duration (to compute tau)
- **azimuth_dim** (*str*) – name of azimuth dimension
- **nperseg** (*dict of int*) – number of point per periodogram. Dict of form {dimension_name(str): number of point (int)}
- **noverlap** (*dict of int*) – number of overlapping point per periodogram. Dict of form {dimension_name(str): number of point (int)}

Keyword Arguments

kwargs (*dict*) – keyword arguments passed to compute_looks()

Returns

SLC cross_spectra

Return type
(xarray)

xsarslc.processing.intraburst.**compute_looks**(*slc*, *azimuth_dim*, *synthetic_duration*, *nlooks=3*, *look_width=0.2*, *look_overlap=0.0*, *look_window=None*, ***kwargs*)

Compute the N looks of an slc DataArray. Spatial coverage of the provided slc must be small enough to enure an almost constant ground spacing. Meaning: ground_spacing ~= slant_spacing / sin(mean_incidence)

Parameters

- **slc** (*xarray.DataArray*) – (bi-dimensional) array to process
- **azimuth_dim** (*str*) – name of the azimuth dimension (dimension used to extract the look)
- **nlooks** (*int*) – number of look
- **look_width** (*float*) – in [0,1.] width of a look [percentage of full bandwidth] (nlooks*look_width must be < 1)
- **look_overlap** (*float*) – in [0,1.] look overlapping [percentage of a look]
- **look_window** (*str or tuple*) – instance that can be passed to scipy.signal.get_window()

Returns
keys are '0tau', '1tau', … and values are list of corresponding computed spectra

Return type
(dict)

xsarslc.processing.interburst.**tile_bursts_overlap_to_xspectra**(*burst0*, *burst1*, *geolocation_annotation*, *calibration*, *noise_range*, *noise_azimuth*, *tile_width*, *tile_overlap*, *lowpass_width={'line': 4750.0, 'sample': 4750.0}*, *periodo_width={'line': 1200.0, 'sample': 2000.0}*, *periodo_overlap={'line': 600.0, 'sample': 1000.0}*, *landmask=None*, *IR_path=None*, ***kwargs*)

Divide bursts overlaps in tiles and compute inter-burst cross-spectra using compute_interburst_xspectrum() function.

Parameters

- **burst0** (*xarray.Dataset*) – first burst (in time) dataset (No need of deramped digital number variable)
- **burst1** (*xarray.Dataset*) – second burst (in time) dataset (No need of deramped digital number variable)
- **geolocation_annotation** (*xarray.Dataset*) – dataset of geolocation annotation

- **tile_width** (*dict*) – approximative sizes of tiles in meters. Dict of shape {dim_name (str): width of tile [m](float)}
- **tile_overlap** (*dict*) – approximative sizes of tiles overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)}
- **lowpass_width** (*dict*) – width for low pass filtering [m]. Dict of form {dim_name (str): width (float)}
- **landmask** (*optional*) – If provided, land mask passed to is_ocean(). None -> xspectra are calculated by default. True -> no xspectra computed.
- **IR_path** (*str, optional*) – a path to the Impulse Response file

Keyword Arguments
kwargs – keyword arguments passed to compute_interburst_xspectrum()

xsarslc.processing.interburst.**compute_interburst_xspectrum**(*mod0*, *mod1*, *mean_incidence*, *slant_spacing*, *azimuth_spacing*, *azimuth_dim='line'*, *nperseg={'line': None, 'sample': 512}*, *noverlap={'line': 0, 'sample': 256}*, ***kwargs*)

Compute cross spectrum between mod0 and mod1 using a 2D Welch method (periodograms).

Parameters

- **mod0** (*xarray*) – modulation signal from burst0
- **mod1** (*xarray*) – modulation signal from burst1
- **mean_incidence** (*float*) – mean incidence on slc
- **slant_spacing** (*float*) – slant spacing
- **azimuth_spacing** (*float*) – azimuth spacing
- **azimuth_dim** (*str*) – name of azimuth dimension
- **nperseg** (*dict of int*) – number of point per periodogram. Dict of form {dimension_name(str): number of point (int)}
- **noverlap** (*dict of int*) – number of overlapping point per periodogram. Dict of form {dimension_name(str): number of point (int)}

Returns
concatenated cross_spectra

Return type
(xarray)

xsarslc.burst.**burst_valid_indexes**(*ds*)

Find indexes of valid portion of a burst. Returned line index are relative to burst only !

Parameters
ds (*xarray.Dataset*) – Dataset of one burst

Returns
index of (first valid sample, last valid sample, first valid line, last valid line)

Return type
(tuple of int)

xsarslc.burst.**crop_IW_burst**(*ds*, *burst_annotation*, *burst_number*, *valid=True*, *merge_burst_annotation=True*)

Crop IW burst from the measurement dataset

Parameters

- **ds** (*xarray.Dataset*) – measurement dataset
- **burst_annotation** (*xarray.dataset*) – burst annotation dataset
- **burst_number** (*int*) – burst number
- **valid** (*bool, optional*) – If true: only return the valid part of the burst
- **merge_burst_annotation** (*bool*) – If true: annotation of the burst are added to the returned dataset

Returns
extraction of valid burst portion of provided datatree

Return type
xarray.Dataset

xsarslc.burst.**deramp_burst**(*burst*, *dt*)

Deramp burst. Return deramped digital numbers

Parameters

- **burst** (*xarray.dataArray or xarray.Dataset*) – burst or portion of a burst
- **dt** (*xarray.dataTree*) – datatree containing all informations of the SLC

Returns
deramped digital numbers

Return type
(xarray.DataArray)

xsarslc.burst.**crop_WV**(*burst*)

Crop WV data. Removes portion of WV with zero values only and removes 25 points on each side to remove windowing effect

xsarslc.processing.impulseResponse.**compute_IWS_subswath_Impulse_Response**(*dt*, *burst_list=None*, *tile_width={'line': 20000.0, 'sample': 20000.0}*, *tile_overlap={'line': 10000.0, 'sample': 10000.0}*, *polarization='VV'*, ***kwargs*)

Compute IWS sub-swath range and azimuth Impulse Response. This function must be applied on homogeneous zone (Amazonia, …) Note: If requested tile is larger than the size of available data. tile will be set to maximum available size :param dt: datatree containing sub-swath information :type dt: xarray.Datatree :param burst_list: list of burst number to process. Default is all :type burst_list: list of int, optional :param tile_width: approximate sizes of tiles in meters. Dict of shape {dim_name (str): width of tile [m](float)} :type tile_width: dict :param tile_overlap: approximate sizes of tiles overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)} :type tile_overlap: dict :param polarization: polarization to be selected for IR computation :type polarization: str, optional

Keyword Arguments
kwargs (*dict*) – keyword arguments passed to tile_burst_to_IR(), land-mask can be added in kwargs. Can contain polarization

Returns
xspectra.

Return type
(xarray)

xsarslc.processing.impulseResponse.**compute_WV_Impulse_Response**(*dt*, *tile_width=None*, *tile_overlap=None*, *polarization='VV'*, ***kwargs*)

Compute WV range and azimuth Impulse Response. This function must be applied on homogeneous zone (amazonia, …) Note: If requested tile is larger than the size of availabe data. tile will be set to maximum available size :param dt: datatree contraining subswath information :type dt: xarray.Datatree :param burst_list: list of burst number to process. Default is all :type burst_list: list of int, optional :param tile_width: approximative sizes of tiles in meters. Dict of shape {dim_name (str): width of tile [m](float)} :type tile_width: dict :param tile_overlap: approximative sizes of tiles overlapping in meters. Dict of shape {dim_name (str): overlap [m](float)} :type tile_overlap: dict :param polarization: polarization to be selected for IR computation :type polarization: str, optional

Keyword Arguments
kwargs (*dict*) – keyword arguments passed to tile_burst_to_IR(), landmask can be added in kwargs. Can contain polarisation

Returns
xspectra.

Return type
(xarray)

0.2.13 Bibliography

BIBLIOGRAPHY

[1] Harald Jonhsen, Fabrice Collard, Romain Husson, and Guillaume Hajduch. Sentinel-1 ocean swell wave spectra (osw) algorithm definition. *https://sentinel.esa.int/documents*, 2021.

[2] N Miranda. Definition of the tops slc deramping function for products generated by the s-1 ipf (technical note cope-gseg-eopg-tn-14-0025, issue 1, rev. 3). Technical Report, Tech. rep., European Space Agency, available at: https://earth. esa. int …, 2017.

[3] P.J. Meadows Nuno Miranda. Radiometric calibration of s-1 level-1 products generated by the s-1 ipf,esa-eopg-cscop-tn-0002. *https://sentinel.esa.int/documents*, 2015.

[4] Ricardo Piantanida, Nuno Miranda, N. Franceschi, and P. Meadows. Thermal denoising of products generated by the s-1 ipf. *https://sentinel.esa.int/documents*, 2017.

[5] Tom Patterson and Nathaniel Vaughn Kelso. Https://scitools.org.uk/cartopy/docs/latest/reference/generated/cartopy.feature.naturalearth Technical Report, https://www.naturalearthdata.com/downloads/10m-physical-vectors/10m-land/ through scipy python library under public licence CC0, 2024.

[6] Geir Engen and Harald Johnsen. Sar-ocean wave inversion using image cross spectra. *IEEE transactions on geoscience and remote sensing*, 33(4):1047–1056, 1995.

[7] Johannes Schulz-Stellenfleth, Thomas König, and Susanne Lehner. An empirical approach for the retrieval of integral ocean wave parameters from synthetic aperture radar data. *Journal of Geophysical Research: Oceans*, 2007.

[8] Huimin Li, Bertrand Chapron, Alexis Mouche, and Justin E Stopa. A new ocean sar cross-spectral parameter: definition and directional property using the global sentinel-1 measurements. *Journal of Geophysical Research: Oceans*, 124(3):1566–1577, 2019.

PYTHON MODULE INDEX

X