



Interferometry

Analysis and Modelling (Remote Sensing) Summer Semester 2023

Author: Emma Garcia Boadas

Abstract

This paper explores interferometry technique for Synthetic Aperture Radar (SAR) images, focusing on the application of free and open-source software. However, the complexity of SAR SLC images, challenges for beginners, remote sensing technicians and the general public. To address this issue, the EZ-InSAR interface has emerged as an open-source solution, using the ISCE+StaMPS/MintPy packages, though still relying on MATLAB, which is not free.

In this study, a successful Interferometric processing and Persistent Scatterer Interferometry (PSI) velocity estimation are demonstrated using a virtual machine free and open-source software. Specifically, we adopt SNAP2StaMPS for the Interferometric part and StaMPS for PSI, explained step by step. Additionally, we list various software and packages for Interferometry technique execution.

This study demonstrates the feasibility of conducting Interferometry with free software and offers valuable insights into the challenges of using open-source solutions. Future work involves exploring software parameter optimization, expanding the processing scope with other tools/software, and implementing Small Baseline Subset (SBAS) processing for the same region.

Index

Abstract	1
Introduction.....	3
Interferometry.....	3
Software	4
Methodology part 1 of the processing.....	6
Image downloading.....	6
Installing the architecture and Preprocessing Snap2StaMPS:	7
Image burst and swath selection:	8
Update the orbit metadata	8
Apply the orbits	10
Master slave co-registration and the interferogram generation	10
Methodology part2 of the processing.....	13
Dependencies and software to install:.....	13
Processing in StaMPS	23
Step1.....	25
Step2.....	28
Step3.....	29
Step4.....	29
Step 5.....	30
Step 6.....	31
Atmospheric correction.....	32
Step 7.....	33
Rerun the step 6 and 7	35
Discussion	39
Conclusion	40
Bibliography.....	41

Introduction

Synthetic Aperture Radar (SAR) images are less used and more inaccessible for the general public and remote sensing technicians. The SAR Single Look Complex (SLC) images are coherent and are composed by amplitude and phase. The first corresponds to the real number and the second to an imaginary number. Phase in SAR corresponds to the difference between the transmission and the signal reception. Phase is the relates the distance between the emisor and receptor and also, corresponds with the electromagnetics characteristics of the receptor. This condition makes SAR SLC images complex. However, there is little documentation and lack of user platforms to debug errors, complex installations dependencies and the unavailability of open source and free GUI makes it fairly inaccessible. It seems that the community is starting to realize about that and it has come out the new interface EZ-InSAR (Hrysiewicz et al., 2023) available in github (<https://github.com/alexisInSAR/EZ-InSAR>). This software with GUI is not completely open source because it uses MATLAB that is a payment software and foremost is based on using the open-source software packages (ISCE+StaMPS/MintPy).

In the following document, we will introduce the interferometry technique. Also, we will perform InSAR processing with a PSI (Persistent scatter interferometry), as a result of velocity. This process will be explained step by step including the installation and the dependencies needed and trying to use free, open source, as a matter of investigation. We will comment on different programs and packages to perform the interferometry technique.

The main objective is to try to perform an InSAR processing with free software on a virtual machine. In our case, we will try to do it with a SNAP2StaMPS for the Interferometric part and for the second part (the PSI) we will use StaMPS. This software is open source available online for download but, it uses Matlab, it means that it is not completely free. In order to be able to run StaMPS we need to run it in a Linux so we will be using the Virtual Machine (VM) of Terascope for the processing. A secondary objective is to compare different software models of insert techniques that are available in open source.

Interferometry

Interferometry is a technique that consists of executing the multiplication of one SLC image by the complex conjugate of the another SLC image(Bamler & Hartl, 1998). It is typically done between two or more Synthetic Aperture Radar (SAR) images which can be referred to as master (S1) and slave (S2). An SLC image can be recognized as 'S' in the following Formula 1.

$$S = A e^{i\phi}$$

Formula 1: SLC Image.

An Interferogram (I) can be defined mathematically by Formula 2.

$$I = S_1 S_2^* = A_1 A_2 \exp(i(\phi_1 - \phi_2))$$

Formula 2: Interferogram.

This technique is done usually avoiding the amplitude (A) and just taking in to account the phase of the signal encapsulated in the SAR SLC image Formula 3(Bamler & Hartl, 1998).

$$\varphi_{\text{int}} = \phi_2 - \phi_1$$

Formula 3: Interferogram phase part.

The phase is composed by a real number and an imaginary number. Also in SAR corresponds to the difference between the transmission and the signal reception. Related with distance between the emisor and receptor taking in to account the electromagnetics characteristics of the receptor. We can say that the phase is formed by the following Formula 4: First (green) phase due to the properties of reflection of the objects (depending on geometric properties and dielectric properties. Aleatory for distributed dispersors). Second (red) the geometric phase due to the distance to the surface or object/sensor. Third (purple) the phase contribution due to the ionospheric and atmospheric propagation. Finally (blue) the noise inherent to the system.

$$\varphi_1 = \phi_1 + \frac{4\pi}{\lambda}R + \alpha_1 + \eta_1$$

Formula 4: Phase components.

The Interferogram signal is formed by different components (Crosetto et al., 2016; Pepe & Calò, 2017) Formula 5, first (green) is the flat earth phase component, the second (red) topography phase component, third (yellow) movement or displacement phase and the last one (blue) is the atmosphere and noise. These components explain the characteristics of the interferogram and all the component that we see when there are analyzed.

$$\Delta\varphi_{\text{int}} = \frac{4\pi}{\lambda r_0} \frac{B_n \Delta r}{\tan \alpha} + \frac{4\pi}{\lambda r_0} \frac{B_n \Delta h}{\sin \alpha} + \frac{4\pi}{\lambda} \Delta\rho + \Delta\varphi_{\text{APS}} + \Delta\varphi_N$$

Formula 5: Interferogram signal.

To have the images ready to generate and interferogram is key to preprocess the images as they have to match. The matching is done with the co-registration of two SLC products, one master and one slave, of the sub swaths with the same satellite orbits and a digital elevation model to calibrate the orbits with one of reference. This process allows the generation of interferograms. This process will be explained with more detail for our example of processing in the methodology. Also, in the methodology, we will explain the exact process to extract the velocity of the interferogram in our processing workflow.

Software

The Software is a big part of the process as it is needed a long list of steps to perform this analysis. In the following lines you will find a list of software that have different functionalities to perform interferograms, obtain InSAR results and analyze time series. We will perform a PSI analysis and as we have the chance to have access to Matlab we will use SNAP and StaMPS. These two programs have a code that is called SNAP2StaMPS to make the process more agile and automatic.

Commercial software

InSAR processing

- ENVISARscape Time series PSI and SBAS (<https://www.l3harris.com/all-capabilities/envi-sarscape>)
- SARproz Time series PSI and SBAS (<https://www.sarproz.com/>)
- GAMMA Time series PSI and SBAS (<https://www.gamma-rs.ch/>)
- SARscape Time series PSI and SBAS (<https://www.sarmap.ch/index.php/software/sarscape/>)
- **StaMPS** (Matlab) Time series PSI and SBAS
(<https://homepages.see.leeds.ac.uk/~earahoo/stamps/>)

Free/Open sours

Interferometric process

- **StaMPS** (Matlab is required and it is not free) Time series PSI and SBAS
(<https://homepages.see.leeds.ac.uk/~earahoo/stamps/>)
- **GMTSAR** Interferometric process Time series SBAS (<https://github.com/gmtsar/gmtsar>)
- **SNAP (with GUI)** Interferometric process Time series SBAS
- **Doris** Delft object-oriented radar interferometric **DinSAR** (<http://doris.tudelft.nl/>)

Dem and orbits

- **ISCE** Interferometric process (<https://github.com/isce-framework/isce2#software-dependencies>) (<https://github.com/isce-framework/isce2>)
- **GIANT** insar analysis toolbox (<http://earthdef.caltech.edu/projects/giant/wiki>)

Time series analysis software

- **MintPy** (<https://github.com/insarlab/MintPy>) (ICEYE images)
- **Py rate** (<https://github.com/GeoscienceAustralia/PyRate>)
- **KFTS** (<https://github.com/ManonDls/KFTS-InSAR>)
- **MPITS** (<https://github.com/jolivet/mpits>)

Tropospheric Noise Correction Software

- **PyAPS** Python based Atmospheric Phase Screen Estimation.
(<https://github.com/insarlab/PyAPS>)
- **TRAIN** Toolbox for Reducing Atmospheric InSAR Noise
(<https://github.com/dbekaert/TRAIN>)

Methodology part 1 of the processing

Image downloading

The images are downloaded from ASF data search. First of all, you need to create an account to be able to download the data. Secondly you need to zoom and select the area of interest, in our case; New Zealand. In the additional filters you can select the start date and the end date and also the file type (L1 single look complex). Next, we will choose descending and we will then select the footprint that will be beneficial for our processing.

The information that is important of the product is the Path: 146 (relative orbit) and Frame: 731. When we know the frame that we desire we click on the right part of the path/frame on the wheel and we press “set as both” and go to “Download” and select “Data Download” to download the Python Script (.py). Then we put it inside the folder Original. Run the code to start the download in the same folder where the file is. At this stage you have the images necessary for your process. It is mandatory for the next steps to have python 2.7 installed.

```
C:\InSARprocessing\Original\Original>python download-all-2023-04-11_17-55-41.py
No existing URS cookie found, please enter Earthdata username & password:
(Credentials will not be stored, saved or logged anywhere)
> Cookiejar is bunk: None
Username: egarcibol4
Password (will not be displayed):
```

Figure 1: Run the download script.



Figure 2: Area of interest yellow star, swath and burst situation in the image

29 images from August 2022 until July 2023
DESC
Path: 146 (relative orbit)
Frame: 731

Swath: 1
4 bursts: 6-9

Installing the architecture and Preprocessing Snap2StaMPS:

Download in the <https://github.com/mdelgadoblasco/snap2stamps> the code in a .zip file. Extract the file and copy the folders “bin and graphs” Figure 3 of the file and put it in a folder called “Project”. Now you are ready to start the process. The folder “bin” contains all the python scripts ready to configure. The folder “graphs” have all the graphs created in SNAP that will be run by the python scripts.

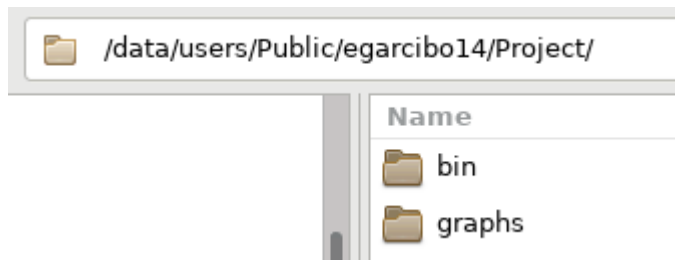


Figure 3: Copied folders

There are incompatibilities in the files, which is why some parameters from two xml files will need to be altered. These can be found in the “graphs” folder - “coreg_ifg_computation.xml” and “coreg_ifg_computation_subset.xml”. For the file “coreg_ifg_computation_subset.xml” you have to change the line 49 of the code as it is shown in the Figure 4 for the data in the Figure 5 . The same procedure is done in the file “coreg_ifg_computation.xml”. The parameters need to be updated.

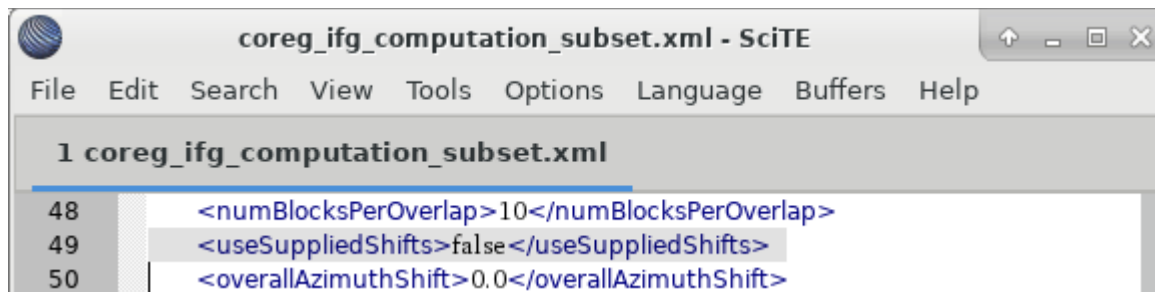


Figure 4: Data to **delete** in gray.

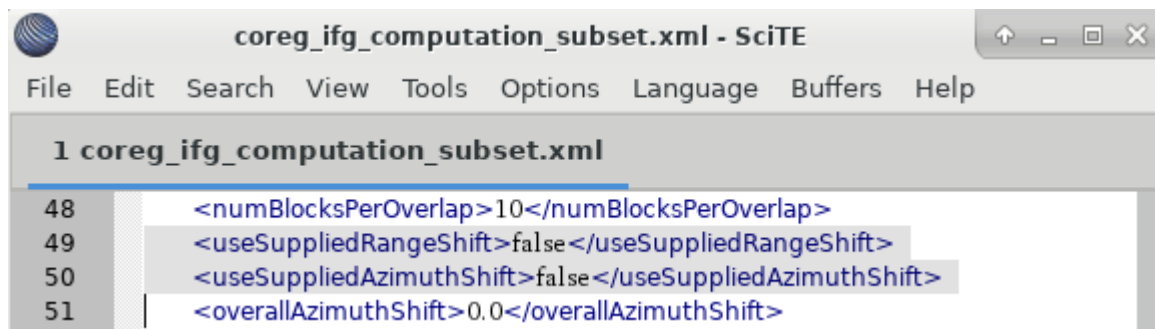


Figure 5: Data to **add** in gray.

Image burst and swath selection:

The area of interest is positions in one burst and swath. It is necessary to see which burst it is going to process. This case is the second swath (depending this first processing we will have to process the third swath) and the second and third burst that have the AOI.

The next step is to select the master image and create the folders Master and Slave.

The Master is selected through SNAP program Figure 6. For this, we will go to “Radar=> Interferometric=> InSAR Stack Overview => Add open” this will load all the images. By clicking “overview” the images will then be scanned and the process will select the best image to be the master.

The Slaves images from the folder “Original” are copied in the folder “Project “in a new folder “slaves”(it has to be named like this) .

File Name	Type	Acquisition	Track	Orbit
S1A_IW_SLC__1SDV_20220810T173928...	SLC	10Aug2022	146	44493
S1A_IW_SLC__1SDV_20220822T173929...	SLC	22Aug2022	146	44668
S1A_IW_SLC__1SDV_20220903T173930...	SLC	03Sep2022	146	44843
S1A_IW_SLC__1SDV_20220915T173930...	SLC	15Sep2022	146	45018
S1A_IW_SLC__1SDV_20221021T173930...	SLC	21Oct2022	146	45543
S1A_IW_SLC__1SDV_20221009T173930...	SLC	09Oct2022	146	45368
S1A_IW_SLC__1SDV_20220927T173930...	SLC	27Sep2022	146	45193
S1A_IW_SLC__1SDV_20221102T173930...	SLC	02Nov2022	146	45718
S1A_IW_SLC__1SDV_20221208T173929...	SLC	08Dec2022	146	46243
S1A_IW_SLC__1SDV_20221126T173930...	SLC	26Nov2022	146	46068
S1A_IW_SLC__1SDV_20221114T173930...	SLC	14Nov2022	146	45893
S1A_IW_SLC__1SDV_20221220T173928...	SLC	20Dec2022	146	46418
S1A_IW_SLC__1SDV_20230101T173928...	SLC	01Jan2023	146	46593

File Name	Ref/Sec	Acquisition	Track	Orbit	Bperp [m]	Btemp [days]	Modeled Co...	Height Amb...	Delta fDC [...]
S1A_IW_SLC__1S...	Reference	01Jan2023	146	46593	0.00	0.00	1.00	m	0.00
S1A_IW_SLC__1S...	Secondary	10Aug2022	146	44493	1.31	144.00	0.87	-11950.88	-0.44
S1A_IW_SLC__1S...	Secondary	22Aug2022	146	44668	34.95	132.00	0.85	-446.53	-1.55
S1A_IW_SLC__1S...	Secondary	03Sep2022	146	44843	240.89	120.00	0.71	-64.78	0.05
S1A_IW_SLC__1S...	Secondary	15Sep2022	146	45018	90.52	108.00	0.83	-172.39	-1.18
S1A_IW_SLC__1S...	Secondary	21Oct2022	146	45543	-56.62	72.00	0.89	275.59	0.85
S1A_IW_SLC__1S...	Secondary	09Oct2022	146	45368	-79.88	84.00	0.86	195.36	1.82
S1A_IW_SLC__1S...	Secondary	27Sep2022	146	45193	-33.02	96.00	0.89	472.57	-0.98
S1A_IW_SLC__1S...	Secondary	02Nov2022	146	45718	-79.12	60.00	0.88	197.24	-2.87
S1A_IW_SLC__1S...	Secondary	08Dec2022	146	46243	-29.81	24.00	0.95	523.42	-3.40
S1A_IW_SLC__1S...	Secondary	26Nov2022	146	46068	84.98	36.00	0.90	-183.63	1.73
S1A_IW_SLC__1S...	Secondary	14Nov2022	146	45893	15.05	48.00	0.94	-1037.18	0.00
S1A_IW_SLC__1S...	Secondary	20Dec2022	146	46418	-131.74	12.00	0.88	118.46	-4.02

Figure 6: InSAR master selection.

Update the orbit metadata

The second step is to Apply-Orbit-File and TOPSAR-Split Figure 7. These two steps are completed in the master file, in our case, the master is **20230101**. A new folder in “Project/master” is created to update the orbit metadata in the sentinel 1 file in order to choose the sentinel precise orbit. The TOPSAR needs to be applied to any interferometric processing, most of the operators need to work with one swath at a time. Here we select the Swath to process and the Bursts to process.

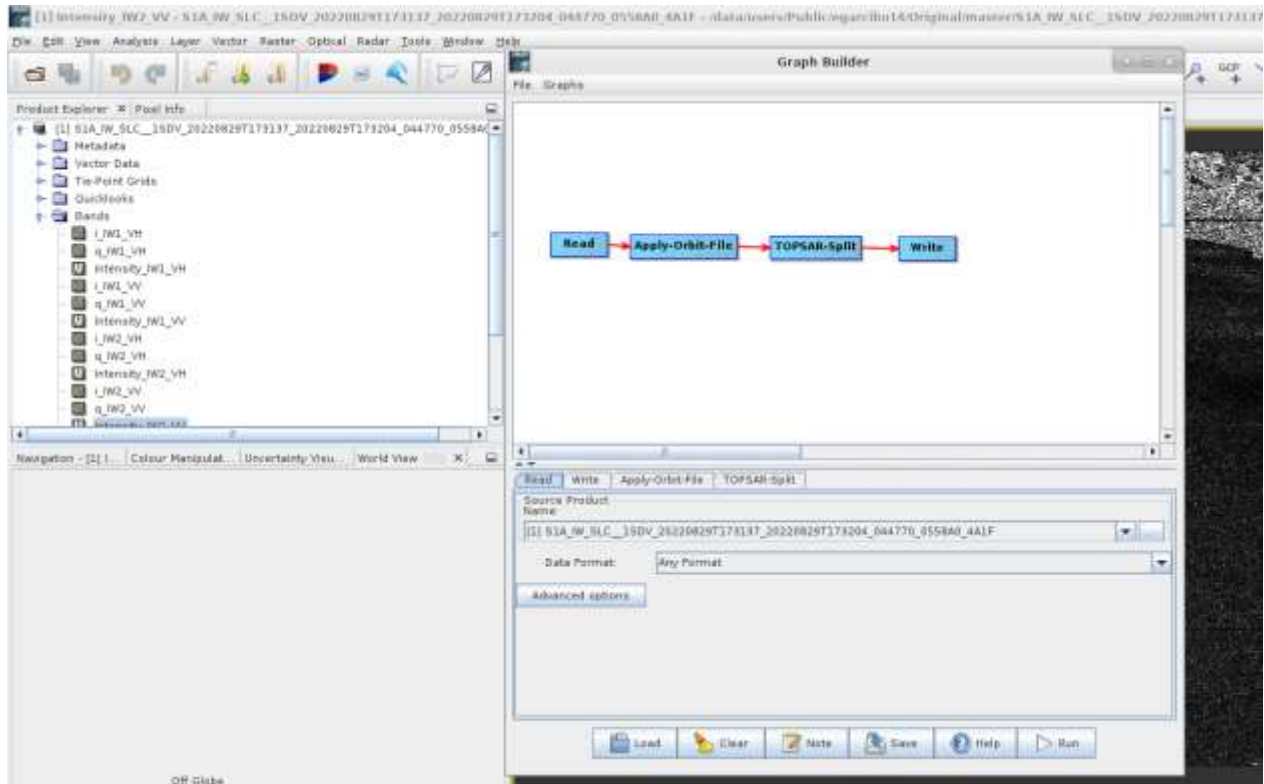


Figure 7: Update the orbit metadata.

The next step is to configure the SNAP2STAMS processing. The “bin” folder contains all the automated scripts for executing the SNAP2STAMS. The configuration file that we will be using is located in “bin” and the file name is, “project.conf” this file has several parameters that make up the information, Figure 8. In this config file is important to locate the master image and the path of the program of SNAP. Also, is important to put the capacity of the machine and you can check it in the terminal with the following =>lscpu.

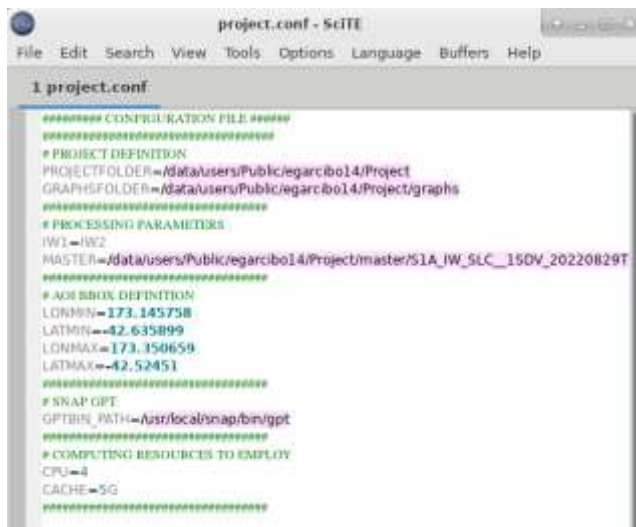


Figure 8: Configuration file for SNAP2STAMS.

It is necessary to put in order the slave's images in folder names with the date of the acquisition like in the image Figure 9 for and automatic processing.

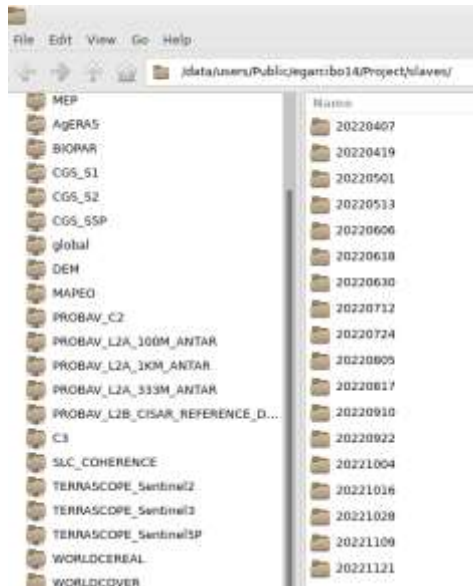


Figure 9: Slaves images in folders with the date of acquisition.

The script "slaves_prep.py" can be run with Python 2.7.5. With the following command Figure 10:

```
t/bin$ python2 slaves_prep.py project.conf
```

Figure 10: Command for preparation of slaves.

In order to run the scripts after the configuration file is important to have installed Python 2.7.5.

Apply the orbits

The next step is to apply the orbits calculated previously to all the slave images, the graph to apply is "Project/graphs/Slave_split_applyorbit.xml" with the following command we apply the orbits with a python script:

```
[egarcibo14@egarcibo14 bin]$ python2 splitting_slaves.py project.conf
```

Figure 11: Command for splitting the slaves.

The previous step is not time effective, for every image takes roughly one minute to process. Once the process is done in the "Project" folder will appear another folder named "split" that will contain the dim format with the sub swath product and the orbits applied.

Master slave co-registration and the interferogram generation

The next step is the master slave co-registration and the interferogram generation. In to the "bin" folder we will use the "correg_ifg_topsar.py" file to calculate the co-registration and the interferograms. The Graph that will be executed will be "coreg_ifg_computation_subset.xml" because we have indicated a

subset in our configuration file. If we do not setup any spatial setup it will execute “coreg_ifg_computation.xml”. If the graph is opened in SNAP is seen as the image below Figure 12.

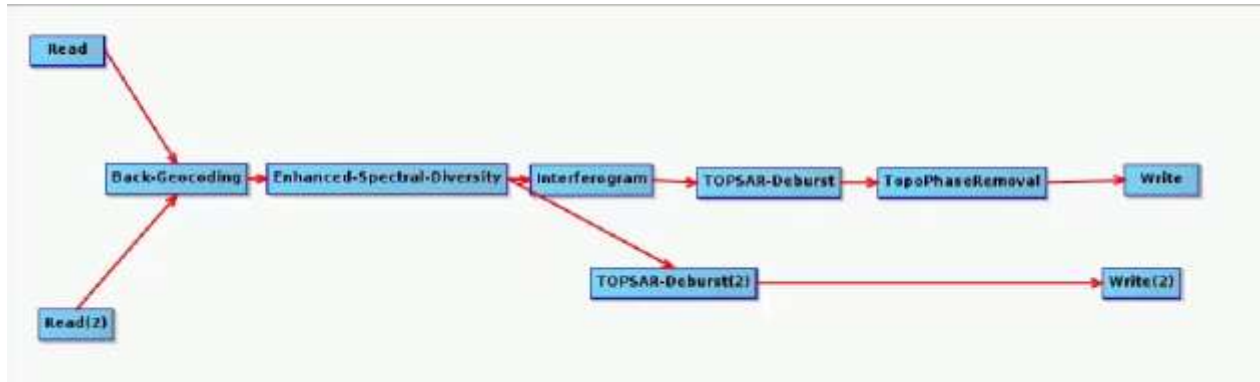


Figure 12: Graph opened in SNAP.

The graph “coreg_ifg_computation.xml” have several operations explained below:

The **Back-Geocoding** is connected to a master and to the slaves, this operation co-registration two SLC products, one master and one slave of the sub swaths with the same orbits and a digital elevation model.

The **Enhanced-Spectral-Diversity** operator, estimates the final constant range offset for the all sub swath.

The **Interferogram** which computes the complex interferogram by subtraction of the flat earth or reference phase it can also be run without. The Flat earth phase is the phase that is present in the interferometric signal due to the curvature of the reference surface Earth. Then we have the TOPSAR-Deburst (every sub-swath image consists of a series of bursts and every burst process as a separate SLC image and the individually focused complex burst images are included in the azimuth order in a single sub swath image with a black fill line in between) this processor merge the bursts in to a continuous image based on their zero Doppler time and removes the line.

The **TopoPhaseRemoval** this operator is a removal tool which estimates and subtracts the topographic phase from the interferogram.

Write operator writes the output.

At the end of the graph the co-registered images and the interferograms are exported.

```
C:\InSARprocessing\Project\Project\bin>python coreg_ifg_topsar.py project.conf
```

Figure 13:exported

After running this command two folders are created “ifg” the folder with the interferograms and the “coreg” co-registration folder.

We then need to check every interferogram that has been created to make sure there are no empty interferograms, otherwise it will create problems in the future processes, specifically with the amplitude dispersion and index estimation suitable for the PSI analysis. It is needed to open the “Phase ifg_srd_VV”

Figure 14 in Snap and see if the interferograms are populated and properly crated. We can observe the food print of the interferograms in Figure 15.

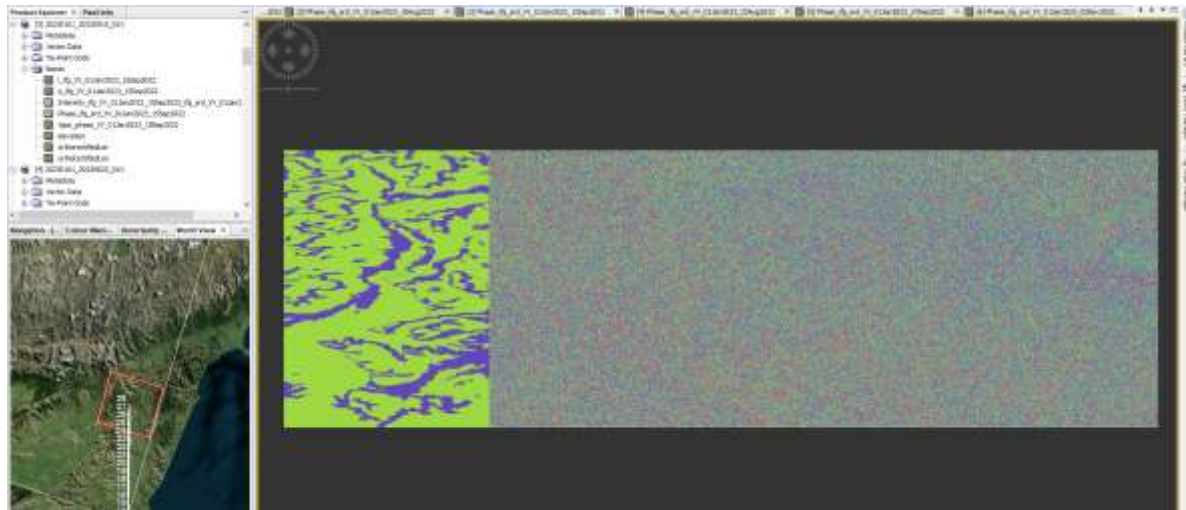


Figure 14: Snap visualization interferograms.

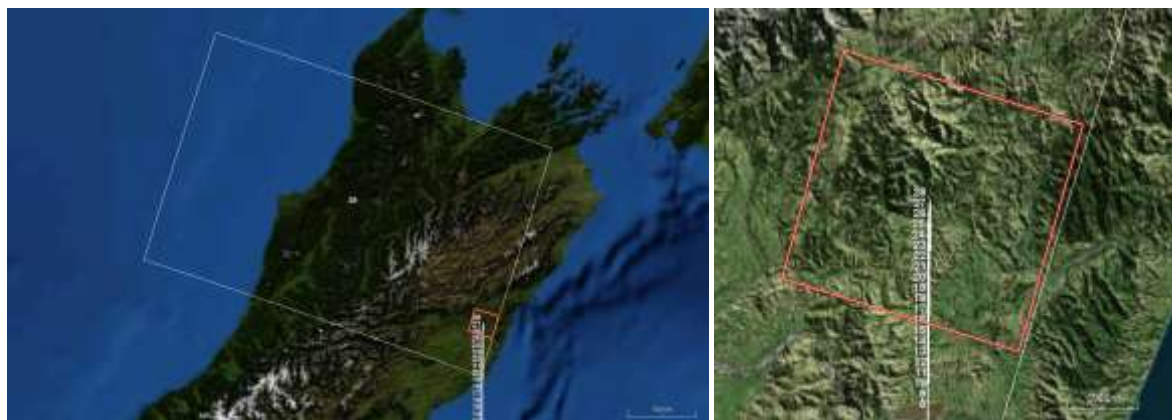


Figure 15: Interferogram foot print.

Once all the interferograms are checked and all are correct, the data can then be exported to introduce it to StaMPS with the following command Figure 16.

```
C:\InSARprocessing\Project\Project\bin>python stamps_export.py project.conf
```

Figure 16: Export the data to introduce to StaMPS

If the process is run properly we will have a folder named "INSAR_20230101" that look as the Figure 17.



Figure 17: Folder ready to start the process to StaMPS.

Methodology part2 of the processing

For the second part of the processing to obtain the velocity and accumulated deformation it is done with the software StaMPS. This software needs to be run in a Linux base machine and install the dependencies and the software to be able to run StaMPS. In this case we will use Terrascope Virtual Machine (VM).

The Virtual Machines of Terrascope are Linux-based virtual desktop environments with:

Free analysis tools: QGIS and GRASS GIS, SNAP desktop and GDAL utilities.

Code development environment for programming in Python, R or Java (Eclipse IDE).

Access to Global Land Service products and ancillary datasets on mounted disk volumes.

Free-of-charge standard configuration with 4 CPU, 8GiB RAM, 4GiB swap space, 80GiB root disk and Private and Public workspace folders of 100GiB each, with the option to extend at a price.

(<https://land.copernicus.eu/global/faq/what-terrascope-virtual-machine>)

In the following lines you will find the installation process and the second part of the processing with StaMPS.

Dependencies and software to install:

First of all before any installation we will go to the home folder and we will use the command “sudo yum update” to update all the installed packages to the latest version Figure 18. After the update we will proceed with the rest of installations.

```
[egarcibol14@egarcibol14 ~]$ sudo yum update
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
--> Package Rlib-DBI.noarch 0:1.0.0-0 will be updated
--> Package Rlib-DBI.x86_64 0:1.1.2-2 will be an update
--> Package Rlib-R.methodsS3.noarch 0:1.7.1-0 will be updated
--> Package Rlib-R.methodsS3.x86_64 0:1.8.1-2 will be an update
```

Figure 18: Update of installed packages.

Configure snappy SNAP - Python interface

The snappy is a python module, it can be installed as the follows from the “bin” folder of your python as the Figure 19.

```
[egarcibol14@egarcibol14 bin]$ ./snappy-conf /usr/bin/python2
```

Figure 19: Snappy configuration.

If it works should appear this Figure 20:

```
[egarcibol4@egarcibol4 bin]$ ./snappy-conf /usr/bin/python2
Configuring SNAP-Python interface...
Done. The SNAP-Python interface is located in '/home/egarcibol4/.snap/snap-python/snappy'
When using SNAP from Python, either do: sys.path.append('/home/egarcibol4/.snap/snap-python')
or copy the 'snappy' module into your Python's 'site-packages' directory.
```

Figure 20: Snappy proper configuration.

After it needs to copy the snappy folder to the site-packages folder of your python installation, the problem with Terrascope VM is that you cannot Figure 21. This is not a big problem because it do not give problems afterwards.

```
[egarcibol4@egarcibol4 bin]$ sudo cp -r /home/egarcibol4/.snap/snap-python/snappy /usr/lib/python2.7/site-packages
cp: error reading '/proc/sys/fs/binfmt_misc/register': Invalid argument
cp: failed to extend '/usr/lib/python2.7/site-packages/proc/sys/fs/binfmt_misc/register': Invalid argument
cp: cannot open '/proc/sys/net/ipv4/route/flush' for reading: Permission denied
cp: error reading '/proc/sys/net/ipv6/conf/all/stable_secret': Input/output error
cp: failed to extend '/usr/lib/python2.7/site-packages/proc/sys/net/ipv6/conf/all/stable_secret': Input/output error
cp: error reading '/proc/sys/net/ipv6/conf/default/stable_secret': Input/output error
cp: failed to extend '/usr/lib/python2.7/site-packages/proc/sys/net/ipv6/conf/default/stable_secret': Input/output error
cp: error reading '/proc/sys/net/ipv6/conf/docker0/stable_secret': Input/output error
cp: failed to extend '/usr/lib/python2.7/site-packages/proc/sys/net/ipv6/conf/docker0/stable_secret': Input/output error
cp: error reading '/proc/sys/net/ipv6/conf/eth0/stable_secret': Input/output error
cp: failed to extend '/usr/lib/python2.7/site-packages/proc/sys/net/ipv6/conf/eth0/stable_secret': Input/output error
cp: error reading '/proc/sys/net/ipv6/conf/lo/stable_secret': Input/output error
```

Figure 21: Snappy copy to the site-packages of python.

Website to check instructions from snappy:

<https://senbox.atlassian.net/wiki/spaces/SNAP/pages/50855941/Configure+Python+to+use+the+SNAP-Python+snappy+interface+SNAP+versions+9>

Sentinelsat

This is a python module which allows to enter esa's Copernicus Open Access Hub to download remote sensing data. The code that is used to install that module is the following Figure 22:

```
[egarcibol4@egarcibol4 ~]$ sudo pip install sentinelsat
```

Figure 22: Sentinelsat installation.

Website to check instructions of sentinelsat:

<https://sentinelsat.readthedocs.io/en/stable/index.html>

Pygeoj

This is a python module to read, write and work with the GeoJSON format. The following code is run to install the module Figure 23.

```
[egarcibol4@egarcibol4 ~]$ sudo pip install pygeoj
```

Figure 23: Pygeoj inatallation.

Website to check instructions of Pygeoj:

<https://pythonhosted.org/PyGeoJ/>

Triangle

The Triangle is a software generating exact Delaunay triangulations, constrained Delaunay triangulations, conforming Delaunay triangulations, Voronoi diagrams, and high-quality triangular meshes. It is used in the StaMPS workflow in step 4. We will download triangle software form this website <http://www.cs.cmu.edu/~quake/triangle.html>. The file downloaded is “triangle.zip”. After the download we will decompress the file triangle.zip in a folder called “triangle”. In order to install the software we do write click and Open Terminal Here will open a terminal in the same folder Figure 24.

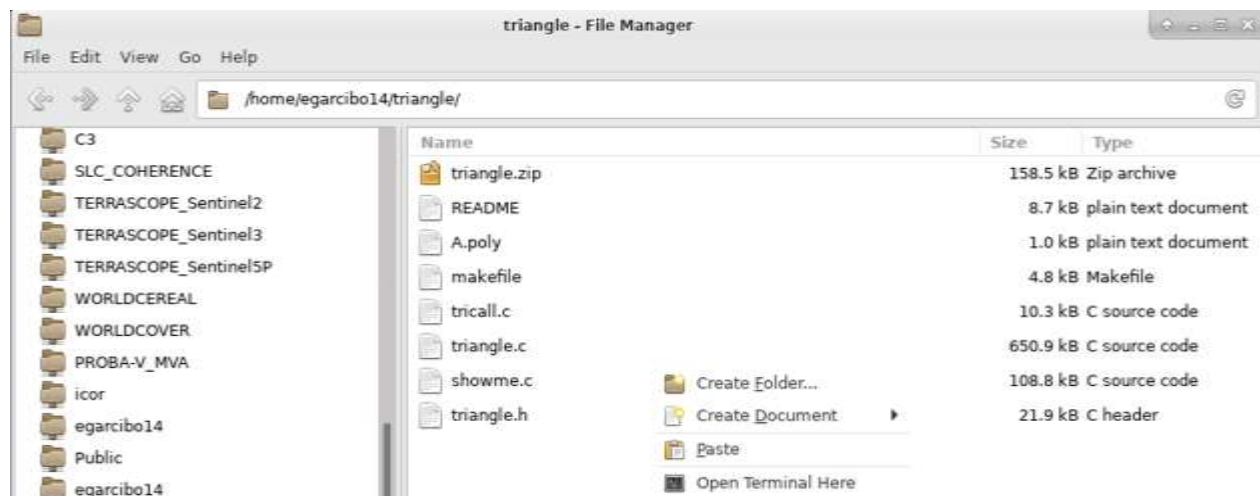


Figure 24: Triangle folder, opening of a terminal, write click Open Terminal Here.

Inside the terminal we will write the installation code “make” Figure 25 and click enter to execute. During the installation it is created two executable files Figure 26. To check if the Triangle software is properly installed we can execute the code “./triangle” and will show its properties Figure 25.

```

Terminal - egarcibo14@egarcibo14:~/triangle
File Edit View Terminal Tabs Help
[egarcibo14@egarcibo14 triangle]$ make
cc -O -DLINUX -I/usr/X11R6/include -L/usr/X11R6/lib -o ./triangle ./triangle.c -lm
cc -O -DLINUX -I/usr/X11R6/include -L/usr/X11R6/lib -o ./showme ./showme.c -lX11
[egarcibo14@egarcibo14 triangle]$ triangle
bash: triangle: command not found
[egarcibo14@egarcibo14 triangle]$ ./triangle
triangle [-prq_a_uAcDjevngBPNEIOXzo_YS_iFlsCQVh] input file
-p Triangulates a Planar Straight Line Graph (.poly file).
-r Refines a previously generated mesh.
-q Quality mesh generation. A minimum angle may be specified.
-a Applies a maximum triangle area constraint.
-u Applies a user-defined triangle constraint.
-A Applies attributes to identify triangles in certain regions.
-c Encloses the convex hull with segments.
-D Conforming Delaunay: all triangles are truly Delaunay.
-j Jettison unused vertices from output .node file.
-e Generates an edge list.
-v Generates a Voronoi diagram.
-n Generates a list of triangle neighbors.
-g Generates an .off file for Geomview.

```

Figure 25: Triangle installation code and comprovation.

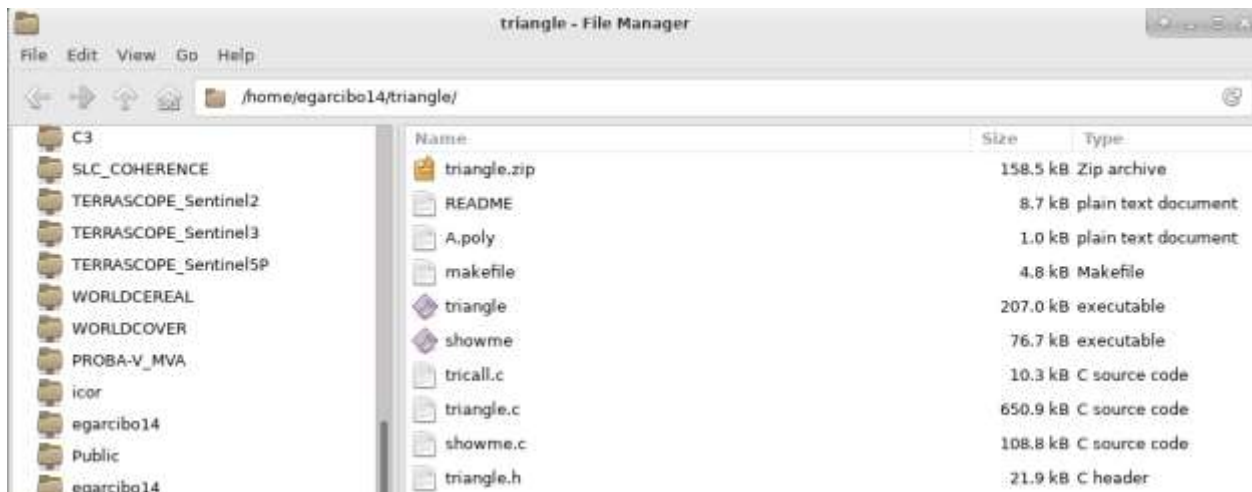


Figure 26: Triangle executable files.

Snaphu

The Snaphu is a software for two dimensional phase unwrapping. This software is used in the StaMPS processing in step 6 phase unwrapping. Have a look at this page for a full description.

The installation of the software should be done snaphu software can not be installed as it is done in ubuntu so to be able to install snaphu in Linux as it is in Terrascope VM it is necessary to download the program file and install it yourself manually.

The file can be downloaded in the following link:
<https://web.stanford.edu/group/radar/softwareandlinks/sw/snaphu/>

This link contain a file (snaphu-v2.0.5.tar.gz) that should be saved in the home folder of the Virtual machine and decompressed Figure 27.

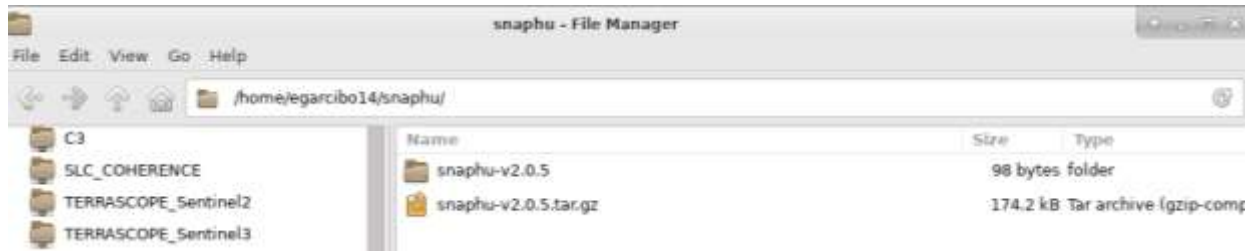


Figure 27: Snaphu software file decompression.

Inside the folder "src" it is a file Makefile that allow the installation in Linux Figure 28.

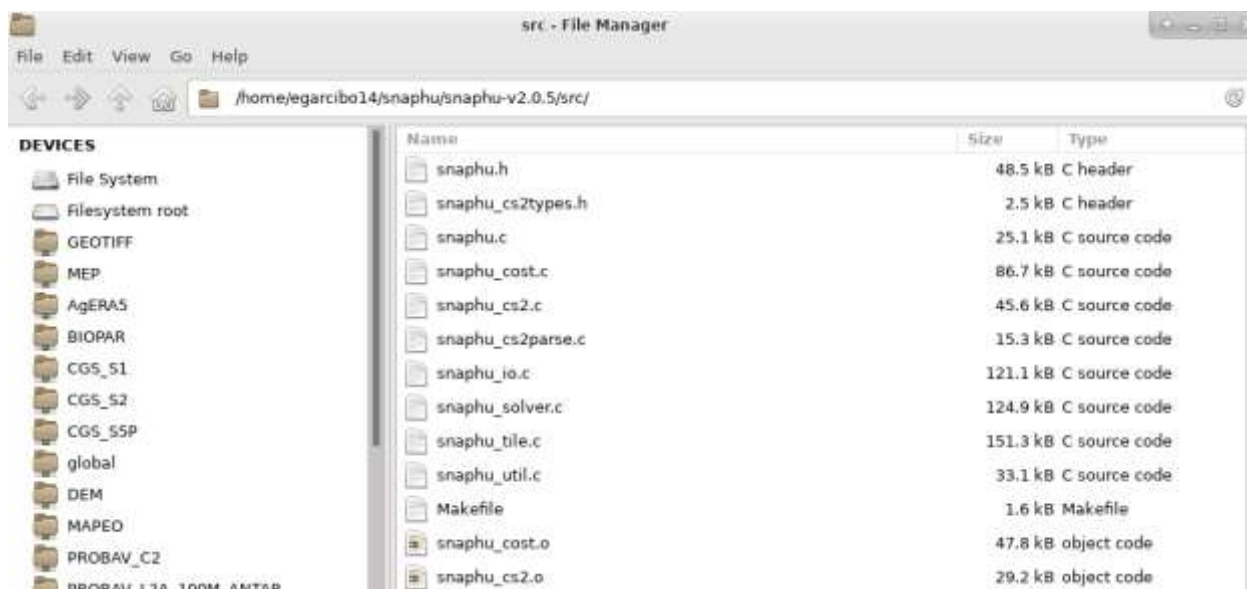


Figure 28: Snaphu folder src. Figure 28

You open the terminal and write the following code Figure 29 to install.

```
[egarcibo14@egarcibo14 src]$ makefile install
```

Figure 29: Snaphu installation.

The program will proceed to install itself.

The program should be properly installed to verify the functionality of the software you can run the following command "./snaphu" Figure 30 and the snaphu software will show its options:

```

Terminal - egarcibo14@egarcibo14:~/snaphu/snaphu-v2.0.5/bin
File Edit View Terminal Tabs Help
[egarcibo14@egarcibo14 bin]$ ./snaphu

snaphu v2.0.5
usage: snaphu [options] infile linelength [options]
most common options:
  -t          use topography mode costs (default)
  -d          use deformation mode costs
  -s          use smooth-solution mode costs
  -C <confstr> parse argument string as config line as from conf file
  -f <filename> read configuration parameters from file
  -o <filename> write output to file
  -a <filename> read amplitude data from file
  -c <filename> read correlation data from file
  -M <filename> read byte mask data from file
  -b <decimal> perpendicular baseline (meters)
  -i          do initialization and exit
  -S          single-tile reoptimization after multi-tile init
  -l <filename> log runtime parameters to file
  -u          infile is already unwrapped; initialization not needed
  -v          give verbose output
  --mst      use MST algorithm for initialization (default)
  --mcf      use MCF algorithm for initialization
  --tile <nrow> <ncol> <rowovrlp> <colovrlp> unwrap as nrow x ncol tiles
  --nproc <integer> number of processors used in tile mode

type snaphu -h for a complete list of options
[egarcibo14@egarcibo14 bin]$

```

Figure 30: Snaphu command of verification of functionality.

Another option is to download the plugin in SNAP. The Plugin can be found when you open SNAP and you go to Tools=> Plugins=> Available Plugins=> SNAPHU Unwrapping=> click "Install" **Figure 31**.

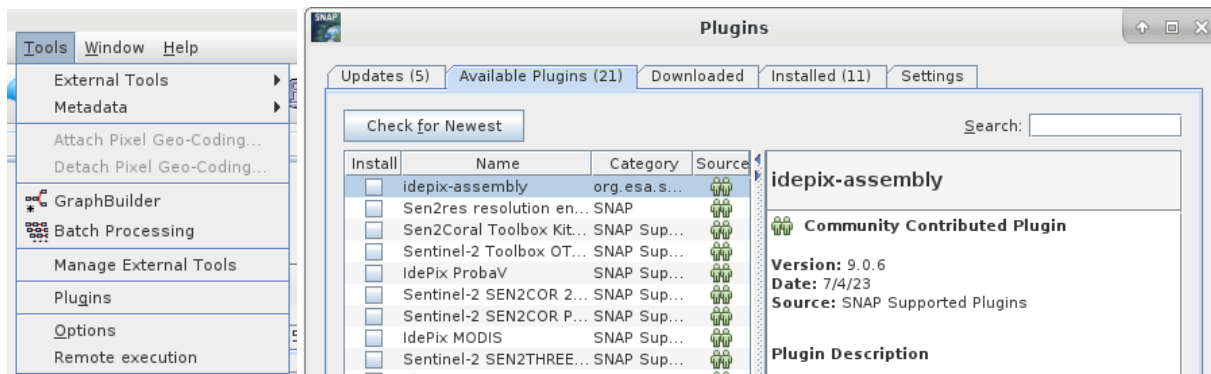


Figure 31: Installation SNAPHU Unwrapping plugin.

csh

Csh is an interpreter for C-shell which is needed to run scripts within the StaMPS installation.

To install it it is necessary to go in the folder of home and execute the command "sudo yum install csh" **Figure 32**.

```
Terminal - egarcibo14@egarcibo14:~
File Edit View Terminal Tabs Help
[egarcibo14@egarcibo14 ~]$ sudo yum install csh
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
--> Package tcsh.x86_64 0:6.18.01-15.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch             Version          Repository        Size
=====
Installing:
tcsh               x86_64           6.18.01-15.el7  os                338 k
=====

Transaction Summary
=====
Install 1 Package

Total download size: 338 k
Installed size: 662 k
Is this ok [y/d/N]: y
Downloading packages:
tcsh-6.18.01-15.el7.x86_64.rpm | 338 kB 00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : tcsh-6.18.01-15.el7.x86_64 1/1
  Verifying  : tcsh-6.18.01-15.el7.x86_64 1/1

Installed:
tcsh.x86_64 0:6.18.01-15.el7

Complete!
[egarcibo14@egarcibo14 ~]$
```

Figure 32: Csh installation.

Train

The Toolbox for Reducing Atmospheric InSAR Noise (TRAIN) can be used to estimate tropospheric delays (see [Bekaert et al., 2015a, b]). Train corrections are applied on the fly while plotting the results. This toolbox can be downloaded from tis website: <https://github.com/dbekaert/TRAIN> Ones TRAIN is downloaded it needs to be saved in a folder and decompress Figure 33.

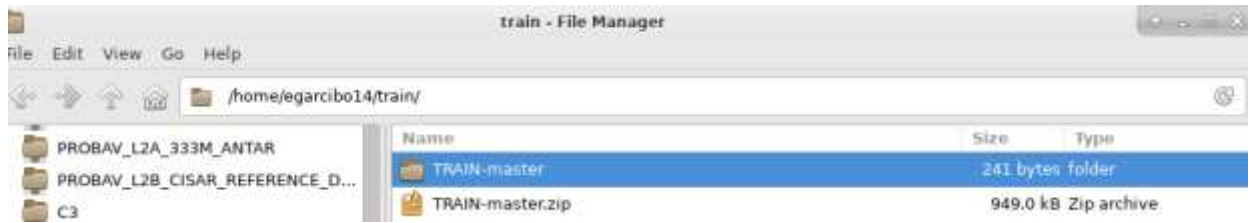


Figure 33: TRAIN decompression.

To be able to execute files and call the Toolbox it is necessary to configurate the APS_CONFIG.sh and modify the line 21 Figure 35, with the path of the folder where the toolbox is decompress Figure 34.

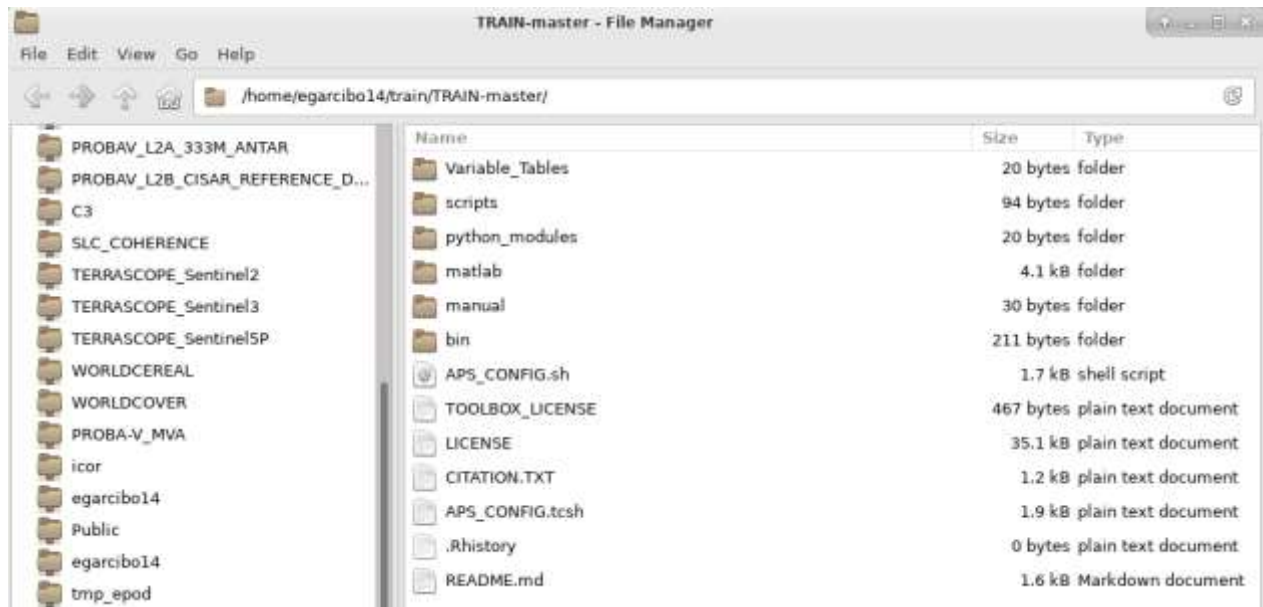


Figure 34: TRAIN decompress.

```

APS_CONFIG.sh - SciTE
File Edit Search View Tools Options Language Buffers Help

1 APS_CONFIG.sh
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
--# Copyright (C) 2015 Bekaert David - University of Leeds
--# Email: eadpsb@leeds.ac.uk or davidbekaert.com
--#
--# This program is free software; you can redistribute it and/or modify
--# it under the terms of the GNU General Public License as published by
--# the Free Software Foundation; either version 2 of the License, or
--# (at your option) any later version.
--#
--# This program is distributed in the hope that it will be useful,
--# but WITHOUT ANY WARRANTY, without even the implied warranty of
--# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
--# GNU General Public License for more details.
--#
--# You should have received a copy of the GNU General Public License along
--# with this program; if not, write to the Free Software Foundation, Inc.,
--# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
--#
--# Give the correct path to the APS toolbox
--export APS_toolbox="/nfs/see-fs-01_users/eadpsb/software/vva_aps"
export APS_toolbox="/home/egarcibo14/train/TRAIN-master/"
--# export PYTHONPATH="$PYTHONPATH:/nfs/see-fs-01_users/eadpsb/software/vva_aps/python"
--# full path to the get_modis.py file
--# export get_modis_filepath="/nfs/see-fs-01_users/eadpsb/software/python_packages/oscar-client-python/get_modis.py"
--#
--# shouldn't need to change below here
--#
case "$MATLABPATH:" in
  *) export MATLABPATH="$MATLABPATH:${APS_toolbox}/matlab";;
  *) export MATLABPATH="$MATLABPATH:${APS_toolbox}/matlab:${MATLABPATH}";;
esac

export APS_toolbox_scripts="${APS_toolbox}/scripts"
export APS_toolbox_bin="${APS_toolbox}/bin"
export PATH="$PATH:${APS_toolbox_bin}:${PYTHONPATH}"

```

Figure 35: TRAIN file to configure the path of the folder where the toolbox is decompress.

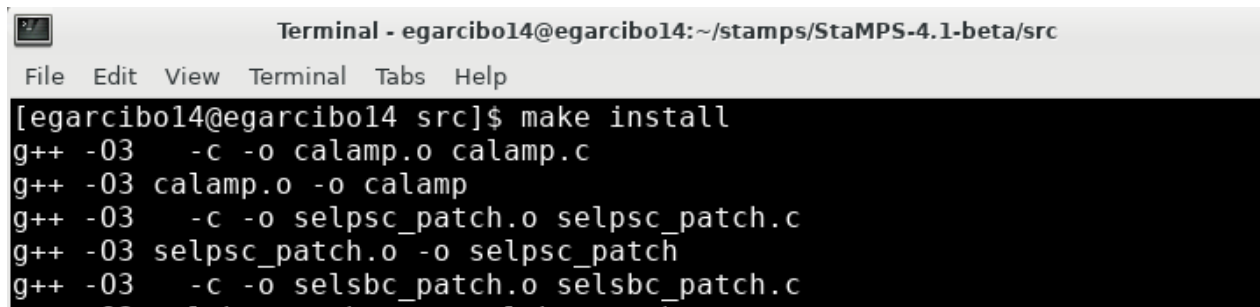
StaMPS

The next step is to install StaMPS from this website

<https://homepages.see.leeds.ac.uk/~earahoo/stamps/> the file is decompress and to install the program you should be on the "src" Figure 36 folder and run the code "make install" Figure 37



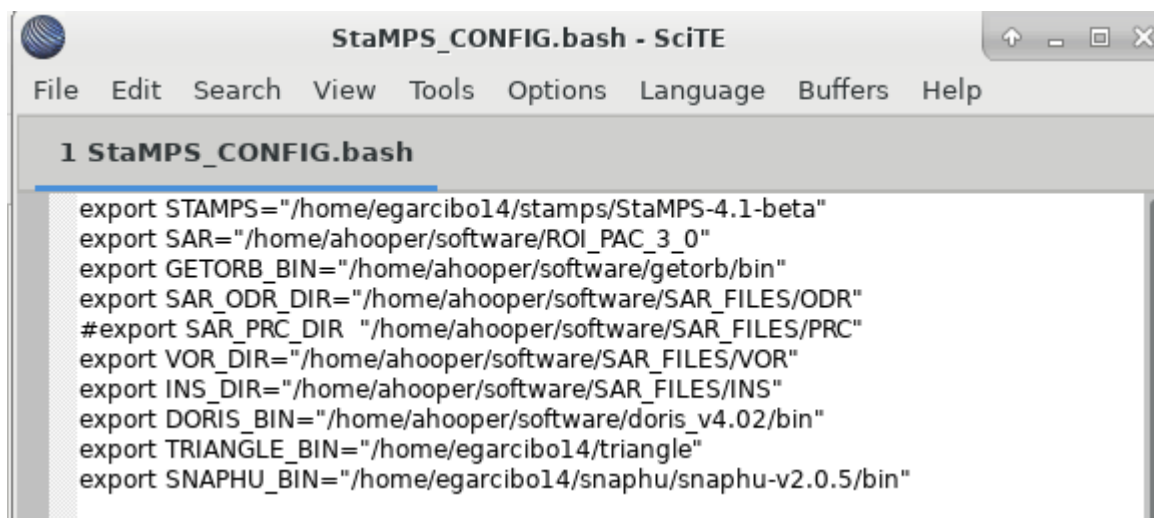
Figure 36: StaMPS src folder.



```
Terminal - egarcibo14@egarcibo14:~/stamps/StaMPS-4.1-beta/src
File Edit View Terminal Tabs Help
[egarcibo14@egarcibo14 src]$ make install
g++ -O3 -c -o calamp.o calamp.c
g++ -O3 calamp.o -o calamp
g++ -O3 -c -o selpsc_patch.o selpsc_patch.c
g++ -O3 selpsc_patch.o -o selpsc_patch
g++ -O3 -c -o selsbc_patch.o selsbc_patch.c
```

Figure 37: StaMPS installation code.

For the software stamps it is necessary to configure the first lines of the file “StaMPS_CONFIG.bash”. This first lines are the paths to the STAMPS, TRIANGLE_BIN and SNAPHU_BIN directories Figure 38.



```
StaMPS_CONFIG.bash - SciTE
File Edit Search View Tools Options Language Buffers Help
1 StaMPS_CONFIG.bash
export STAMPS="/home/egarcibo14/stamps/StaMPS-4.1-beta"
export SAR="/home/ahooper/software/ROI_PAC_3_0"
export GETORB_BIN="/home/ahooper/software/getorb/bin"
export SAR_ODR_DIR="/home/ahooper/software/SAR_FILES/ODR"
#export SAR_PRC_DIR "/home/ahooper/software/SAR_FILES/PRC"
export VOR_DIR="/home/ahooper/software/SAR_FILES/VOR"
export INS_DIR="/home/ahooper/software/SAR_FILES/INS"
export DORIS_BIN="/home/ahooper/software/doris_v4.02/bin"
export TRIANGLE_BIN="/home/egarcibo14/triangle"
export SNAPHU_BIN="/home/egarcibo14/snaphu/snaphu-v2.0.5/bin"
```

Figure 38: Configuration of StaMPS_CONFIG.bash.

Matlab installation

The Matlab needs to be installed is the only software not open source and free off the process. The following software I will not explain the installation as it depends the licensing you have.

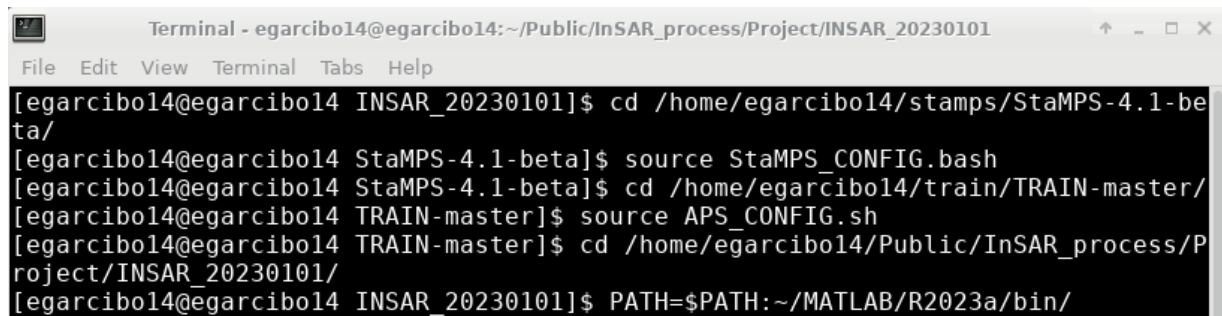
Matlab licensing with the toolboxes:

- Parallel computing Toolbox (DM)
- Image Processing Toolbox (IP)
- Signal Processing Toolbox (SG)
- Statistics and Machine Learning Toolbox (ST)

With this steps is done all the installation process.

Processing in StaMPS

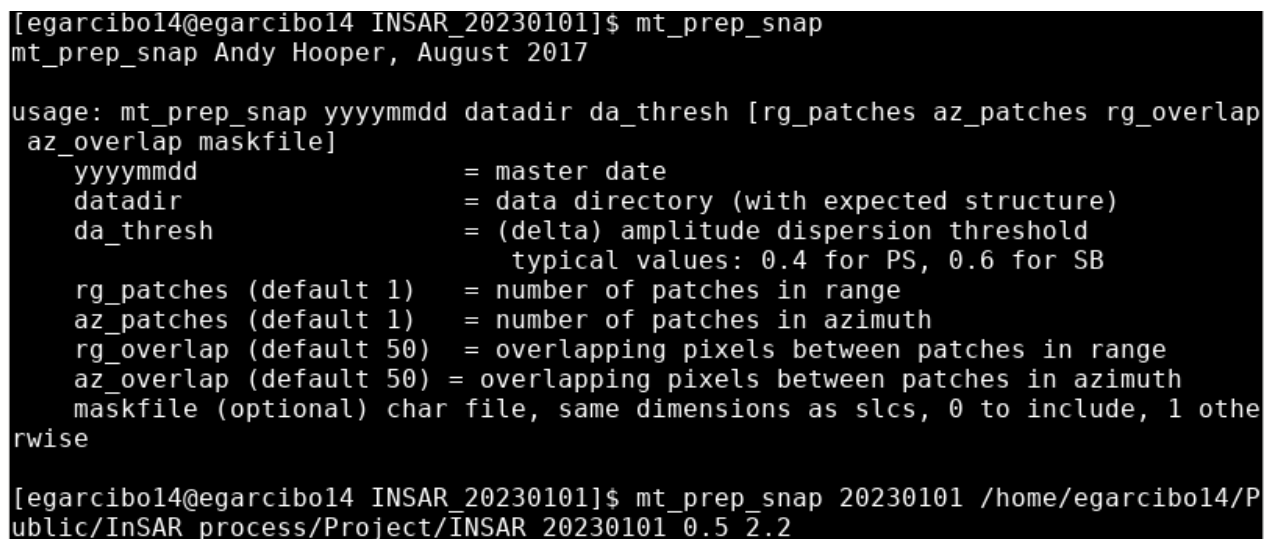
To be able to run StaMPS is needed to indicate some variables to the bash. The variables will be call inside the Stamps software. It is necessary open a terminal in the folder were the Interferograms are generated in this case is the folder is “INSAR_20230101”. In this terminal we link the variable to the bash to be able to call the functions that we want. The code is “source” and we call the files inside every path, “StaMPS_CONFIG.bash” and “APS_CONFIG.sh” to have the variables Figure 39. In the same terminal we will point the path to the Matlab software Figure 39.



```
Terminal - egarcibo14@egarcibo14:~/Public/InSAR_process/Project/INSAR_20230101
File Edit View Terminal Tabs Help
[egarcibo14@egarcibo14 INSAR_20230101]$ cd /home/egarcibo14/stamps/StaMPS-4.1-beta/
[egarcibo14@egarcibo14 StaMPS-4.1-beta]$ source StaMPS_CONFIG.bash
[egarcibo14@egarcibo14 StaMPS-4.1-beta]$ cd /home/egarcibo14/train/TRAIN-master/
[egarcibo14@egarcibo14 TRAIN-master]$ source APS_CONFIG.sh
[egarcibo14@egarcibo14 TRAIN-master]$ cd /home/egarcibo14/Public/InSAR_process/Project/INSAR_20230101/
[egarcibo14@egarcibo14 INSAR_20230101]$ PATH=$PATH:~/MATLAB/R2023a/bin/
```

Figure 39: Variable setting.

The first step to be able to ingest the data from Snap2StaMPS to StaMPS is to calculate the amplitude dispersion index. The amplitude dispersion index is a value that describes the amplitude stability that is used to preselect the pixels for the phase analysis. The recommended range for the pixel selection is from 0.4 to 0.42 the higher the threshold the more pixels will be selected for phase analysis. In this case we will select a 0.5 threshold due to the area that we have selected. The area is full of vegetation and it normally trend to have unstable amplitude values so we chose higher threshold, to include more points. To run this step first we provide the “mt_prep_snap” Figure 40 then the master date, then we provide the path to the “INSAR_20230101” folder, the 0.5 threshold and we chose also the patches in range and also the patches in azimuth in this case 2 2.



```
[egarcibo14@egarcibo14 INSAR_20230101]$ mt_prep_snap
mt_prep_snap Andy Hooper, August 2017

usage: mt_prep_snap yyyyymmdd datadir da_thresh [rg_patches az_patches rg_overlap
az_overlap maskfile]
  yyyyymmdd           = master date
  datadir              = data directory (with expected structure)
  da_thresh            = (delta) amplitude dispersion threshold
                      typical values: 0.4 for PS, 0.6 for SB
  rg_patches (default 1) = number of patches in range
  az_patches (default 1) = number of patches in azimuth
  rg_overlap (default 50) = overlapping pixels between patches in range
  az_overlap (default 50) = overlapping pixels between patches in azimuth
  maskfile (optional) char file, same dimensions as slcs, 0 to include, 1 otherwise

[egarcibo14@egarcibo14 INSAR_20230101]$ mt_prep_snap 20230101 /home/egarcibo14/Public/InSAR_process/Project/INSAR_20230101 0.5 2.2
```

Figure 40: Prepare patches and the parameters.

After the process you will see the 4 folders with the patches are created and also different files Figure 41.

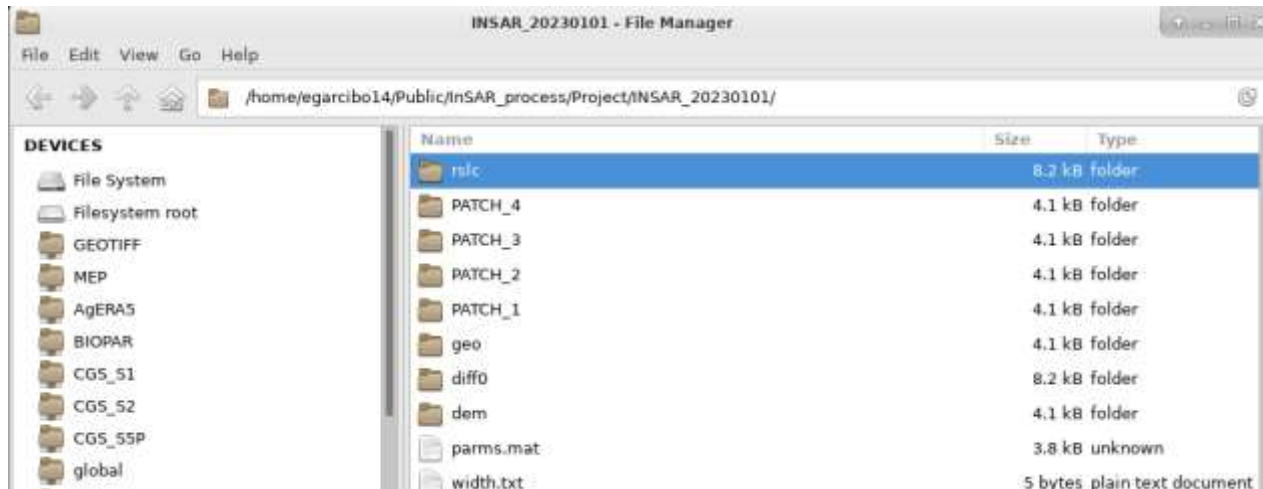


Figure 41: Patch creation.

The next step is to open Matlab at the same terminal we have processed the patches. We are going to write “matlab” and press enter Figure 42.

```
[egarcibo14@egarcibo14 INSAR_20230101]$ matlab
MATLAB is selecting SOFTWARE_OPENGL rendering.
```

Figure 42.MATLAB.

Once the Matlab is open we will check if the stamps is installed properly in to the system. We write the coman “help stamps” and if it show the same as the Figure 43 it is well installed.

```
>> help stamps
stamps Stanford Method for Persistent Scatterers
stamps(START_STEP,END_STEP,PATCHES_FLAG,EST_GAMMA_FLAG) Default is to run all steps.
A subset of steps may be selected with START_STEP and/or END_STEP
STEP 1 = Initial load of data
STEP 2 = Estimate gamma
STEP 3 = Select PS pixels
STEP 4 = Weed out adjacent pixels
STEP 5 = Correct wrapped phase for spatially-uncorrelated look angle error and merge patches
STEP 6 = Unwrap phase
STEP 7 = Calculate spatially correlated look angle (DEM) error
STEP 8 = Filter spatially correlated noise
STEP 0 = Continue from the last known stage till the end-stage selected

PATCHES_FLAG Default 'y'. Set to 'n' to process all data as one patch

EST_GAMMA_PARM is an optional parameter passed to PS_EST_GAMMA_QUICK

PATCH_LIST_FILE is an optional argument specifying the file list of
patches to be processed. Note that from step 5 and above one should use
all patches to merge results.

If current directory is a single patch, stamps only operates in the
current directory, but if current directory contains many patches,
stamps operates on them all.

Andy Hooper, June 2006
```

Figure 43: Stamps comprovation.

It is necessary that the path to the Atmospheric toolbox TRAIN is also recognizable in Matlab so we will write the command “addpath” followed by the path of TRAIN that contain Matlab as in the Figure 44.

```
>> addpath('/home/egarcibo14/train/TRAIN-master/matlab')
```

Figure 44: insertion of path TRAIN that contain Matlab.

Once the path is added we can confirm that it was successful writing “help aps_linear” Figure 45.

```
>> help aps_linear
```

Figure 45: Successful path linkage in Matlab.

To start with the Stamps processing you have to take in to account that to process the patches it is needed to do the steps 1-4 for every patch and ones they are finalized and you start the step 5 were all the patches are merged in this step. To start the process you can check all the parameters that you can change in stamps, for that you only need to type “getparm” Figure 46.

```
>> getparm
                Created: '27-Jul-2023'
                clap_alpha: 1
                clap_beta: 0.3000
    clap_low_pass_wavelength: 800
                clap_win: 32
                density_rand: 20
                drop_ifg_index: []
                filter_grid_size: 50
                filter_weighting: 'P-square'
    gamma_change_convergence: 0.0050
    gamma_max_iterations: 3
    gamma_stdev_reject: 0
                heading: 194.6092
    insar_processor: 'snap'
                lambda: 0.0555
    lonlat_offset: [0 0]
```

Figure 46: Parameters in Stamps.

And if you want to set any parameter it can be done just typing “setparm(name_of_the_parameter)” Figure 47.

```
>> setparm('plot_scatterer_size',30)
SETPARM: plot scatterer size = 30
```

Figure 47: Example of setting parameters.

Step1

The process we do now will be applied to the Patch 1, so it is run the first step just typing “stamps (1, 1)” Figure 48. This indicates that you want to start with the step 1 and you also want to finalize with step 1, if these is note indicated the software would run all the steps in one.

This step lodes the first persistent scatter candidates of the dataset.

```
>> cd PATCH_1
>> stamps(1, 1)
```

Figure 48: Step1.

If we want to know which files have been created we type “ls *1.mat” Figure 49 and will appear all the files created. Moreover is possible to know more information about the points if it is type “ps_info” which will give us the list of the interferograms that we have including the perpendicular baseline distance Figure 49.

```
>> ls *1.mat
bp1.mat dal.mat hgt1.mat la1.mat ph1.mat pml.mat psl.mat

>> ps_info
 1 10-Aug-2022  -77 m
 2 22-Aug-2022  -54 m
 3 03-Sep-2022  174 m
 4 15-Sep-2022   26 m
 5 27-Sep-2022 -105 m
 6 09-Oct-2022 -140 m
 7 21-Oct-2022 -109 m
 8 02-Nov-2022  -99 m
 9 14-Nov-2022   6 m
10 26-Nov-2022  82 m
11 08-Dec-2022  -42 m
12 20-Dec-2022 -145 m
13 01-Jan-2023   0 m
14 13-Jan-2023  -72 m
15 25-Jan-2023  72 m
16 06-Feb-2023  -95 m
17 18-Feb-2023 -112 m
18 02-Mar-2023  23 m
19 14-Mar-2023  -80 m
20 26-Mar-2023  23 m
21 07-Apr-2023  -80 m
22 19-Apr-2023 -291 m
23 01-May-2023  -3 m
24 13-May-2023  56 m
25 25-May-2023 -111 m
26 06-Jun-2023 -189 m
27 18-Jun-2023 -107 m
28 30-Jun-2023 -162 m
29 12-Jul-2023  12 m
Number of stable-phase pixels: 273568
```

Figure 49: Information of the files created and the interferograms.

Now we will plot the wrap phase using the code line “ps_plot ('w’)” Figure 50. In the plot we have the 29 interferograms and we can see that the master image has a phase difference of 0.

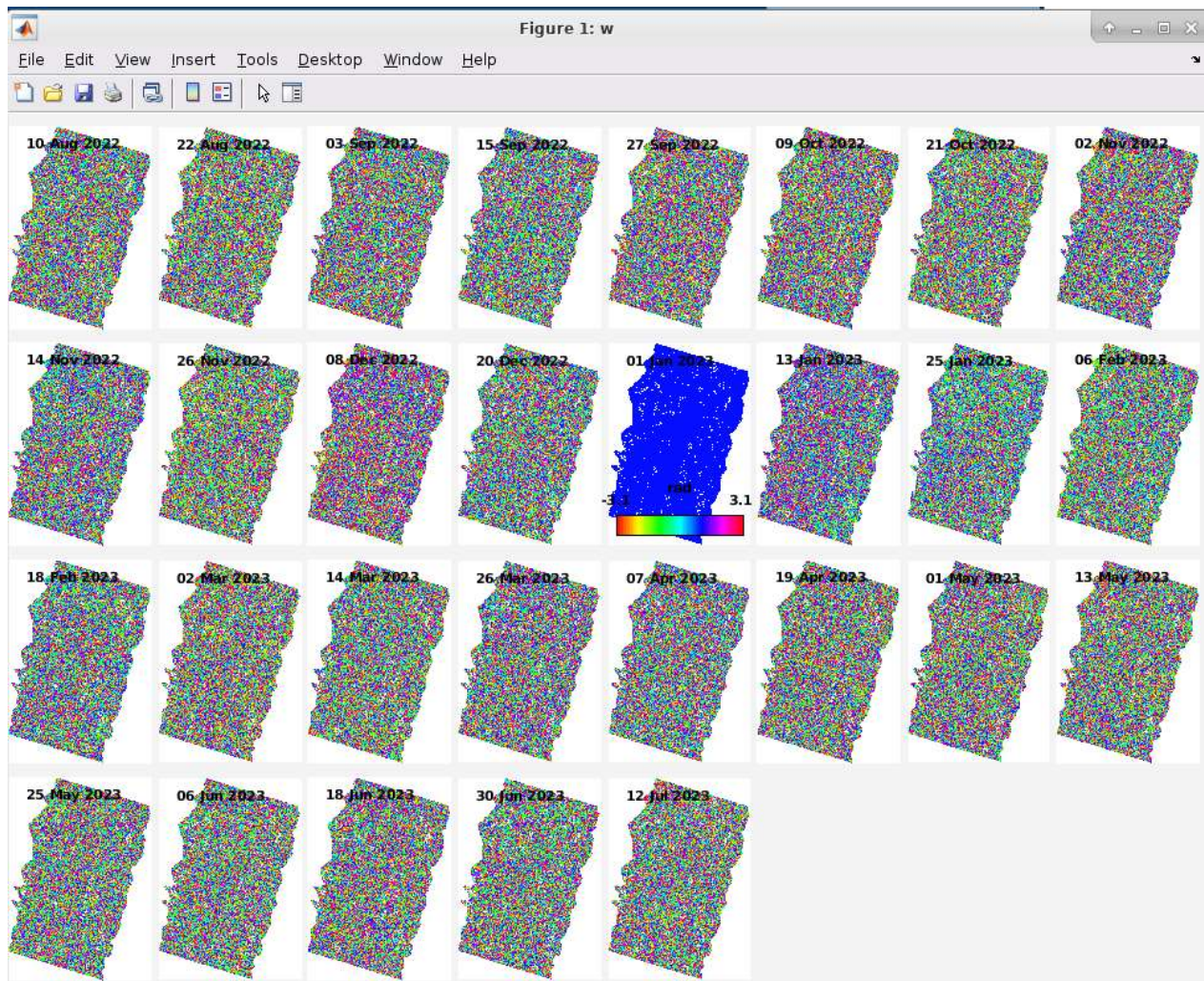


Figure 50: Wrap interferograms.

Also we can plot the mean amplitude of all the interferograms this will show how the interferograms are more or less coherent respect the darkness of the area of the interferogram.

In this case we can see that on the Figure 51 the master image of 20230101 it has darker areas where the amplitude is not consistent.

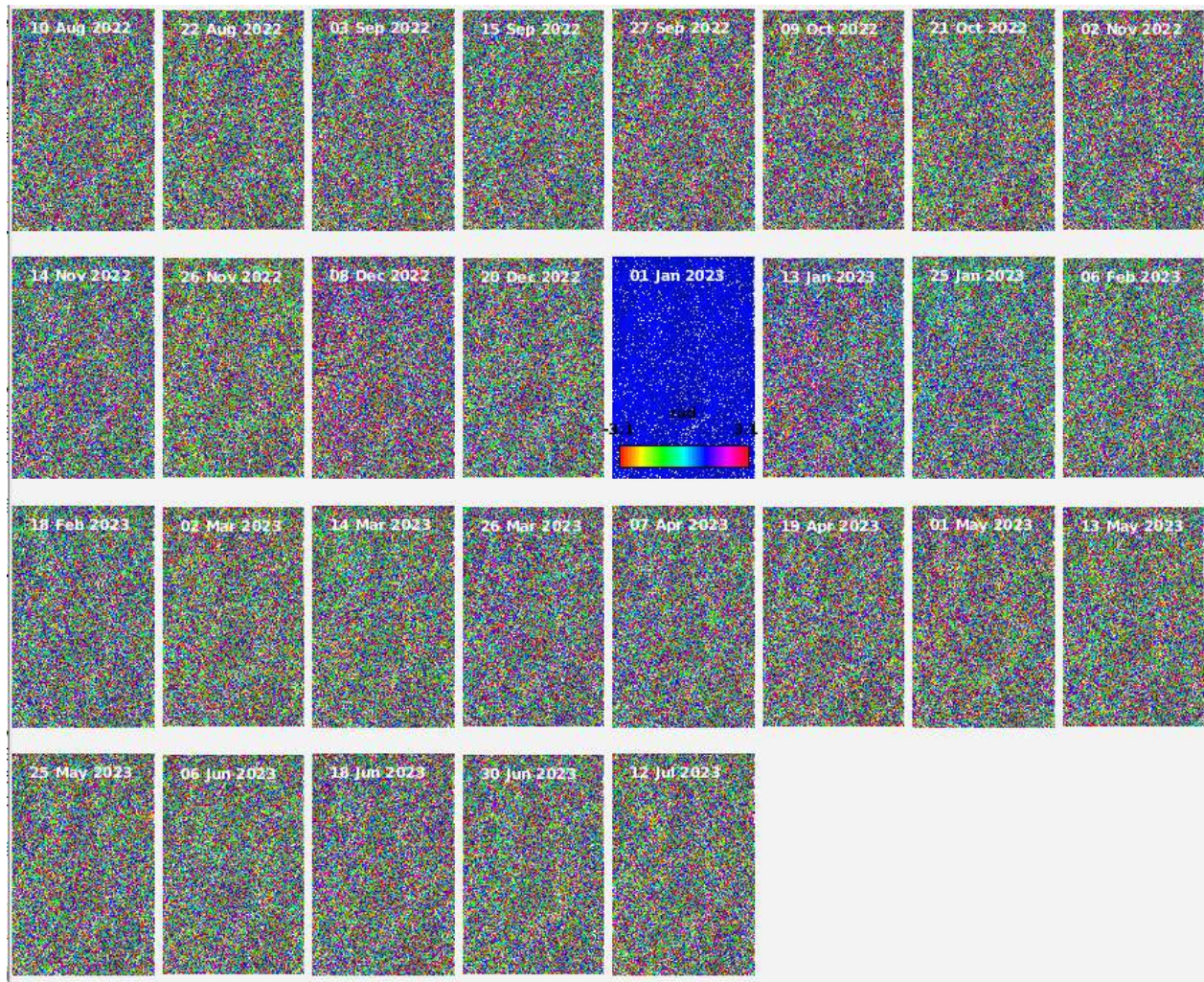


Figure 51: Mean amplitude of the interferograms.

Step2

Now we can move to the step 2 but before we have to take in account several parameters:

Maximum topographic error is the parameter “max_topo_err” correlation is the maximum uncorrelated digital elevation model error in meters, pixels that have the error larger than the value of the parameter will not be selected. If this value is increased it will also increase the gamma value and more pixels will be selected. The parameter that we will use is 20.

The parameter is “filter_weghting => ‘P-square’” it means that the persistent scatter probability squared, the other possibility is the signal to noise ratio. We are selecting the “P-square” because it perform better.

```
>> stamps (2,2)
```

Figure 52: Stamps Run step 2

The step 2 is run with all the parameters as default. One of the characteristics that we can see or control is the running iterations. The predefined are 3 iterations and the threshold that this iterations are

expecting to acquire are the “gamma_change_convergence: 0.005” so every iteration should be closer to this parameter. In the case of this processing it starts with 0.322898 and ends with -0.035137.

Step3

The step 3 is the initial persistent scatter selection, it is selected the PS points based on probability by comparing the results by data with random phase. This is done twice after the first selection the temporal coherence for each pixel is then reestimated more accurately by dropping the pixels it self from the estimation of the spatially correlated phase. The reestimation of the random phase and selection of pixels that are more likely to be PS. This is done by different selection methods one is by percentage or density, is the fraction that we accept of having false positives in a area of one square kilometer. In this processing we leave the parameters by default “Density” and the parameter is “density_rand:20” at these stage we can accept higher density as in the next step this is more restrictive. We write “>> stamps (3.3)”.

Step4

In this step the pixels that have been selected in the step number 3 are readed which means that we drop pixels that had been selected due to signal contribution from the neighboring pixels and also those PS points that are seamed as too noisy. This step is governed by a number of different parameters: “weed_standar_deviation” which is set to one by default, this is a threshold for standard deviation. basically for each pixel the face noise standard deviation for all pixel pairs including the pixel is calculated and if the minimum standard deviation is greater than the threshold then the pixel is dropped. The values recommended are set to 1 to 1.2.

“weed_max_noise” by default is set to Infinite. This is the threshold of the maximum noise allowed per pixel. For each pixel the minimum pixel pair noise is estimated per inerferogram and the pixels with maximum interferogram noise higher than this value are then dropped.

“weed_time_win” this is the time window in days, temporal reading.

“weed_neightbours” this enables the dropping of neighbours based on the proximity of each other. This parameter normally is wanted to set to yes so we will change the default Figure 53.

```
>> setparam('weed_neightbours', 'y')
SETPARM: weed_neightbours = y
```

Figure 53: Set parameters for “weed_neightbours”.

“weed_zero_elevation” is when your area of interest have sea on it and you expect that on the 0 elevation points you do not expect PS, because water should not have information.

After the change of parameters we run the process typing “>> stamps (4.4)”.

We have to run the same 4 process for all the patches.

Step 5

The step 5 is the phase correction, in this step the wrapped face of the selected pixels is corrected for the spatially uncorrelated look angle, which is the angle of digital elevation model error. This error is estimated in the step2. At the end of this step all the patches are merged. There are two main parameters that govern this step that are:

Resample size, that is to save memory, if this parameter is set it is appropriate to set the merge standard deviation parameter, for each pixel the phase noise standard deviation is computed and if the standard deviation is greater than the threshold than the resampled pixel is dropped.

The Step 5 needs to be run from the INSAR main directory, "INSAR_20230101". The step 5 is also **run by default**. We run the process typing ">> stamps (5.5)".

The results of this step is same on the main directory with the suffix 2 and contains all the information for the second selection of the PS points and is merged all of the study area. Now we can visualize is the wrapped phase for the entire area, and for all the interferograms. We run the command "ps_plot('w') and it plot the wrapped phase with the values of pi to minus pi, Figure 54, Figure 55.

```
>> ps_plot('w')
```

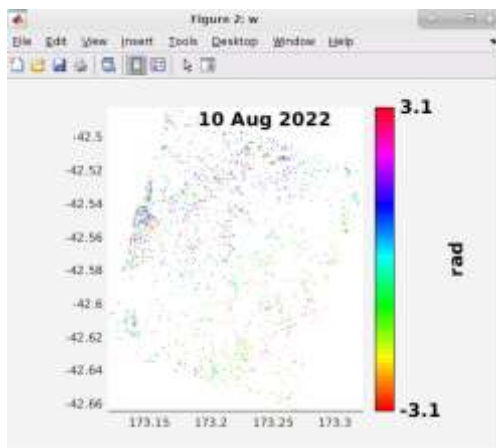


Figure 54: Wrapped interferogram.

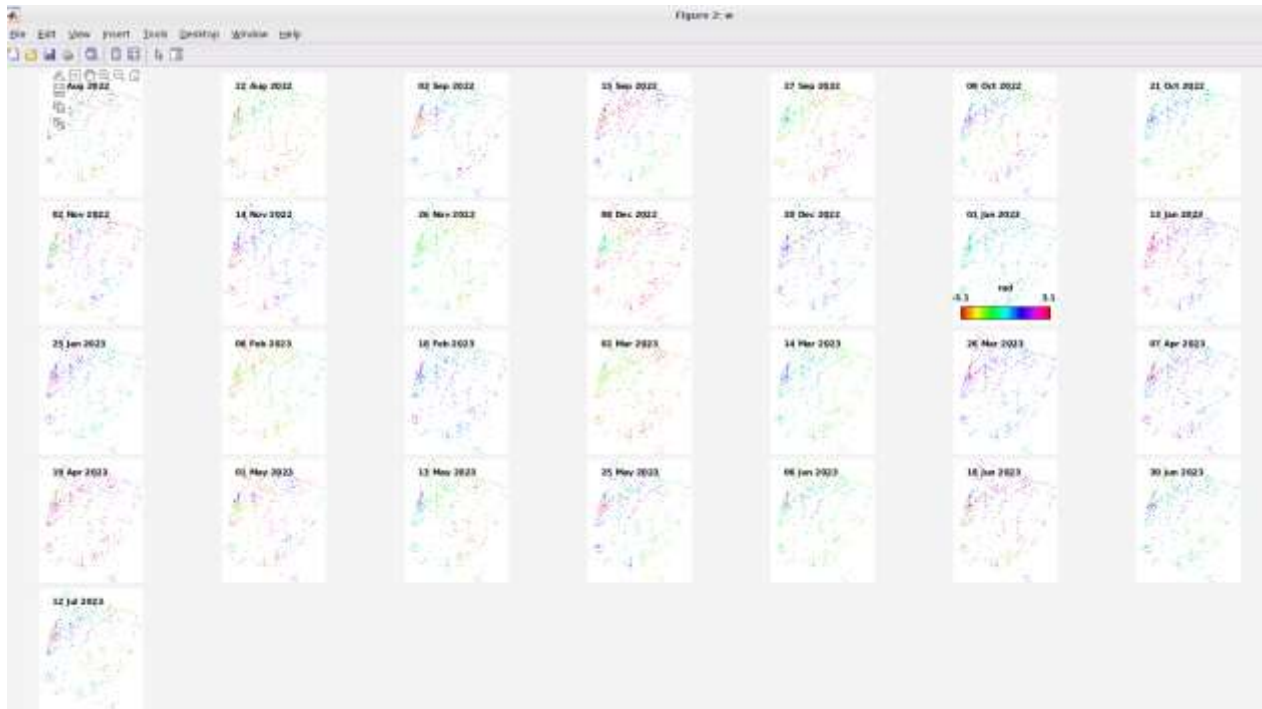


Figure 55: All wrapped interferograms.

Step 6

Step6 is the phase unwrapping, consist of knowing the fraction of the phase difference on the two 2π cycles and reconstructing the original phase shift. For Sentinel one cycle of the wavelength 2π corresponds to 5,5 cm, meaning that 0 to 2π is 5.5cm, and the maximum fraction of the phase calculate between images is 1.5cm. This is called phase ambiguity, and this is solved during the phase unwrapping which is the process of reconstructing the original phase shift from this wrapped representation. This is solved adding and subtracting multiples of two in the appropriated places to make the phase image as smooth as possible in space and time, to reconstruct the Absolut original phase difference.

This step is ruled by several important parameters:

“unwrap_method” that is set ad 3D it takes in to account the unwrapping not just in space, also in time.

“unwrap_prefilter_flag” set to yes 'y' it will provoke to do a pre-filtered before unwrapping to reduce noise. This means that the spatial correlated look up angle error and the master atmosphere and also the orbit error which are actually estimated on the step 7, (so we have to rerun step six after step seven again), are removed to improve the unwrapping.

“unwrap_grid_size” this is also linked to the pre-filtering before unwrapping to reduce noise. If this parameter is set to yes, then the phase is resampled to this grid and it should be at least as large as the merge grid size. The high values can reduce noise but may lead to under sampling of their signal, so by default this is set 200. This parameter we will leave it as default.

The parameter that are linked to the gold strain filter are:

“unwrap_gold_alpha” and “unwrap_gold_n_win”.

“unwrap_patch_phase” this parameter by default is enabled, this allows the unwrapping for each patch separately.

“unwrap_time_win” which is a smoothing window in days for the unwrapping in the temporal domain.

To run the step, we leave all the parameters as default. We run the process typing “>> stamps (6.6)””.

After the process is finished, we can visualize the first unwrapped interferogram typing “ps_plot (‘u’, 1, 0, 0, 1)” the phase from the interferogram Figure 56 is form 12.2 to – 7.7 rad.

```
>> ps_plot('u', 1, 0, 0, 1)
```

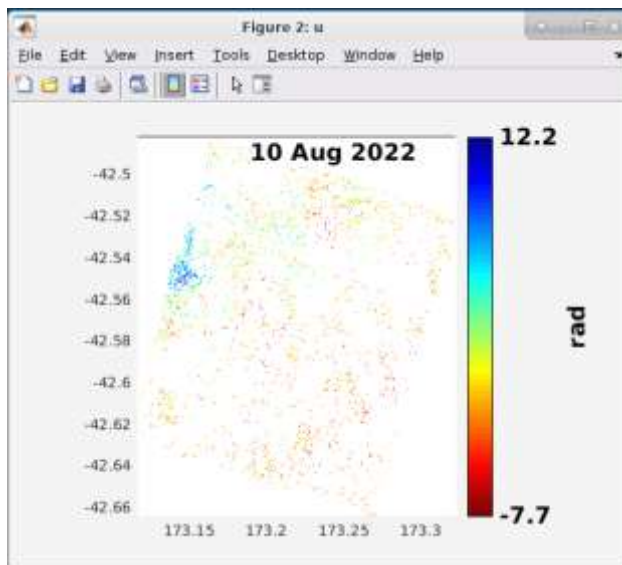


Figure 56: Unwrapping phase.

Atmospheric correction

The next step is to estimate the atmospheric phase screen or the phase delay due to atmosphere. This step is done by the dependency called TRAIN. The Atmospheric correction for InSAR is really complex topic but for this process we will use the linear tropospheric correction that can be computed based on phase and topographic information. This can be run writing “aps_linear” Figure 57 and the tropospheric contribution will be calculated Figure 58. Each estimation should be different for every interferogram

```
>> aps_linear
```

Figure 57: Atmospheric correction command.

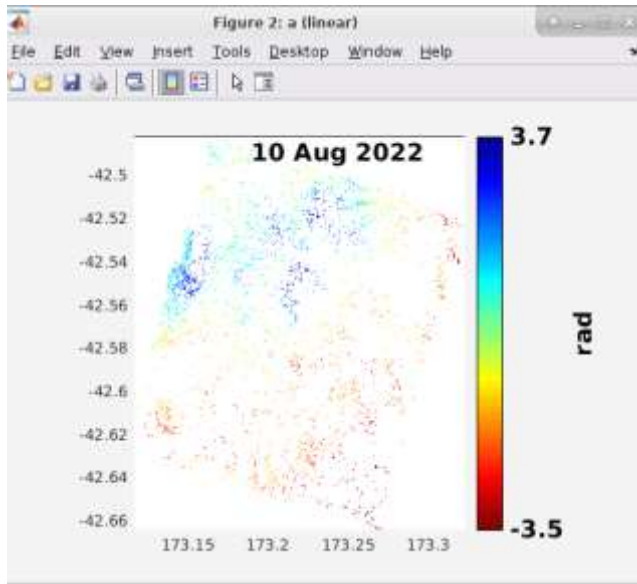


Figure 58: Plot of the atmospheric correction or the tropospheric contribution to the phase.

Step 7

The stamps step 7 is the estimation of spatially correlated errors. In the step 3 we have calculated the spatially uncorrelated error for look angle, then we have removed this error in step 5 and now in step 7 we will remove the spatially correlated look angle error. This in step 7 includes the error in the digital elevation model itself, for example incorrect mapping of the DEM in to the radar coordinates. The master atmosphere and the orbit error are also estimated at this step and we can visualize them at the end.

For this step **we are going to set one parameter** “scla_deramp” Figure 59 if this parameter is set to yes, a face ramp is estimated for each interferogram and the estimated ramp is subtracted. This is useful in local signals but is good to see the effect to the study area. We set up the parameter and after we run the “>> stamps(7.7)”.

Once the process is done we can visualize the spatially correlated errors also the master atmosphere and the estimated ramps. We do it with the “ps_plot(‘d’)” we will visualize the total DEM error radian per meter per baseline Figure 60 It is really low between 0.0154 and -0.0175. The next thing we can check is the master atmosphere. Before we have estimated the tropospheric correction or the tropospheric contribution to the phase and this step also estimates the contribution of the master atmosphere. So, we write “ps_plot(‘m’)” Figure 61 it is observe values much more larger and the next one we can visualize the estimated ramps “ps_plot(‘o’)” Figure 62.

```
>> setparam('scla_deramp', 'y')
SETPARM: scla_deramp = y
```

Figure 59: Parameter for phase ramp estimation and subtraction.

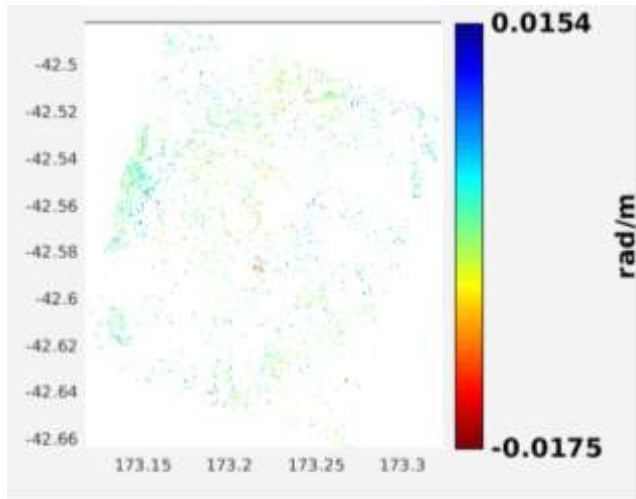


Figure 60: Error per meter per baseline is really low.

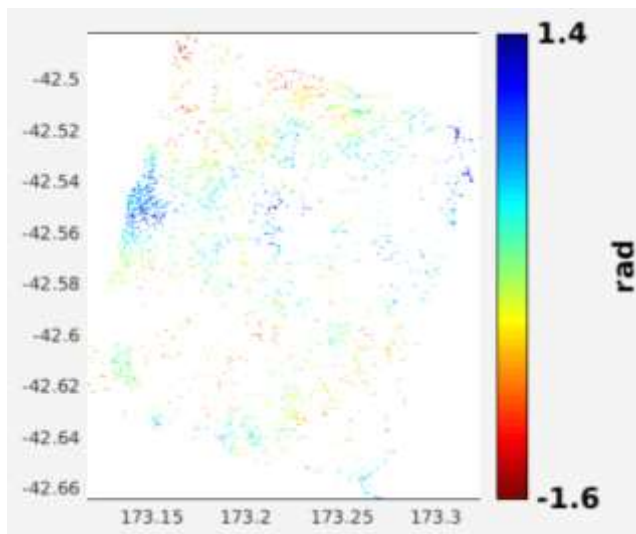


Figure 61: Contribution of the master atmosphere.

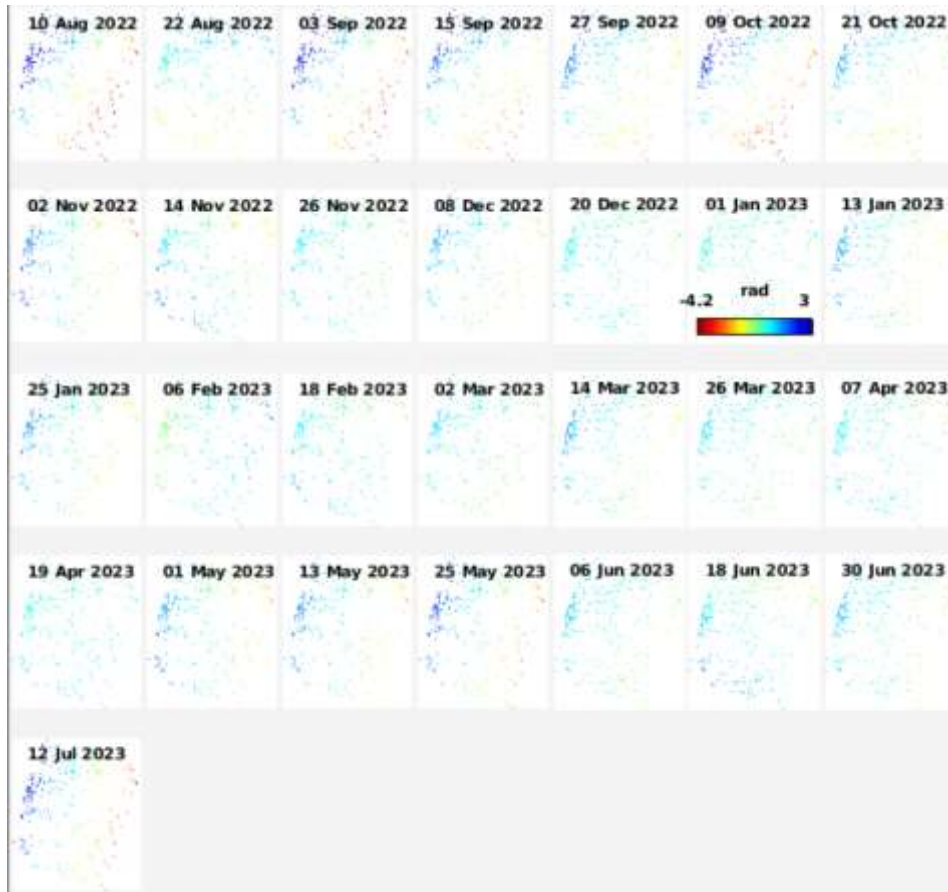


Figure 62: Plot of the estimated ramps.

Rerun the step 6 and 7

It is appropriate to rerun the step 6 to remove the parameter of (scla) error this corresponds of the spatially correlated look angle error and the master atmosphere and the orbit error that we have just estimated. Now we will rerun the phase unwrapping because all this phase contributions errors would be removed before the phase unwrapping and we will have a phase with less noise. Now that we have the estimated noise we can visualized typing "ps_info" Figure 63 the value of degrees od noise. In order to re-run the steps we will type ">> stamps (6.7)". Now we can compare with the phase unwrapped before and now Figure 64.

```

>> ps_info
  1 10-Aug-2022  -77 m  78.611 deg
  2 22-Aug-2022  -55 m  55.105 deg
  3 03-Sep-2022  173 m  48.108 deg
  4 15-Sep-2022   25 m  46.844 deg
  5 27-Sep-2022 -106 m  44.888 deg
  6 09-Oct-2022 -140 m  47.220 deg

```

Figure 63: Estimated noise for every interferogram.

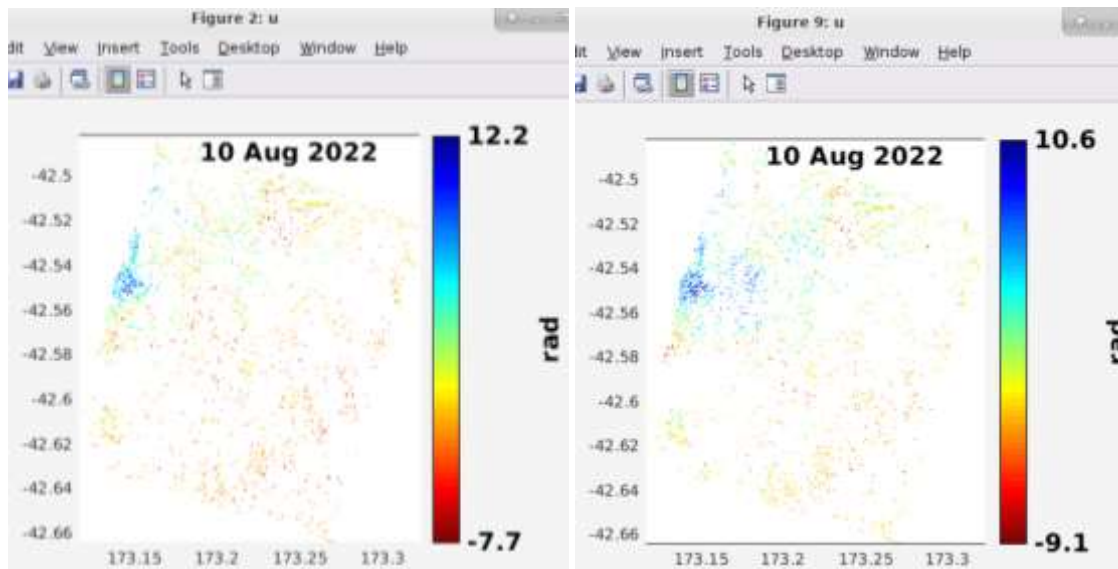


Figure 64: Unwrapping phase on left (before), and unwrapping phase without error in the right (now).

Also, in this same step we can do the estimation of deformation velocity. We type “ps_plot(‘v’) and it appear this Figure 65 velocity. This velocity is estimated form the unwrap phase but without the DEM errors the ramps and atmosphere removed. This means that this is the first one and we have to remove all this to obtain purely the velocity related to the topography deformation, this can be done by typing “ps_plot(‘v-dao’, ‘a_linear’)” and the result is in Figure 66, we can see that the velocity is much more clean in the second estimation of velocity. The Figure 67 shows all the time states with the velocity mm/yr. This velocity values that we have visualize before are estimated from the relative mean velocity of the entire image, so zero velocity is not actually zero. In order to know exactly what the absolute velocity values are in the loss direction, line of sight direction, we need to set a reference velocity by setting a point, area or known velocity. If we do not have any reference point is got to set a point where the movement is not expected. We can set this in the set parameters typing “set_parm(‘ref_lat’, [min max])” and “set_parm(‘ref_lon’, [min max])”. In this case I can not show the result and the final velocity with a reference point because my virtual machine is block at the time.

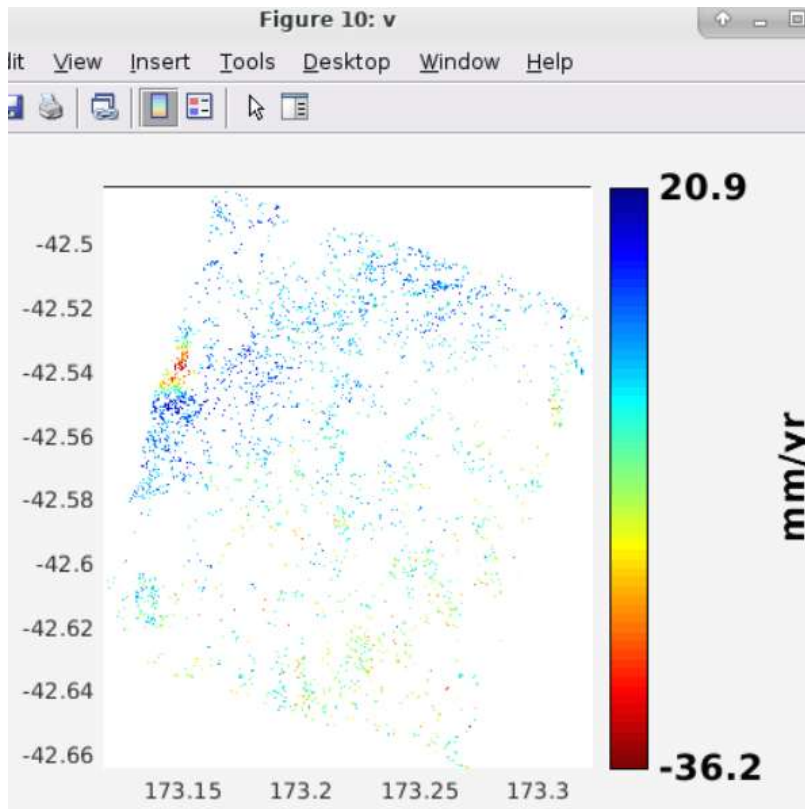


Figure 65: Velocity estimated from the unwrap phase but without the DEM errors the ramps and atmosphere removed.

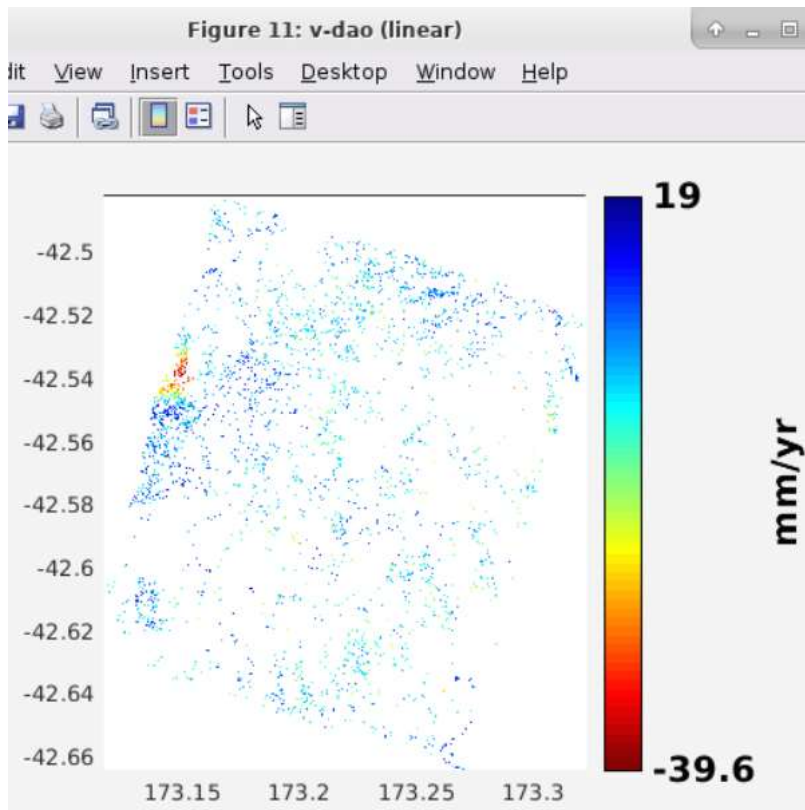


Figure 66: Velocity estimated from the unwrap phase.

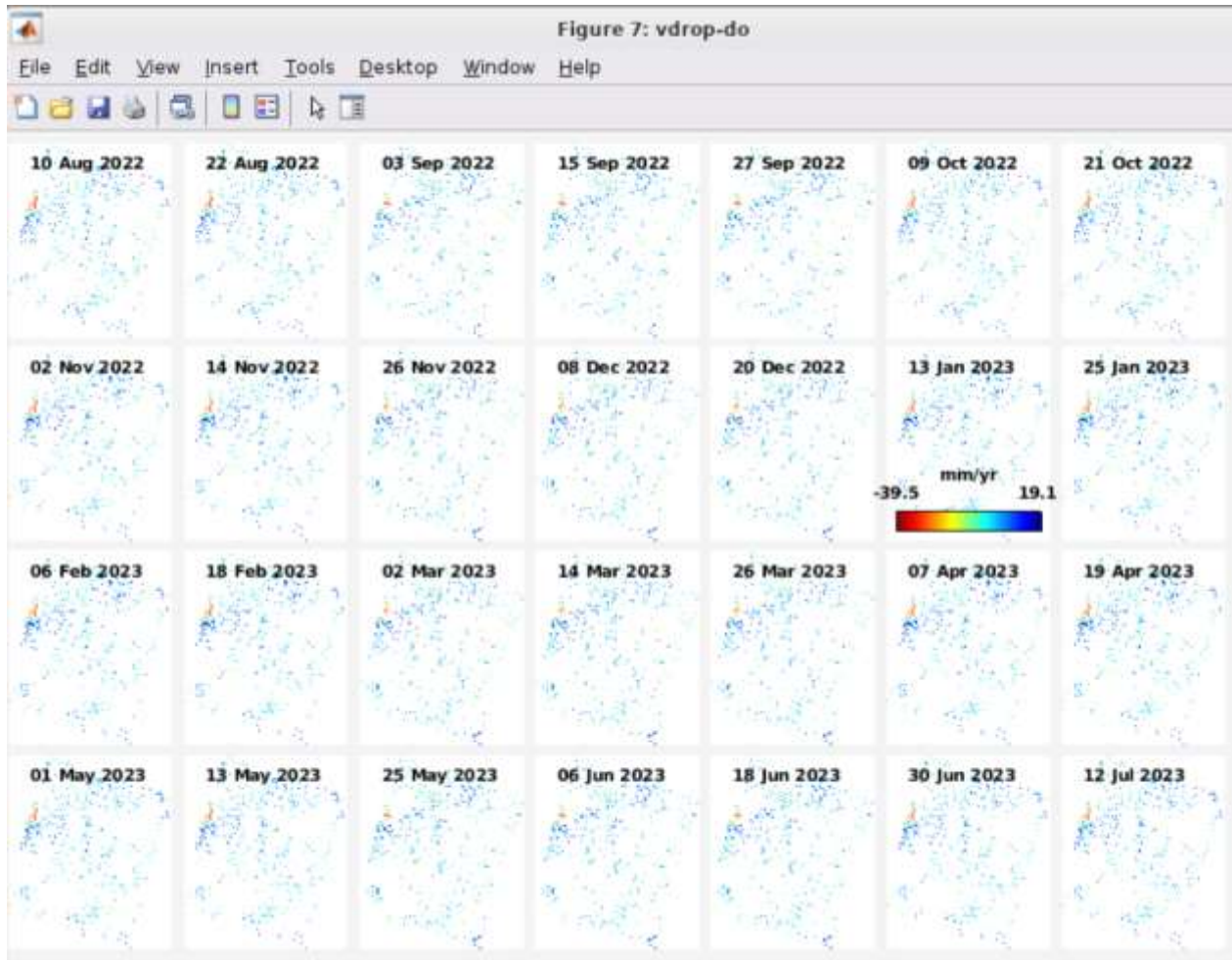


Figure 67: All the time states with the velocity mm/yr.

Certain highlighted issues can include the difficult installation and dependencies of all the software for the specific Linux version. A definitive inconvenience is the unavailability to have root permits to install directly in the virtual machine (copy and paste some libraries of python) and the difficulty to reference the variables inside the VM.

It would be a good idea to be able to have root permits in the VM or being able to update the Ubuntu version that is recommended for the software that is going to be used.

Conclusion

To sum up, this study demonstrates the feasibility of conducting Interferometry with free software. The process of installation and debugging of the software errors and compatibilities have proven to be really time consuming. Now there are clear guidelines to be able to execute this process smoothly. The next steps would be to understand all the parameters available in the software to perform a better processing and to perform a SBAS process for the same area. The objective was accomplished. For future investigations, it would be nice to be able to perform the same processing with other software's.

Bibliography

Bamler, R., & Hartl, P. (1998). Synthetic Aperture Radar Interferometry. *Inverse Problems*, 14.

Crosetto, M., Monserrat, O., Cuevas-González, M., Devanthery, N., & Crippa, B. (2016). Persistent Scatterer Interferometry: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 115, 78–89. <https://doi.org/10.1016/j.isprsjprs.2015.10.011>

Hrysiewicz, A., Wang, X., & Holohan, E. P. (2023). EZ-InSAR: An easy-to-use open-source toolbox for mapping ground surface deformation using satellite interferometric synthetic aperture radar. *Earth Science Informatics*, 16(2), 1929–1945. <https://doi.org/10.1007/s12145-023-00973-1>

Pepe, A., & Calò, F. (2017). A Review of Interferometric Synthetic Aperture RADAR (InSAR) Multi-Track Approaches for the Retrieval of Earth's Surface Displacements. *Applied Sciences*, 7(12), Article 12. <https://doi.org/10.3390/app7121264>