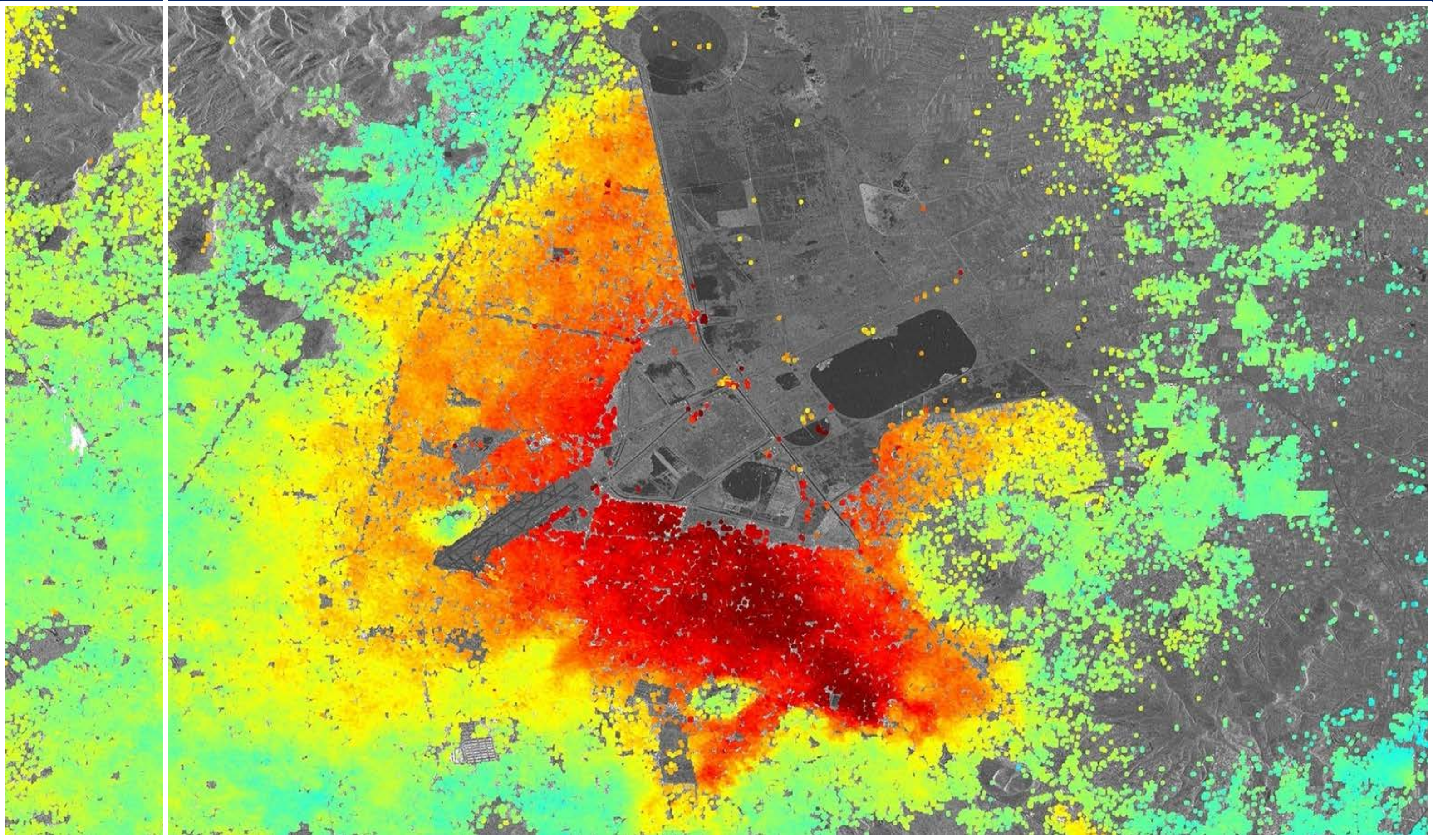


TRAINING KIT – HAZA12

StaMPS: Persistent Scatterer Interferometry Processing

Case Study: Mexico City, Nov. 2019 - Nov. 2020

Research and User Support for Sentinel Core Products

The RUS Service is funded by the European Commission, managed by the European Space Agency and operated by CSSI and its partners.

Authors would be glad to receive your feedback or suggestions and to know how this material was used. Please, contact us on training@rus-coperenicus.eu

Cover images produced by RUS Copernicus

The following training material has been prepared by Serco Italia S.p.A. within the RUS Copernicus project.

Date of publication: June 2021

Version: 1.1

Suggested citation:

Serco Italia SPA (2020). *StaMPS: Presistent Scatterer Interferometry Processing - Mexico City 2021 (version 1.1).* Retrieved from RUS Lectures at https://rus-copernicus.eu/portal/the-rus-library/learn-by-yourself/

Table of Contents

1 Introduction

The Research and User Support for Sentinel core products (RUS) service provides a free and open scalable platform in a powerful computing environment, hosting a suite of open-source toolboxes pre-installed on virtual machines, to handle and process data derived from the Copernicus Sentinel satellites constellation. In this tutorial we will employ RUS to identify and map land subsidence in Mexico City using Sentinel-1 data.

Mexico City's buildings are seriously leaning due to land subsidence. Photo credit: JOSH HANER/THE NEW YORK TIMES (http://www.sciencemag.org)

Land subsidence in Mexico City caused by groundwater overexploitation over the last century has been more than 9 meters, resulting in damages to buildings, streets, sidewalks, sewers, storm water drains and other infrastructure [1].

Due to the fact that the city is partially built on the area of a former lake (Lago Texcoco), it rests on the heavily saturated clay which is collapsing due to the over-extraction of groundwater. Current subsidence rates using Sentinel-1 SAR data approximate 2.5 cm/month [3].

Persistent Scatterer Interferometry (PSI) is a powerful advanced DInSAR technique able to measure and monitor displacements of the Earth's surface over time with hight accuracy. Hooper et al. (2004). proposed a novel PS selection using phase characteristics, which is suitable to find low-amplitude natural targets with phase stability that cannot be identified by amplitude-based algorithms. This work originated one of the most widely used PSI software packages, StaMPS. [1]

SNAP2StaMPS is a Python workflow developed by José Manuel Delgado Blasco and Michael Foumelis in collaboration with Prof. A. Hooper to automate the pre-processing of Sentinel-1 SLC data and their preparation for ingestion to StaMPS.

2 Training

Approximate time needed to complete this training session is two hours.

2.1 Data used

- 31 Sentinel-1A images acquired between November 1, 2019 and November 1, 2020. [downloadable @ASF Data Search portal – Vertex]
- Auxiliary data stored locally @ /shared/Training/HAZA09_SNAP2StaMPs_DataPrep_TutorialKit/AuxData

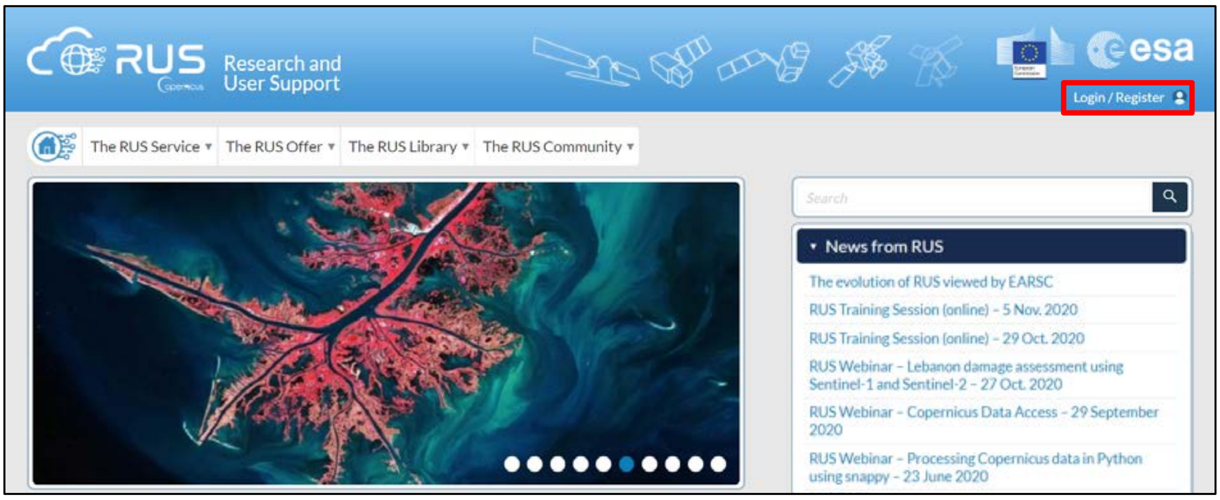
2.2 Software in RUS environment

MATLAB, StaMPS, TRAIN, StaMPS Visualizer, snaphu

(Commercial SW not available in RUS! If you wish to replay this exercise you have to secure your own MATLAB license)

3 Register to RUS Copernicus

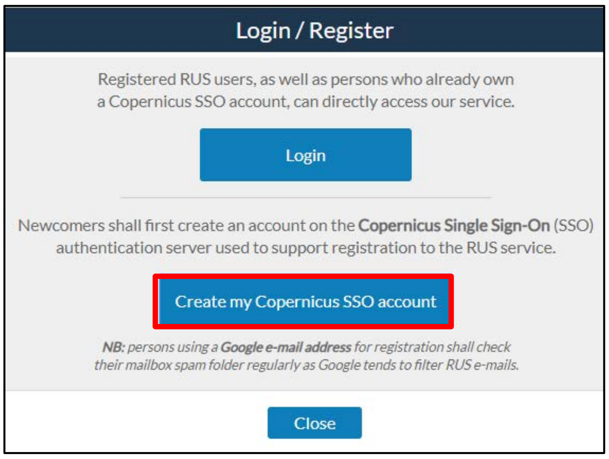
To repeat the exercise using a RUS Copernicus Virtual Machine (VM), you will first have to register as a RUS user. For that, go to the RUS Copernicus website (www.rus-copernicus.eu) and click on ***Login/Register*** in the upper right corner.



News from RUS

- The evolution of RUS viewed by EARSC
- RUS Training Session (online) – 5 Nov. 2020
- RUS Training Session (online) – 29 Oct. 2020
- RUS Webinar – Lebanon damage assessment using Sentinel-1 and Sentinel-2 – 27 Oct. 2020
- RUS Webinar – Copernicus Data Access – 29 September 2020
- RUS Webinar – Processing Copernicus data in Python using snappy – 23 June 2020

Select the option ***Create my Copernicus SSO account*** and then fill in ALL the fields on the **Copernicus Users' Single Sign-On Registration**. Click ***Register***.



Login / Register

Registered RUS users, as well as persons who already own a Copernicus SSO account, can directly access our service.

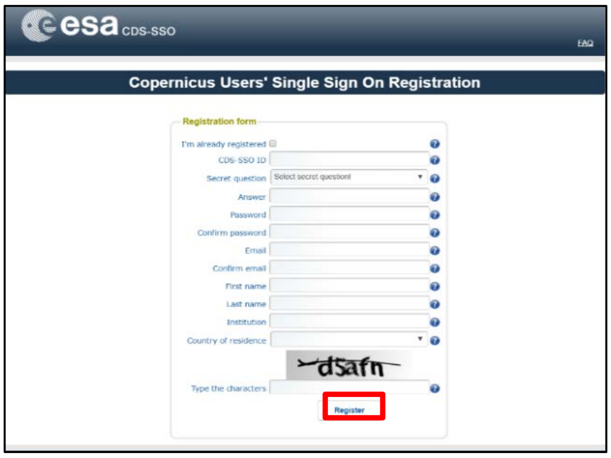
Login

Newcomers shall first create an account on the **Copernicus Single Sign-On** (SSO) authentication server used to support registration to the RUS service.

Create my Copernicus SSO account

*NB: persons using a **Google e-mail address** for registration shall check their mailbox spam folder regularly as Google tends to filter RUS e-mails.*

Close



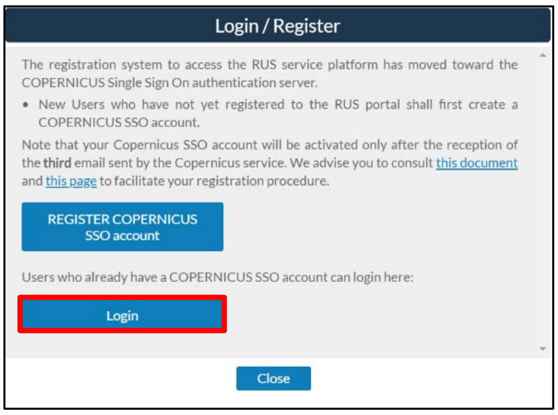
Copernicus Users' Single Sign On Registration

Registration form

I'm already registered	☐
CDS-SSO ID	
Secret question	Select secret question!
Answer	
Password	
Confirm password	
Email	
Confirm email	
First name	
Last name	
Institution	
Country of residence	
Type the characters	

Register

Within a few minutes you will receive an e-mail with activation link. Follow the instructions in the e-mail to activate your account. You can now return to https://rus-copernicus.eu/, click on ***Login/Register***, choose ***Login*** and enter your chosen credentials.



Login / Register

The registration system to access the RUS service platform has moved toward the COPERNICUS Single Sign On authentication server.

- New Users who have not yet registered to the RUS portal shall first create a COPERNICUS SSO account.

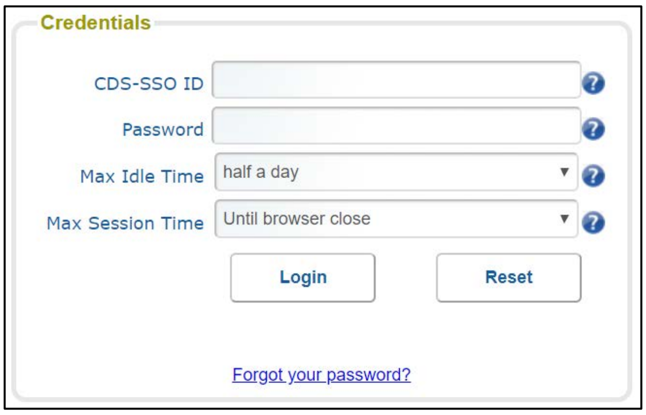
Note that your Copernicus SSO account will be activated only after the reception of the **third** email sent by the Copernicus service. We advise you to consult this document and this page to facilitate your registration procedure.

REGISTER COPERNICUS SSO account

Users who already have a COPERNICUS SSO account can login here:

Login

Close



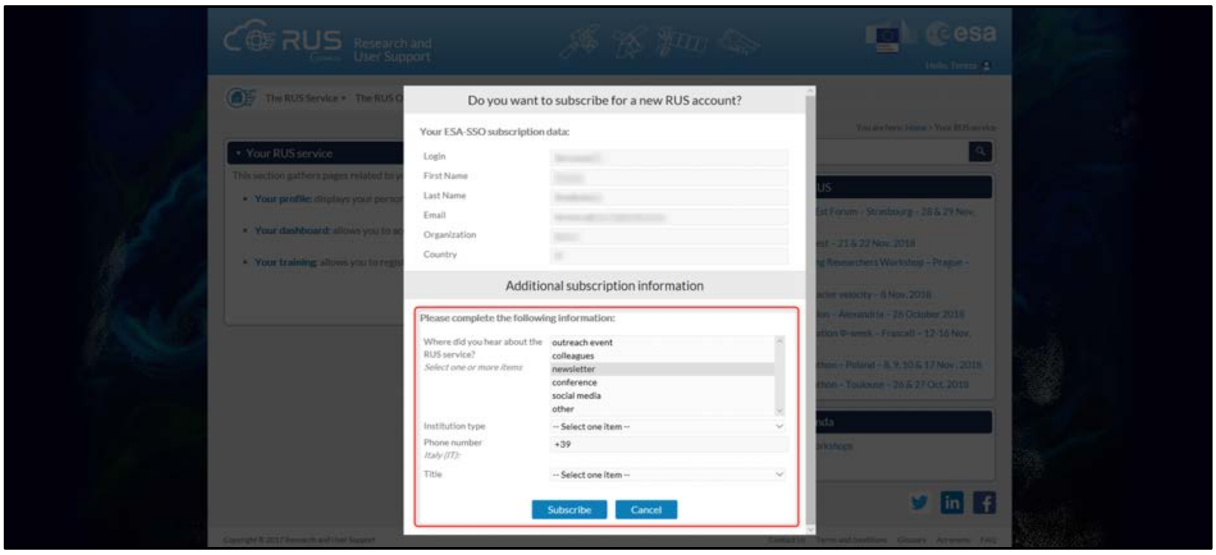
Credentials

CDS-SSO ID	
Password	
Max Idle Time	half a day
Max Session Time	Until browser close

Login Reset

Forgot your password?

Upon your first login you will need to enter some details. You must fill all the fields.



Do you want to subscribe for a new RUS account?

Your ESA-SSO subscription data:

Login	[redacted]
First Name	[redacted]
Last Name	[redacted]
Email	[redacted]
Organization	[redacted]
Country	[redacted]

Additional subscription information

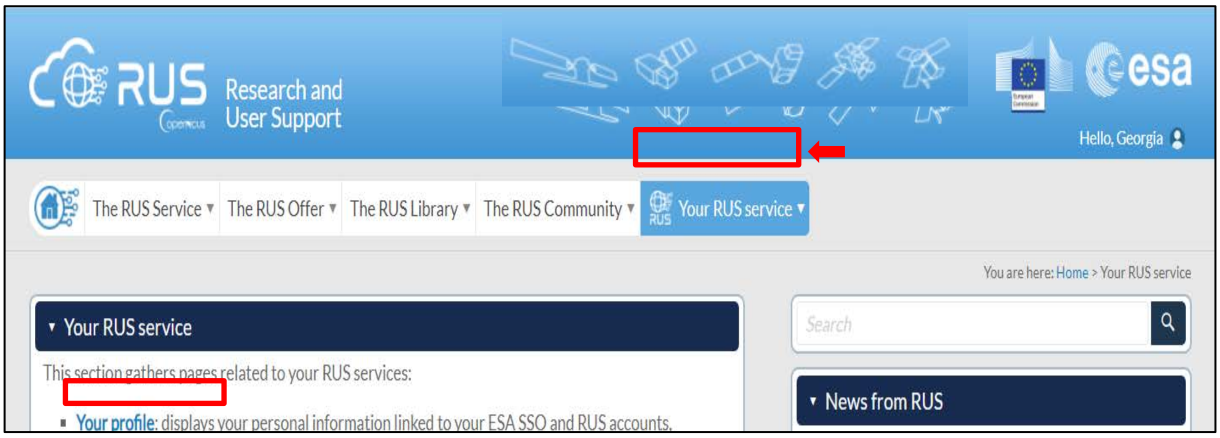
Please complete the following information:

Where did you hear about the RUS service? *Select one or more items*	outreach event colleagues newsletter conference social media other
Institution type	-- Select one item --
Phone number *Italy (IT)-*	+39
Title	-- Select one item --

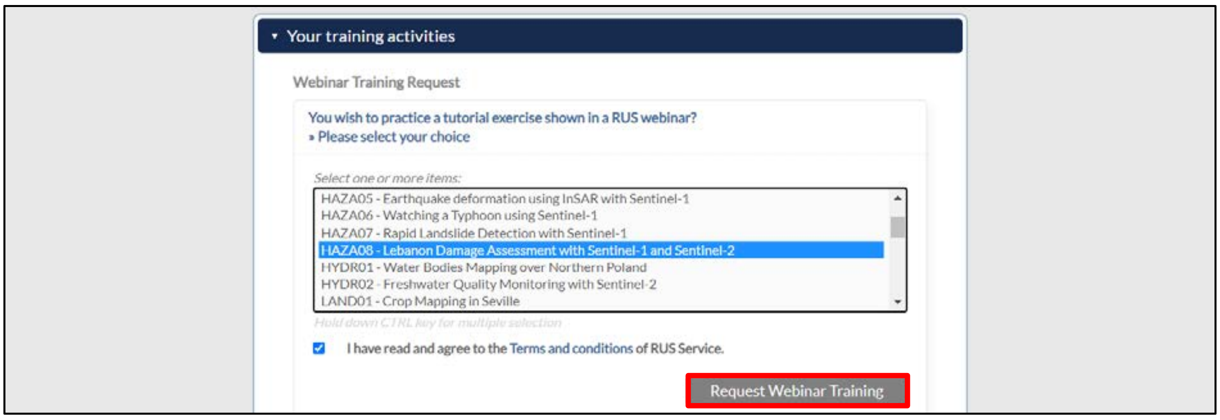
Subscribe Cancel

4 Request a RUS Copernicus Virtual Machine to repeat a Webinar

Once you are registered as a RUS user, you can request a RUS Virtual Machine to repeat this exercise or work on your own projects using Copernicus data. For that, log in and click on ***Your RUS Service*** **→** ***Your training activities***.


RUS Research and User Support
esa
Hello, Georgia
The RUS Service
The RUS Offer
The RUS Library
The RUS Community
Your RUS service
You are here: Home > Your RUS service
Your RUS service
This section gathers pages related to your RUS services:
Your profile: displays your personal information linked to your ESA SSO and RUS accounts.
Search
News from RUS

Select **HAZA12** (includes HAZA09), check the field *"I have read and agree to the Terms and conditions of RUS Service"* and then click on ***Request Webinar Training*** to request your RUS Virtual Machine.



Your training activities

Webinar Training Request

You wish to practice a tutorial exercise shown in a RUS webinar?
» Please select your choice

Select one or more items:

HAZA05 - Earthquake deformation using InSAR with Sentinel-1
HAZA06 - Watching a Typhoon using Sentinel-1
HAZA07 - Rapid Landslide Detection with Sentinel-1
HAZA08 - Lebanon Damage Assessment with Sentinel-1 and Sentinel-2
HYDR01 - Water Bodies Mapping over Northern Poland
HYDR02 - Freshwater Quality Monitoring with Sentinel-2
LAND01 - Crop Mapping in Seville

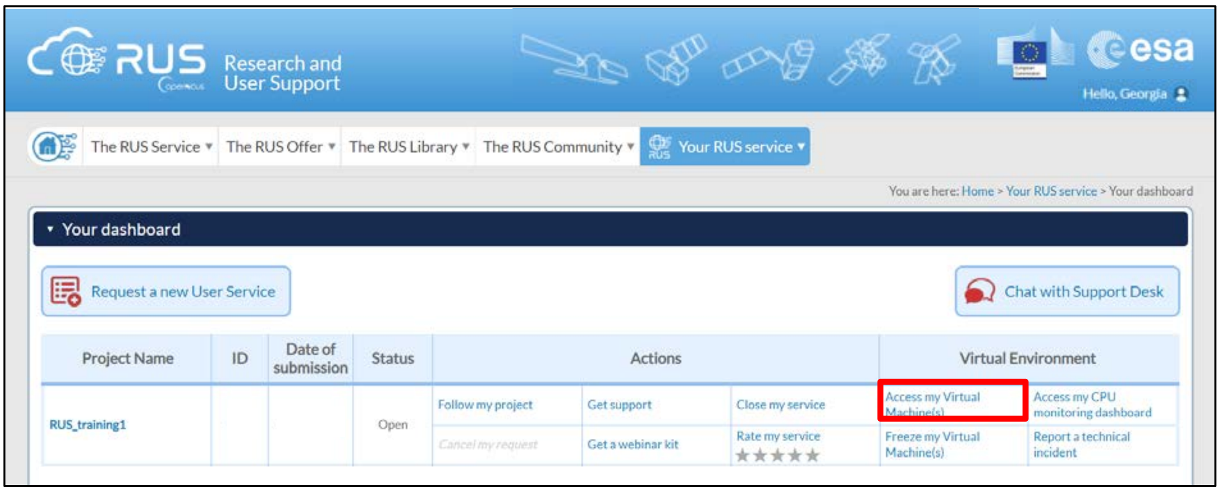
Hold down CTRL key for multiple selection

☑ I have read and agree to the Terms and conditions of RUS Service.

Request Webinar Training

Further to the acceptance of your request by the RUS Helpdesk, you will receive a notification email with all the details about your Virtual Machine.

To access it, go to ***Your RUS Service → Your Dashboard*** and click on ***Access my Virtual Machine***.



You are here: Home > Your RUS service > Your dashboard

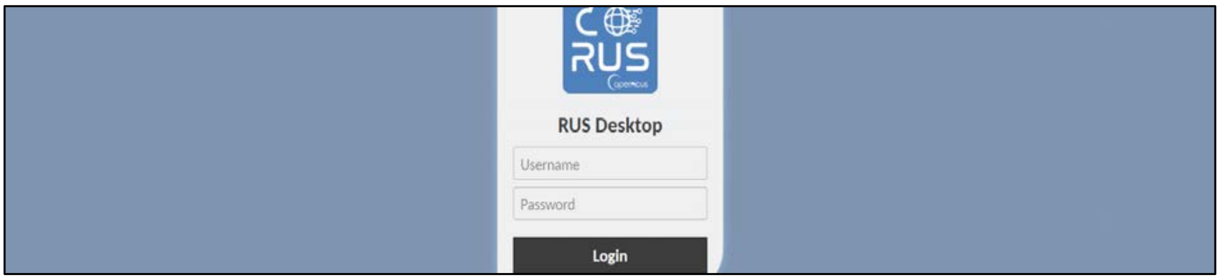
Your dashboard

Request a new User Service

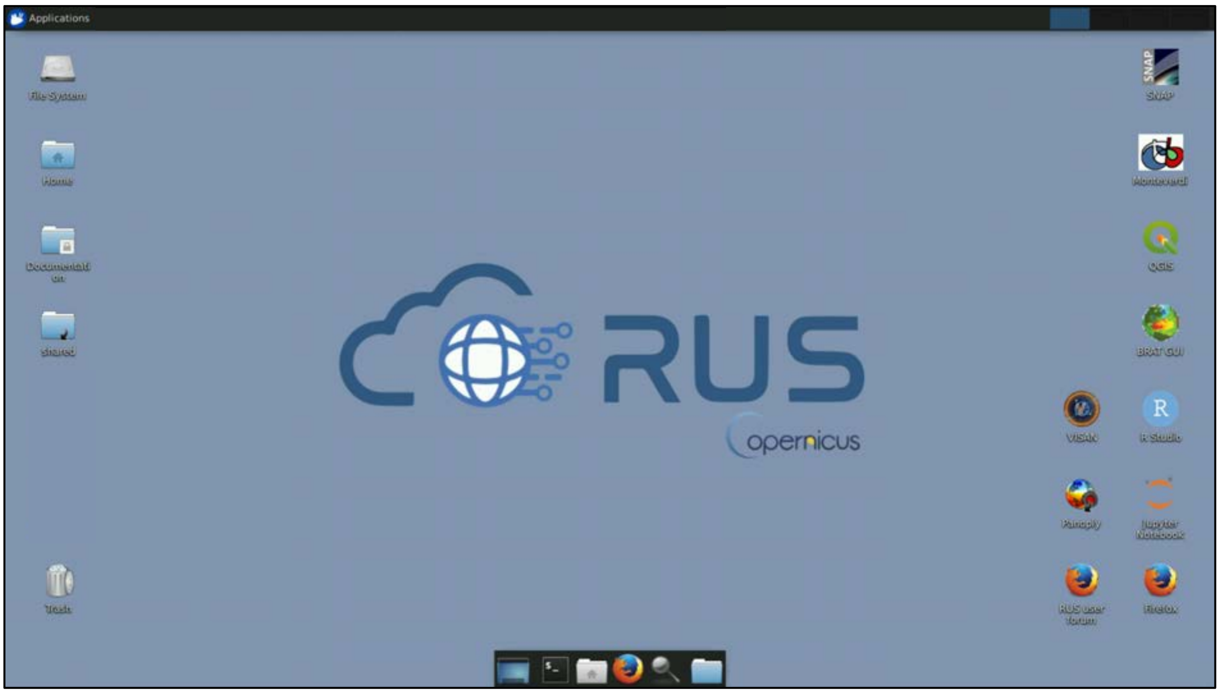
Chat with Support Desk

Project Name	ID	Date of submission	Status	Actions			Virtual Environment	
RUS_training1			Open	Follow my project	Get support	Close my service	Access my Virtual Machine(s)	Access my CPU monitoring dashboard
				Cancel my request	Get a webinar kit	Rate my service ★★★★★	Freeze my Virtual Machine(s)	Report a technical incident

Fill in the login credentials that have been provided to you by the RUS Helpdesk via email to access your RUS Copernicus Virtual Machine.



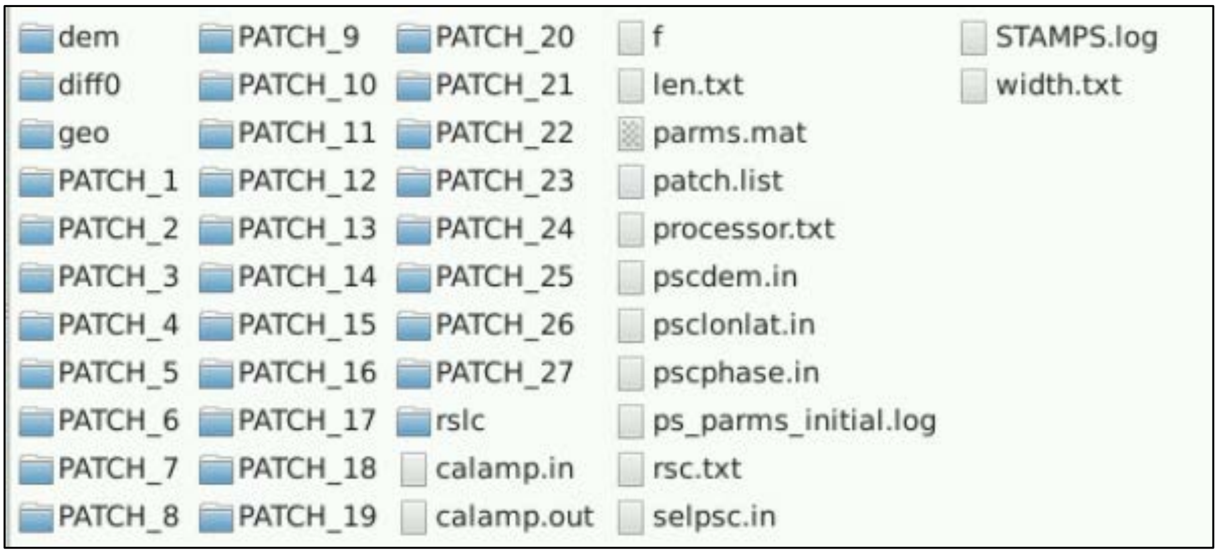
This is the remote desktop of your Virtual Machine.



5 Data download and preparation

In this exercise we will use a dataset prepared during the webinar: ***Data preparation for StaMPS PSI processing with SNAP***. **In order to follow the steps below you first need to prepare the dataset as described here, you can also watch the recording of the webinar available on our YouTube channel - RUS Copernicus Training - or on the RUS Training Portal.**

In the materials you will find all the steps needed to derive the input dataset including the initial data download. The input folder (as derived in the Data Preparation webinar) is named ***"INSAR_20200504"***. The size of the folder is approximately 42.5 Gb, and the structure is as follows:



```
dem        PATCH_9    PATCH_20    f                      STAMPS.log
diff0      PATCH_10   PATCH_21    len.txt                width.txt
geo        PATCH_11   PATCH_22    parms.mat
PATCH_1    PATCH_12   PATCH_23    patch.list
PATCH_2    PATCH_13   PATCH_24    processor.txt
PATCH_3    PATCH_14   PATCH_25    pscdem.in
PATCH_4    PATCH_15   PATCH_26    psclonlat.in
PATCH_5    PATCH_16   PATCH_27    pscphase.in
PATCH_6    PATCH_17   rslc        ps_parms_initial.log
PATCH_7    PATCH_18   calamp.in   rsc.txt
PATCH_8    PATCH_19   calamp.out  selpsc.in
```

TIP: Note that errors have been reported when running StaMPS on Windows – Linux is the preferred environment.

MATLAB should be run from command line (provided that StaMPS_CONFIG.bash is included in .bashrc)

Follow steps outlined in: 1) by Matthias Schlögl in his posts on GIS-Blog
2) in the stamps and snap2stamps manuals
3) StaMPS Persistent Scatterer Exercise by A. Hopper (2015, v3.1)

6 Installation and configuration (Linux)

This chapter will address the setup of the software necessary to perform the exercise as it is not included in the default RUS Copernicus VM installation. The setup here refers to Linux Ubuntu 16.04 OS – if you do not have access to a RUS VM or other Linux PC, you can request the RUS Copernicus Docker image or setup a local virtual environment using for example Virtual Box.

6.1 *MATLAB*

Unfortunately, MATLAB is necessary to run the StaMPS processing and it is a commercial software that requires a license to be run. The license is **NOT** provided by RUS Copernicus and you need to **secure your own paid/trial licence** to repeat the training (even in case of requesting RUS VM).

This exercise was tested with version 2020b and you will need the following MATLAB toolboxes:

- Parallel Computing Toolbox (DM)

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

To install MATLAB for Linux you can follow the steps outlined here.

TIP: If you run to any issues during the installation, they may be related to the libstdc++6, gcc-4.9 libraries, to fix the issue you can upgrade the libraries by running the following commands.

```
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt-get update
sudo apt-get install gcc-4.9
sudo apt-get upgrade libstdc++6
```

After MATLAB is installed, we need to install the dependencies – these include GNU awk, the tcsh unix shell, the matlab support package, the build-management-tool, and the C++ compiler:

```
sudo apt-get install gawk
sudo apt-get install tcsh
sudo apt-get install matlab-support
sudo apt-get make
sudo apt-get build-essential g++
```

During the matlab-support installation you will be asked if you wish to rename the gcc libraries type: “n”

6.2 TRAIN

The Toolbox for Reducing Atmospheric InSAR Noise – ***TRAIN*** – is developed in an effort to include current state of the art tropospheric correction methods into the default InSAR processing chain. Initial development was performed at the University of Leeds. We request ***TRAIN*** users to reference our publication of ***TRAIN***:

Bekaert, D.P.S., Walters, R.J., Wright, T.J., Hooper, A.J., and Parker, D.J. (2015c), ***Statistical comparison of InSAR tropospheric correction techniques****, Remote Sensing of Environment, doi: 10.1016/j.rse.2015.08.035*

There is no need to install TRAIN, just download it from the GitHub repository and place it to the ***/home/rus*** folder **(/home/<username>).**

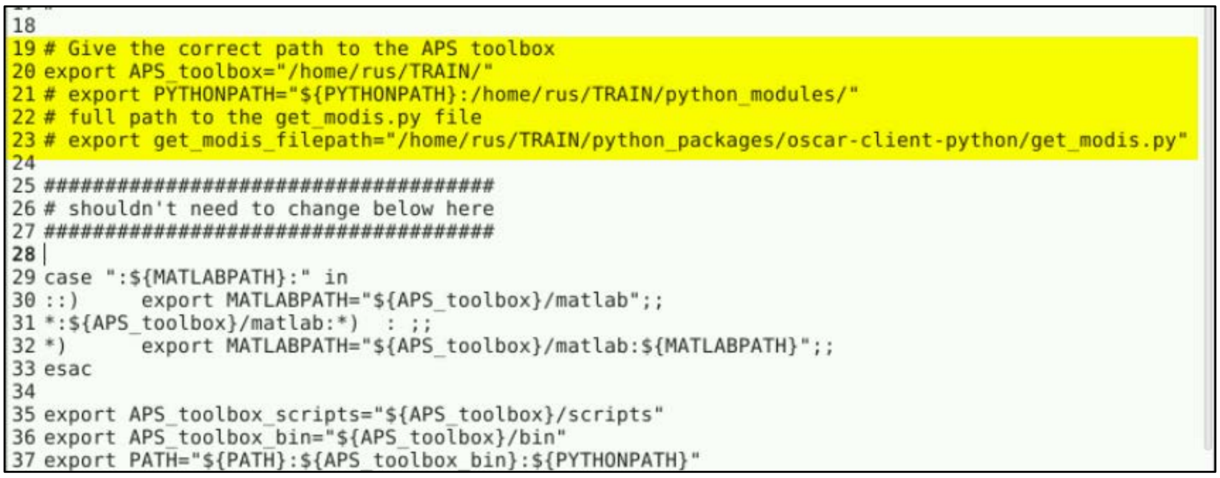
```
git clone https://github.com/dbekaert/TRAIN.git
```

The folder containing the toolbox will be downloaded to ***/home/rus/*** directory (or downloads dir if not on RS VM), we can leave it there. To configure it enter the TRAIN folder and right click the ***APS_CONFIG.sh*** file → Open with Mousepad (or other text editor).

Go to **View → Line numbers**

Now edit the lines 20, 21 and 23 to correspond to the toolbox location in your machine.

```
# Give the correct path to the APS toolbox
export APS_toolbox="/home/rus/TRAIN/"
# export PYTHONPATH="${PYTHONPATH}:/home/rus/TRAIN/python_modules/"
# full path to the get_modis.py file
# export get_modis_filepath="/home/rus/TRAIN/python_packages/oscar-client-
python/get_modis.py"
```



```

# Give the correct path to the APS toolbox
export APS_toolbox="/home/rus/TRAIN/"
# export PYTHONPATH="${PYTHONPATH}:/home/rus/TRAIN/python_modules/"
# full path to the get_modis.py file
# export get_modis_filepath="/home/rus/TRAIN/python_packages/oscar-client-python/get_modis.py"

######################################
# shouldn't need to change below here
######################################

case ":${MATLABPATH}:" in
::)      export MATLABPATH="${APS_toolbox}/matlab";;
*:${APS_toolbox}/matlab:*)  : ;;
*)       export MATLABPATH="${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}"
```

Save the edits.

6.3 Snaphu (Installed in RUS VMs)

Snaphu is installed by default in the RUS VMs (the installation path is /usr/local/snaphu). If snaphu needs to be installed in your system you can follow the steps outlined below.

SNAPHU is an implementation of the Statistical-cost, Network-flow Algorithm for Phase Unwrapping proposed by Chen and Zebker (see references below). This algorithm poses phase unwrapping as a maximum *a posteriori* probability (MAP) estimation problem, the objective of which is to compute the most likely unwrapped solution given the observable input data. (SNAPHU (stanford.edu))

C. W. Chen and H. A. Zebker, ``Network approaches to two-dimensional phase unwrapping: intractability and two new algorithms,'' Journal of the Optical Society of America A, vol. 17, pp. 401-414 (2000).

To install snaphu, open the terminal and run the following commands:

```
sudo apt-get update
sudo apt-get install snaphu
```

In case you follow the above installation steps, snaphu will be likely installed to ***/usr/local/snaphu***

6.4 Triangle

Triangle generates exact Delaunay triangulations, constrained Delaunay triangulations, conforming Delaunay triangulations, Voronoi diagrams, and high-quality triangular meshes. The latter can be generated with no small or large angles and are thus suitable for finite element analysis. (Triangle: A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator (cmu.edu))

To install triangle, open the terminal and run the following command:

```
sudo apt-get install triangle-bin
```

In case you follow the above installation steps, triangle will be likely installed to ***/usr/local/triangle***

6.5 StaMPS

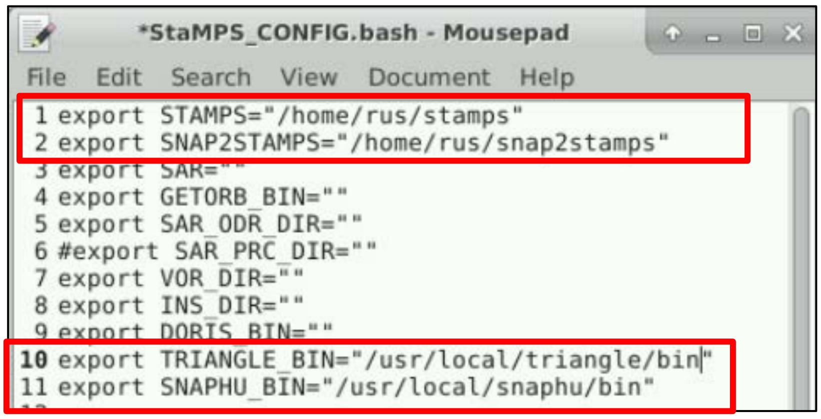
Download StaMPS from the GitHub repository - https://github.com/dbekaert/StaMPS/tree/v4.1-beta

Go to the download location and extract the downloaded archive to /stamps folder. In the terminal type:

```
cd StaMPS-4.1b/src
make
make install
```

The tool will be installed in the same folder from which the installation is run.

To configure StaMPS, go back to the ***/stamps*** folder and open the configuration file ***StaMPS_CONFIG.bash*** in a text editor (e.g., right-click on the file and select **Open in Moustepad**).



*StaMPS_CONFIG.bash - Mousepad
File Edit Search View Document Help

```
1 export STAMPS="/home/rus/stamps"
2 export SNAP2STAMPS="/home/rus/snap2stamps"
3 export SAR=""
4 export GETORB_BIN=""
5 export SAR_ODR_DIR=""
6 #export SAR_PRC_DIR=""
7 export VOR_DIR=""
8 export INS_DIR=""
9 export DORIS_BIN=""
10 export TRIANGLE_BIN="/usr/local/triangle/bin"
11 export SNAPHU_BIN="/usr/local/snaphu/bin"
```

Edit the paths in line 1,2,10,11 to correspond to your system – they may not be the same as shown above (hints to where the installation folders might be located are given for each SW above). SNAP2StaMPS was used to prepare the input data and can be downloaded here – no installation is required.

Finally navigate to your user folder ***(/home/rus/)*** and open the ***.bashrc*** file (this file is usually hidden and you need to make it visible by going to **View → Show Hidden Files**). Open the file in a text editor and scroll to the bottom. Add the following lines pointing to the configuration files for StaMPS and TRAIN.

```
source /home/rus/stamps/StaMPS_CONFIG.bash
source /home/rus/TRAIN/APS_CONFIG.sh
```

6.6 StaMPS Visualizer

Requires R and R Studio installed (Check their websites for installation instructions). Download the StaMPS Visualizer - GitHub - thho/StaMPS_Visualizer: Shiny application to visualize DInSAR results processed by StaMPS/MTI. Extract the folder to ***/home/rus/StaMPS_Visualizer*** (or corresponding folder on your machine).

Launch the R Studio and load the ***/home/rus/StaMPS_Visualizer/install_packages.R***. Run all the lines in the script to install the necessary packages.

Once the packages are installed (it will take a while) open the ***/home/rus/StaMPS_Visualizer/ui.R,*** in the upper left corner you can see a button **Run app**. Click on it to open the StaMPS Visualizer window.

7 Step by step

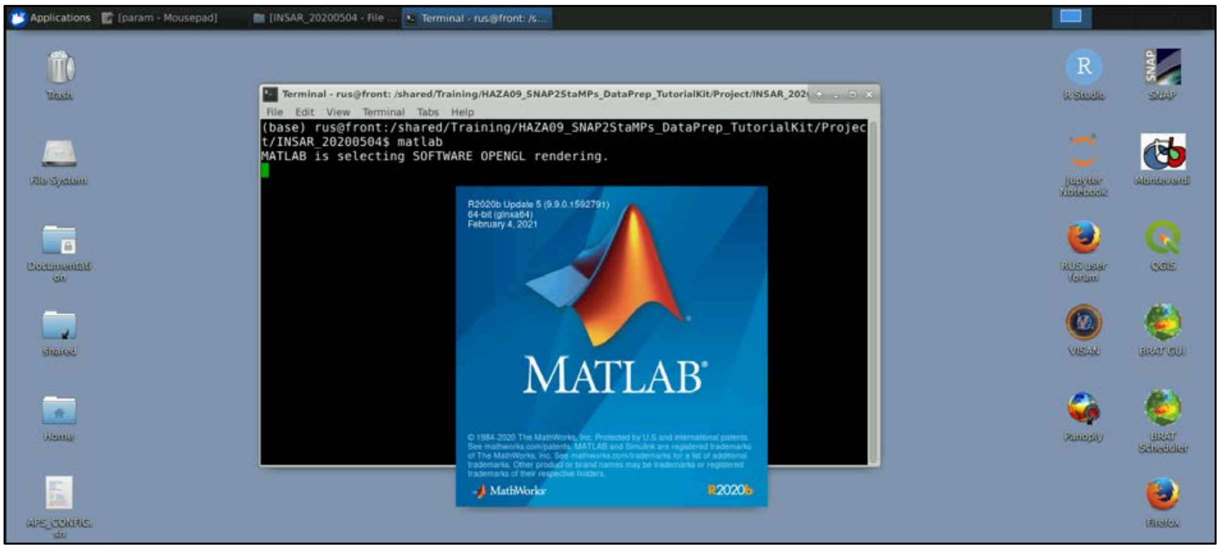
In the file explorer navigate to the StaMPS input folder (Prepared following the steps outlined in the SNAP2StaMPS tutorial).

/shared/Training/HAZA09_SNAP2StaMPs_DataPrep_TutorialKit/Project/INSAR_20200504/

Right-click on the white space in the folder and select Open Terminal Here. A terminal window will open. In the terminal type and run:

```
matlab
```

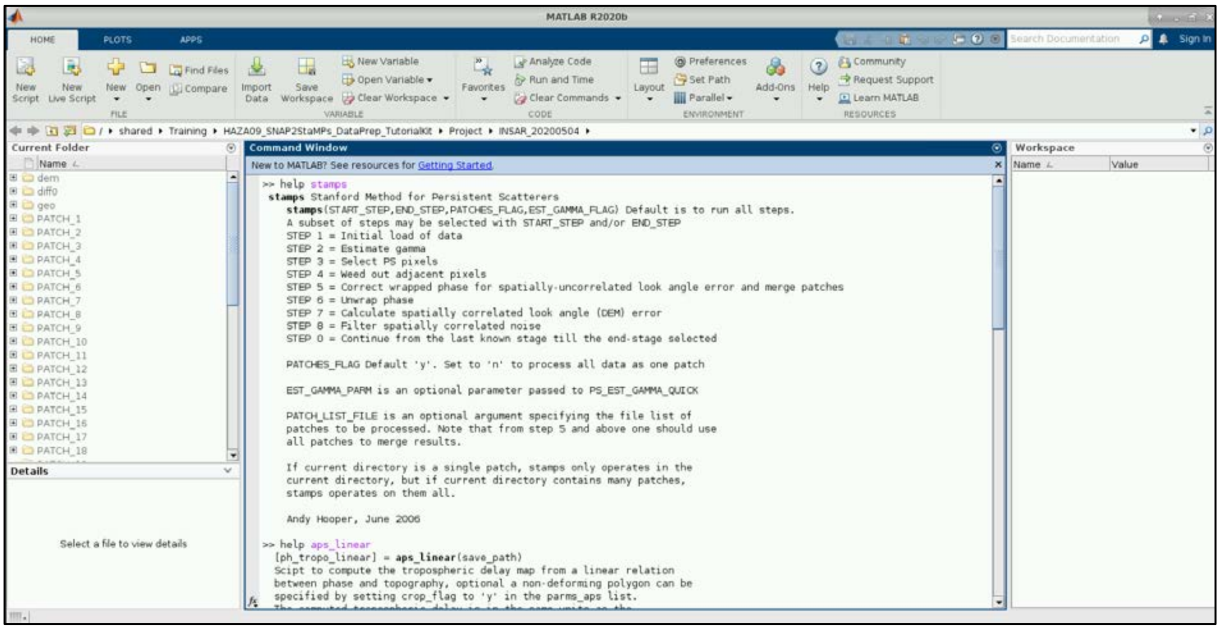
If MATLAB is correctly installed, then the MATLAB window will open. *To work with StaMPS always open MATLAB from the **terminal** (do not use desktop or Start icons otherwise you will not be able to run StaMPS commands).*



```
(base) rus@front:/shared/Training/HAZA09_SNAP2StaMPs_DataPrep_TutorialKit/Projec
t/INSAR_20200504$ matlab
MATLAB is selecting SOFTWARE OPENGL rendering.
```

Once the MATLAB window opens, test the StaMPS and TRAIN setup by typing:

```
help stamps
help aps_linear
```



```
>> 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

>> help aps_linear
  [ph_tropo_linear] = aps_linear(save_path)
  Scipt to compute the tropospheric delay map from a linear relation
  between phase and topography, optional a non-deforming polygon can be
  specified by setting crop_flag to 'y' in the parms_aps list.
```

If you don't receive any errors, then all is set, and we can start with the exercise.

7.1 *Check the parameters*

To check the initial parameters, type:

```
getparm
```

For the moment we will not change any, but we will adapt them at the appropriate steps.

7.2 *STEP 1: Load the PS Candidates*

The stamps commands 1 to 4 can be run either for each patch separately (by entering the patch folders and running the commands from there) or for all patches at the same time (by running the command from the main directory).

We will run the commands 1 - 4 first for a single patch → PATCH_14. Type:

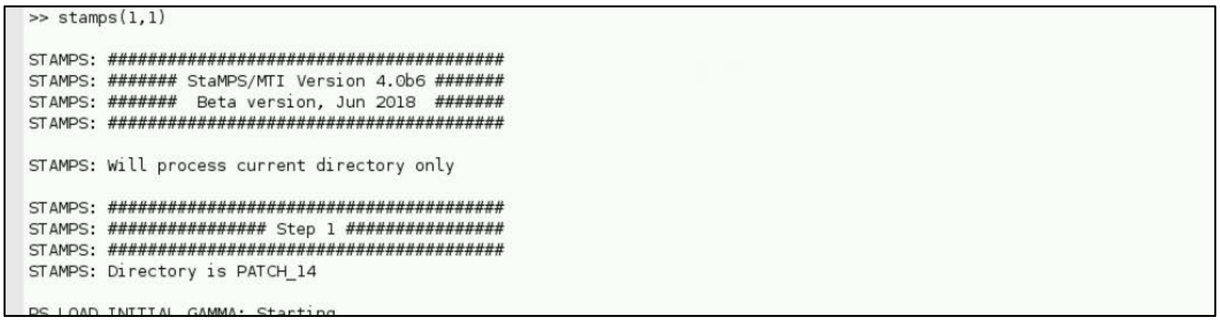
```
cd PATCH_14
```

Step 1 loads the data for the initial candidate PS pixels and stores them in MATLAB workspace files. To run the first step type:

```
stamps(1,1)
```

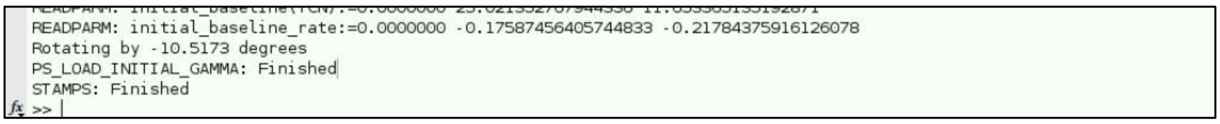
The format of the command is *stamps(START_STEP,END_STEP*), this makes it possible to run multiple steps at once. To start we will however run them separately.

The command will create .mat files with "1" in the name, indicating that they contain the initial PS data.



```
>> stamps(1,1)

STAMPS: ########################################
STAMPS: ####### StaMPS/MTI Version 4.0b6 #######
STAMPS: #######  Beta version, Jun 2018  #######
STAMPS: ########################################

STAMPS: Will process current directory only

STAMPS: ########################################
STAMPS: ################ Step 1 ################
STAMPS: ########################################
STAMPS: Directory is PATCH_14

PS_LOAD_INITIAL_GAMMA: Starting
```

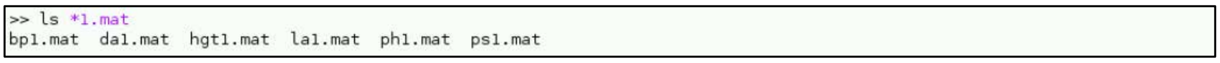
…



```
READPARM: initial_baseline_rate:=0.0000000 -0.17587456405744833 -0.21784375916126078
Rotating by -10.5173 degrees
PS_LOAD_INITIAL_GAMMA: Finished
STAMPS: Finished
>> |
```

The command will create .mat files with "1" in the name, indicating that they contain the initial PS data. To list the newly created files, run:

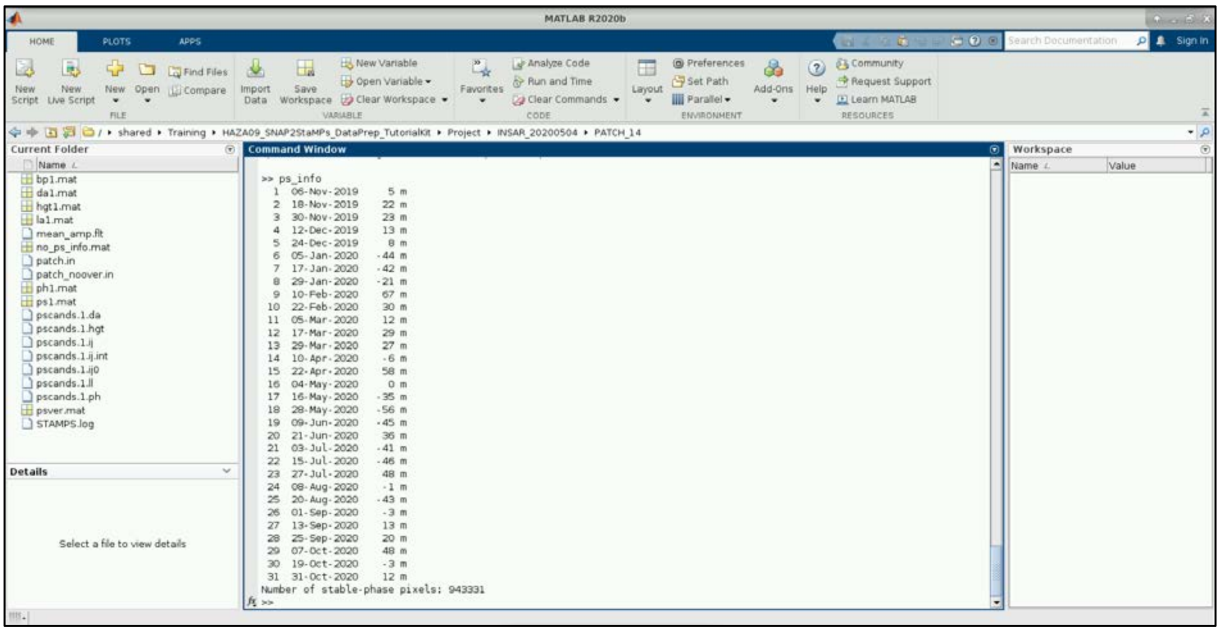
```
ls *.1.mat
```



```
>> ls *1.mat
bp1.mat  da1.mat  hgt1.mat  la1.mat  ph1.mat  ps1.mat
```

Now we can list details for each interferogram (only part contained in PATCH_14). Run the following command:

```
ps_info
```



```
>> ps_info
  1  06-Nov-2019    5 m
  2  18-Nov-2019   22 m
  3  30-Nov-2019   23 m
  4  12-Dec-2019   13 m
  5  24-Dec-2019    8 m
  6  05-Jan-2020  -44 m
  7  17-Jan-2020  -42 m
  8  29-Jan-2020  -21 m
  9  10-Feb-2020   67 m
 10  22-Feb-2020   30 m
 11  05-Mar-2020   12 m
 12  17-Mar-2020   29 m
 13  29-Mar-2020   27 m
 14  10-Apr-2020   -6 m
 15  22-Apr-2020   58 m
 16  04-May-2020    0 m
 17  16-May-2020  -35 m
 18  28-May-2020  -56 m
 19  09-Jun-2020  -45 m
 20  21-Jun-2020   36 m
 21  03-Jul-2020  -41 m
 22  15-Jul-2020  -46 m
 23  27-Jul-2020   48 m
 24  08-Aug-2020   -1 m
 25  20-Aug-2020  -43 m
 26  01-Sep-2020   -3 m
 27  13-Sep-2020   13 m
 28  25-Sep-2020   20 m
 29  07-Oct-2020   48 m
 30  19-Oct-2020   -3 m
 31  31-Oct-2020   12 m
Number of stable-phase pixels: 943331
>>
```

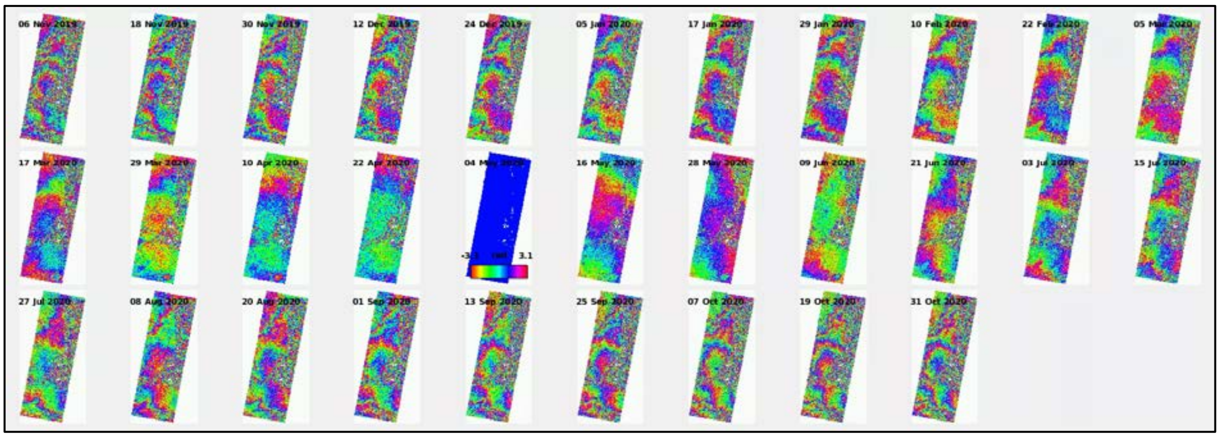
We can now visualize the wrapped phase of the selected initial PS candidate pixels using the ps_plot command, run:

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

TIP: To check the functionality by running:

```
help ps_plot
```

ps_plot(VALUE_TYPE,BACKGROUND,PHASE_LIMS,REF_IFG,IFG_LIST,...)



The blue image corresponds to an interferogram using the master image as both master and slave and so it has a zero-phase difference. We cannot see many details in this view, lets then visualize a single interferogram. We can choose the interferogram we want to visualize. Let's choose the second ifg (slave: 18 November 2019) with the mean amplitude of all the images as background (value 5).

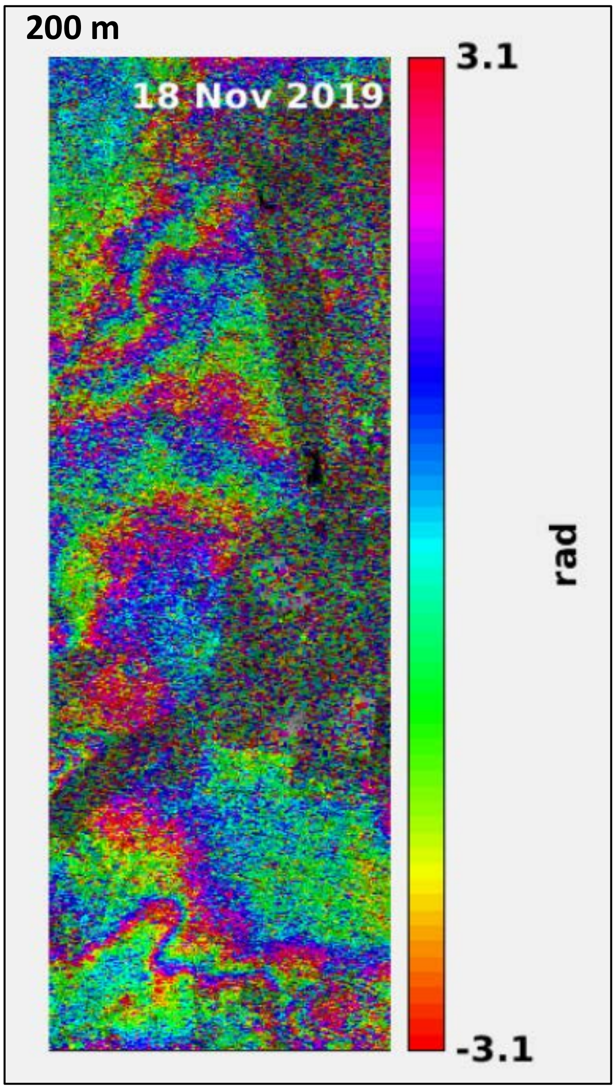
```
ps_plot('w',5,0,0,2)
```

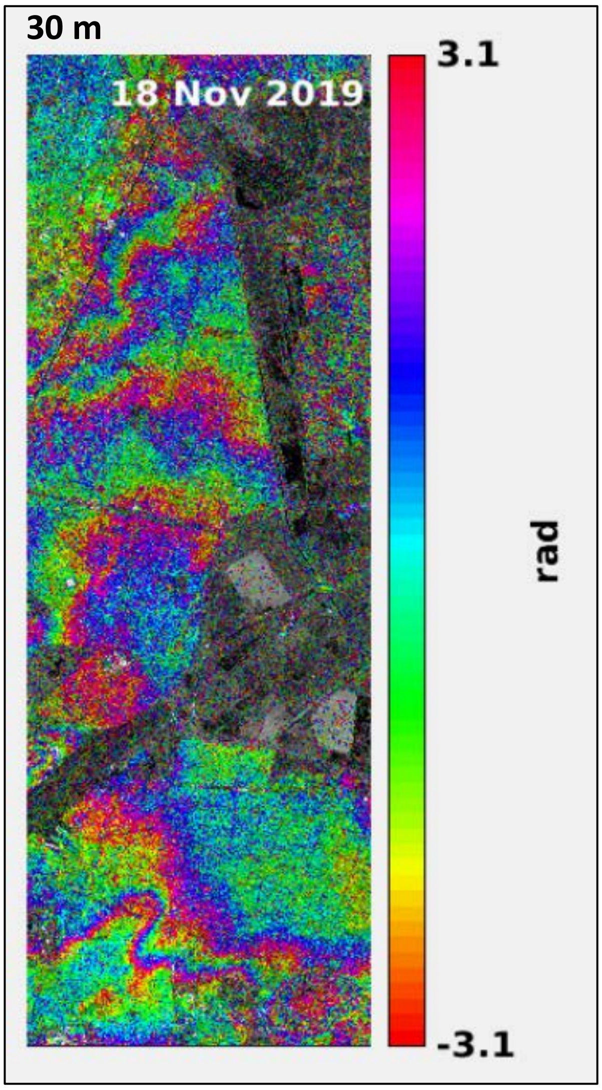
By default, the size of the plotted points is set to 120m, this means that the points overlap and merge into one another. We can change the plotted size by changing the 'plot_scatterer_size' parameter. Run:

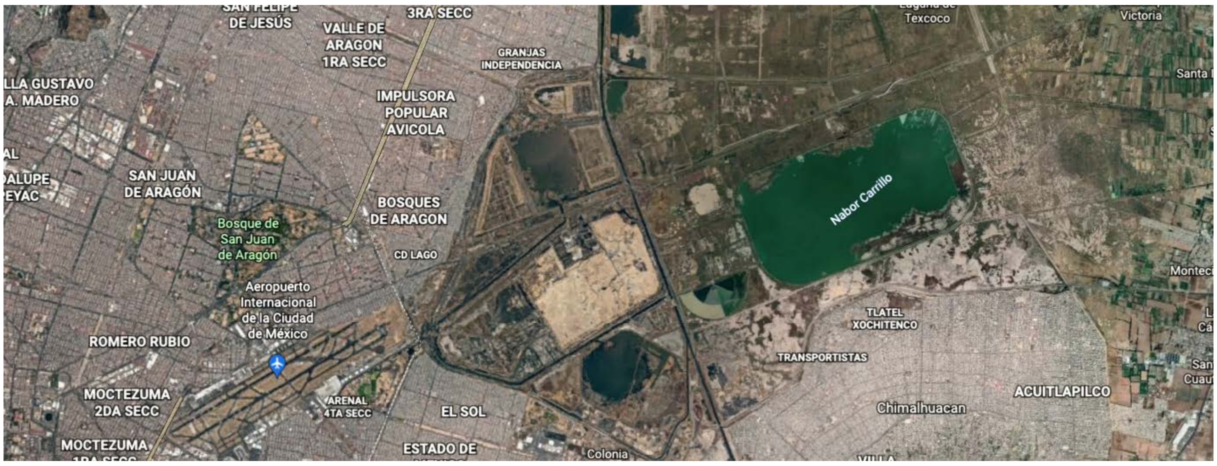
```
setparm('plot_scatterer_size',30)
```

Now run the command to visualize the second interferogram again. If you place the plot with 200m point size and the new one with 30 m size, we can see that the right side of the image contains much less candidate pixels.

You can also notice two grey rectangles in this area, these correspond to bodies of water and should not therefore contain any candidate pixels. This demonstrates that amplitude dispersion is not a perfect proxy for phase noise.







7.3 STEP 2: Calculate temporal coherence

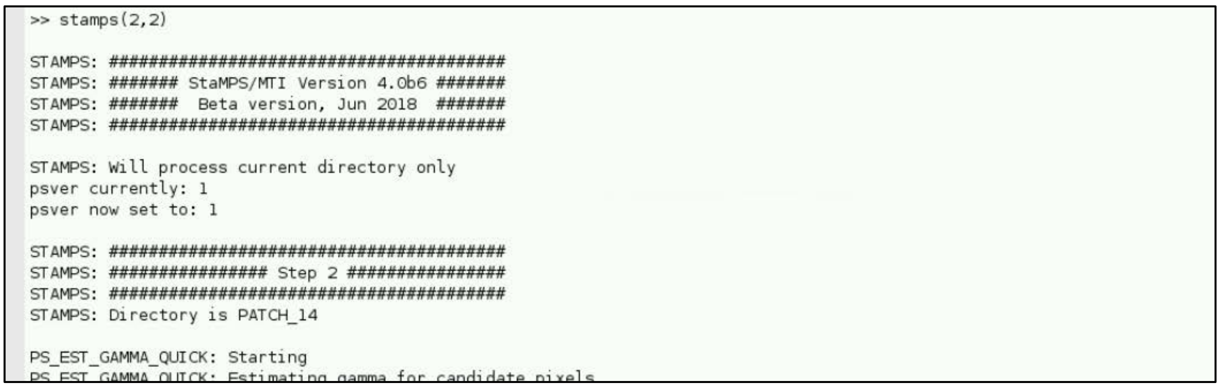
This is an iterative step that estimates the phase noise value (γ) for each candidate pixel in every interferogram (StaMPS manual). Processing is controlled by the following parameters:

Parameter	Default	Description	Influence
max_topo_err	20 m	Maximum uncorrelated DEM error (in m). Pixels with uncorrelated DEM error greater than this will not be picked. Setting this higher, will increases the mean γ value (coherence-like measure) of pixels that have random phase. Typical values are around 10 m.	unknown
filter_grid_size	50 m	Pixel size of grid (in m). Candidate pixels are resampled to a grid with this spacing before filtering to determine the spatially correlated phase. For the application of this filter technique, the irregular grid of the provisionally selected points must be transformed to a regular grid, whereby the grid width must be sufficiently small, so that over an element of this grid no strong variation in the phase takes place (typical values would be 40 to 100 m). The filter works like the Goldstein filter on individual sections of the image (windowed Fourier transformation).	low
filter_weighting	'P-square'	Weighting scheme (PS probability squared), the other possibility being 'SNR'. Candidate pixels are weighted during resampling according to this scheme. P-squared was found to be better (Hooper et al., 2007).	high
clap_win	32	CLAP (Combined Low-pass and Adaptive Phase) filter (2-D FFT) window size (pixel x pixel), typically 32x32 or 64x64. Depending on over what distance pixels are expected to remain spatially correlated. Together with filter grid size, determines the area included in the spatially correlated phase estimation.	unknown
clap_low_pass_w avelength	800	CLAP filter low-pass contribution cut-off spatial wavelength (in m). Wavelengths longer than this are passed.	medium
clap_alpha	1	Weighting parameter for phase filtering (CLAP α\alphaα term). Together with the β term, determines the relative contribution of the low-pass and adaptive phase elements to the CLAP filter.	unknown
clap_beta	0.3	Weighting parameter for phase filtering (CLAP β term).	
gamma_change_c onvergence	0.005	Threshold for change in change in mean value of γ (coherence-like measure). Determines when convergence is reached and iteration ceases. *γ is a measure of the phase noise level and an indicator of whether the pixel is a PS pixel (i.e., a measure of the variation of residual phase for a pixel). Once we have converged on estimates for the phase stability of each pixel, we select those most likely to be PS pixels, with a threshold determined by the fraction of false positives we deem acceptable. We also seek to reject pixels that persist only in a subset of the interferograms and those that are dominated by scatterers in adjacent PS pixels. After every iteration we calculate the root-mean-square change in* γ_x*. When it ceases to decrease, the solution has converged, and the algorithm stops iterating.*	low
gamma_max_iter ations	3	Maximum number of iterations before iteration process ceases. Phase Stability Estimation: Initially, we select a subset of pixels based on analysis of their amplitudes, rejecting those least likely to be PS pixels. We then estimate the phase stability of each of these pixels through phase analysis, which we successively refine in a series of iterations.	unknown
quick_est_gamma _flag	'y'	Employ quick estimation procedure for γ value.	unknown

Table adapted from StaMPS/2-4_StaMPS-steps.md · master · Matthias Schlögl / gis-blog · GitLab & StaMPS Manual

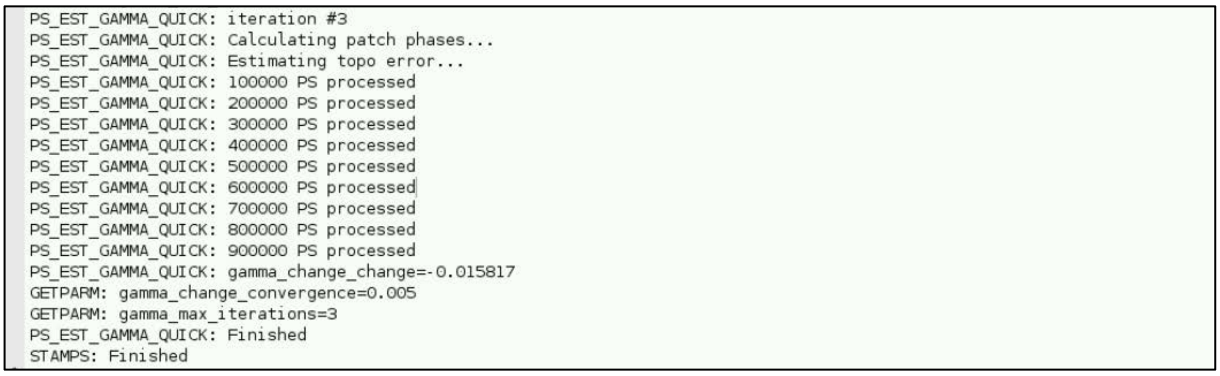
We will keep all parameters as default and run:

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



```
>> stamps(2,2)

STAMPS: ########################################
STAMPS: ####### StaMPS/MTI Version 4.0b6 #######
STAMPS: #######  Beta version, Jun 2018  #######
STAMPS: ########################################

STAMPS: Will process current directory only
psver currently: 1
psver now set to: 1

STAMPS: ########################################
STAMPS: ################ Step 2 ################
STAMPS: ########################################
STAMPS: Directory is PATCH_14

PS_EST_GAMMA_QUICK: Starting
PS_EST_GAMMA_QUICK: Estimating gamma for candidate pixels
```

…



```
PS_EST_GAMMA_QUICK: iteration #3
PS_EST_GAMMA_QUICK: Calculating patch phases...
PS_EST_GAMMA_QUICK: Estimating topo error...
PS_EST_GAMMA_QUICK: 100000 PS processed
PS_EST_GAMMA_QUICK: 200000 PS processed
PS_EST_GAMMA_QUICK: 300000 PS processed
PS_EST_GAMMA_QUICK: 400000 PS processed
PS_EST_GAMMA_QUICK: 500000 PS processed
PS_EST_GAMMA_QUICK: 600000 PS processed
PS_EST_GAMMA_QUICK: 700000 PS processed
PS_EST_GAMMA_QUICK: 800000 PS processed
PS_EST_GAMMA_QUICK: 900000 PS processed
PS_EST_GAMMA_QUICK: gamma_change_change=-0.015817
GETPARM: gamma_change_convergence=0.005
GETPARM: gamma_max_iterations=3
PS_EST_GAMMA_QUICK: Finished
STAMPS: Finished
```

7.4 STEP 3: Initial PS selection

This step makes a selection of PS based on probability, by comparison to results for data with random phase. This is usually done twice. After the first selection, the temporal coherence of each selected pixel is re-estimated more accurately, by dropping the pixel itself from the estimation of the spatially correlated phase. Then the selection process is repeated. (A. Hooper, 2018)

TIP: This step is extremely time demanding due to the gamma re-estimation process. You can disable the re-estimation by changing the corresponding parameter.

```
setparm('select_reest_gamma_flag’, ’n’)
```

The parameters influencing this step are shown below (some parameters set in step 2 are also relevant):

Parameter	Default	Description	Influence
select_method	'DENSITY'	Selection method for pixels with random phase. Other option 'PERCENT'. We select those pixels most likely to be PS pixels, with a threshold determined by the fraction of false positives we deem acceptable. We select PS based on the calculated values of γ_x (measure of the phase stability of the pixel).	unknown
density_rand	20	Maximum acceptable spatial density (per km^2) of selected pixels with random phase. Only applies if select method is 'DENSITY'. At this stage we can usually accept a high density, as most random-phase pixels will be dropped in the next step.	unknown
percent_rand	20	Maximum acceptable percentage of selected pixels having random phase. They are selected on the basis of their noise characteristics. The value can be set higher, since bad pixels will be weeded in the next steps (4 and 5). Only applies if select_method is 'PERCENT'.	medium

Table adapted from StaMPS/2-4_StaMPS-steps.md · master · Matthias Schlögl / gis-blog · GitLab & StaMPS Manual

We will keep all parameters as default and run:

```
stamps(3,3)
```

7.5 STEP 4: PS weeding

Pixels selected in the previous step are weeded, dropping those that are due to signal contribution from neighbouring ground resolution elements and those deemed too noisy. Data for the selected pixels are stored in new workspaces. (*StaMPS Manual*)

Processing is controlled by the following parameters:

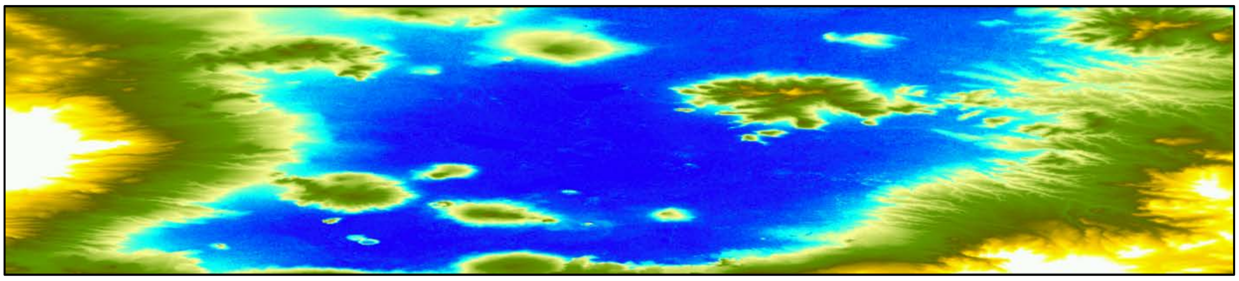
Parameter	Default	Description	Influence
weed_standard_dev	1.0	Threshold standard deviation. For each pixel, the phase noise standard deviation for all pixel pairs including the pixel is calculated. If the minimum standard deviation is greater than the threshold, the pixel is dropped. If set to 10, no noise-based weeding is performed. A pixel may display a period of good phase stability, preceded and/or followed by a period of low stability. To avoid picking these pixels, we include an optional step to estimate the variance of x for each pixel using the bootstrap percentile method with 1000 iterations and drop pixels with standard deviation over a defined threshold, good initial values are 1-1.2.	high: increasing the SD increases the selected PS but also the noise level.
weed_max_noise	Inf	Threshold for the maximum noise allowed for a pixel. For each pixel the minimum pixel-pair noise is estimated per interferogram. Pixels with maximum interferogram noise value higher than the indicated threshold are dropped.	unknown
weed_time_win	730	Smoothing window (in days) for estimating phase noise distribution for each pair of neighbouring pixels. The time series phase for each pair is smoothed using a Gaussian-weighted piecewise linear fit. weed time win specifies the standard deviation of the Gaussian. The original phase minus the smoothed phase is assumed to be noise.	unknown
weed_neighbours	‘n’	Flag for proximity weeding. If set to 'y', pixels are dropped based in their proximity to each other. A Scatterer that is bright can dominate pixels other than the pixel corresponding to its physical location. The error in look angle and squint angle due to the offset of the pixel from the physical location usually results in these pixels not being selected as PS. However, the slight oversampling of the resolution cells can cause pixels immediately adjacent to the PS pixel to be dominated by the same scatterer where the error may be sufficiently small that the pixel appears stable. To avoid picking these pixels, we assume that adjacent pixels selected as PS are due to the same dominant scatterer. As we expect the pixel that corresponds to the physical location to have the highest SNR, for groups of adjacent stable pixels we select as the PS only the pixel with the highest value of γ_x.	high: should be set to 'y'
weed_zero_elevation	‘n’	This parameter should be applied if there are no elevation areas in the study area – e.g. ocean or other waterbodies (depending on the DEM used)	unknown

Table adapted from StaMPS/2-4_StaMPS-steps.md · master · Matthias Schlögl / gis-blog · GitLab & StaMPS Manual

In this step we will turn on the proximity weeding parameter.

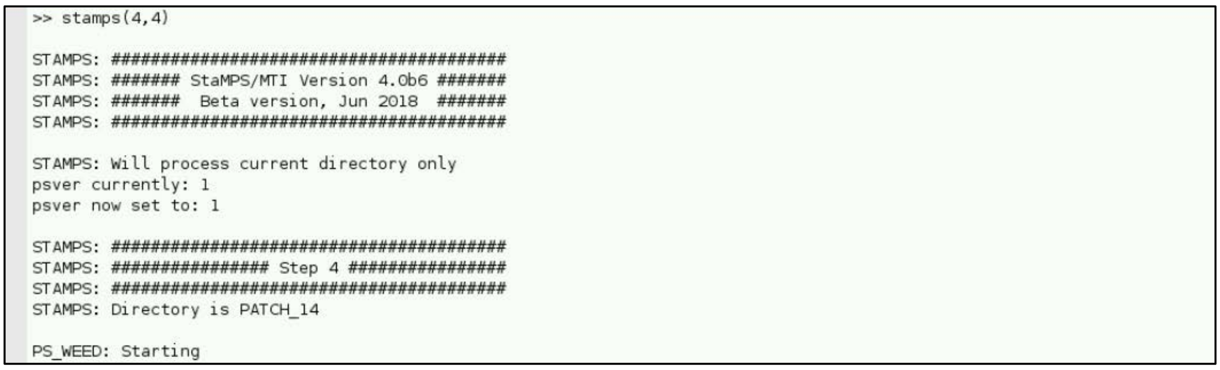
```
setparm('weed_neighbours','y')
```

We can keep the ‘weed_zero_elevation’ parameter as default since there is no zero elevation areas in our data (altitudes between 2200 and 4000 m).



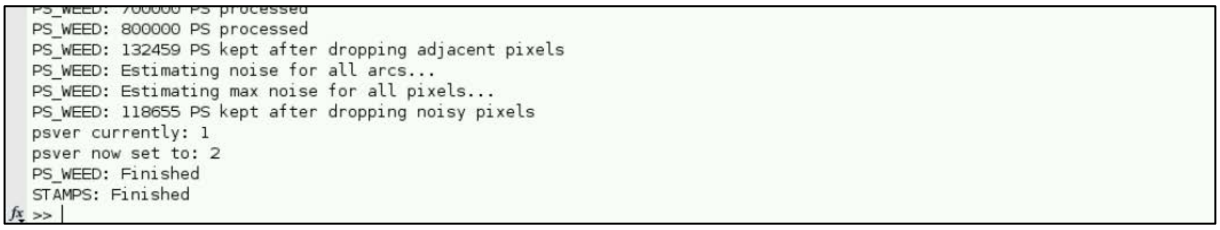
We will keep other parameters as default and run:

```
stamps(4,4)
```

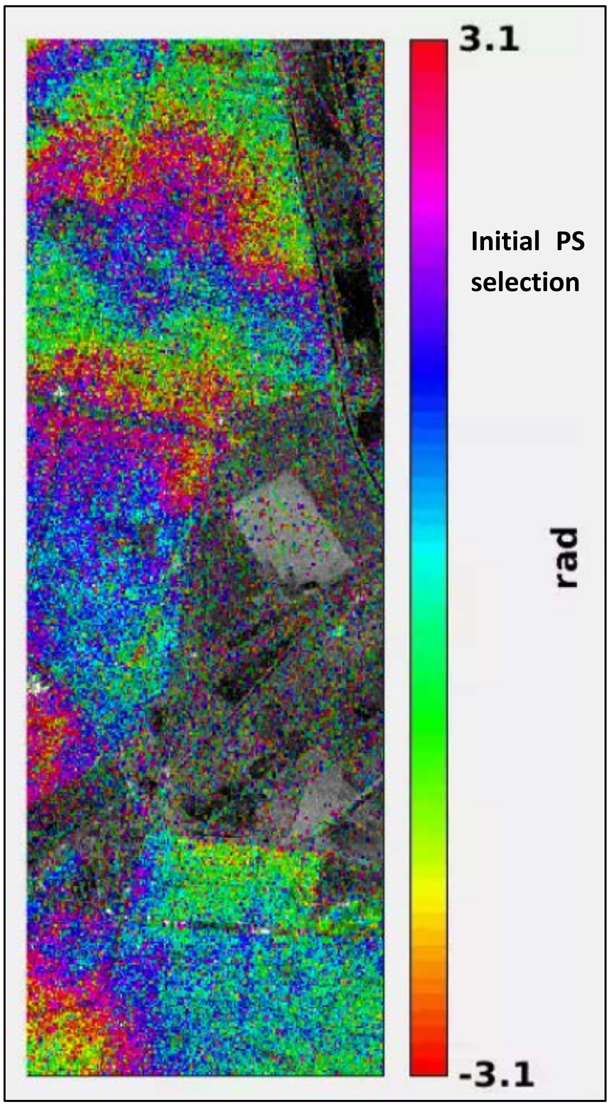


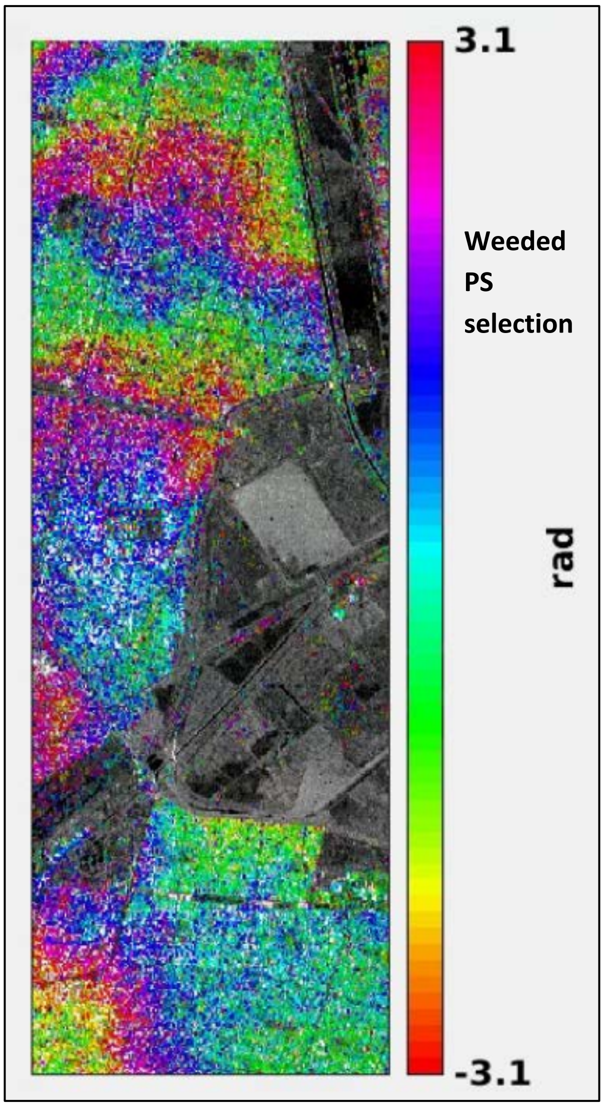
```
>> stamps(4,4)

STAMPS: ########################################
STAMPS: ####### StaMPS/MTI Version 4.0b6 #######
STAMPS: #######  Beta version, Jun 2018  #######
STAMPS: ########################################

STAMPS: Will process current directory only
psver currently: 1
psver now set to: 1

STAMPS: ########################################
STAMPS: ################ Step 4 ################
STAMPS: ########################################
STAMPS: Directory is PATCH_14

PS_WEED: Starting
```

…



```
PS_WEED: 700000 PS processed
PS_WEED: 800000 PS processed
PS_WEED: 132459 PS kept after dropping adjacent pixels
PS_WEED: Estimating noise for all arcs...
PS_WEED: Estimating max noise for all pixels...
PS_WEED: 118655 PS kept after dropping noisy pixels
psver currently: 1
psver now set to: 2
PS_WEED: Finished
STAMPS: Finished
>> |
```





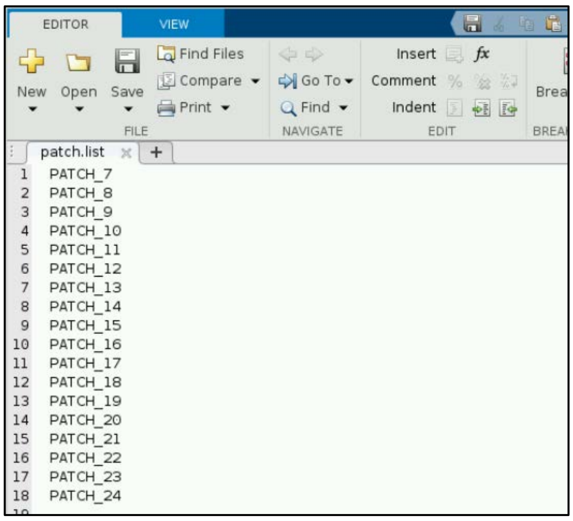
You can test setting the *weed_standard_dev* to 1.2 for example and re-run step 4 to see how the number of selected points will change. Make sure however to set the final run to the value that you wish to use for the other patches as well.

7.6 STEPS 1-4 for other patches

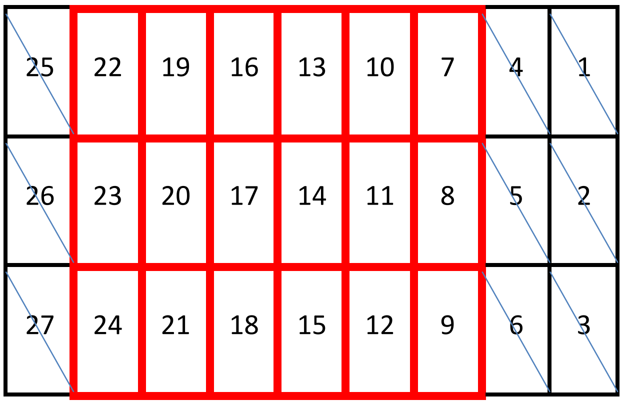
At this point we need to process also the other patches since they will be merged at the end of step 5. First, we need to change the directory to ***INSAR_20200504***. To do this run:

```
cd ..
```

We do not need to process all of the patches – patch 14 is already processed and we will also disregard patches 1 – 6 and 25 – 27 to decrease the processing time. To do this, scroll down in the Current folder menu on the left and double click on the ***patch.list*** file. It will open in MATLAB Editor. Now delete the lines containing patches 1 – 6, 14 and 25 – 27 and click **Save**. Then close the **Editor** window.



```
PATCH_7
PATCH_8
PATCH_9
PATCH_10
PATCH_11
PATCH_12
PATCH_13
PATCH_14
PATCH_15
PATCH_16
PATCH_17
PATCH_18
PATCH_19
PATCH_20
PATCH_21
PATCH_22
PATCH_23
PATCH_24
```



The patches are organized in columns, for the descending pass the patch number 1 is located at the upper right corner.

The process will only consider patches listed in this file. Now, to process them, run:

```
stamps(1,4)
```

NOTE: This step may run for many hours depending on your machine and whether the 'select_reest_gamma_flag' is enabled or no.

7.7 STEP 5: Phase correction

Now when we have all the required patches processed, we can run the phase correction step. The wrapped phase of the selected pixels is corrected for spatially uncorrelated look angle (DEM) error. At the end of this step the patches are merged.

Processing is controlled by the following parameters:

Parameter	Default	Description	Influence
merge_resample_size	0	Coarser posting (in m) to resample to. If set to 0, no resampling is applied. Use in case of memory issues to apply coarser sampling.	unknown
merge_standard_dev	Inf	Threshold standard deviation. For each resampled pixel, the phase noise standard deviation is computed. If the standard deviation is greater than the threshold, the resampled pixel is dropped. Only applied in case of resampling.	unknown

Table adapted from StaMPS/2-4_StaMPS-steps.md · master · Matthias Schlögl / gis-blog · GitLab & StaMPS Manual

We will keep all parameters as default and run:

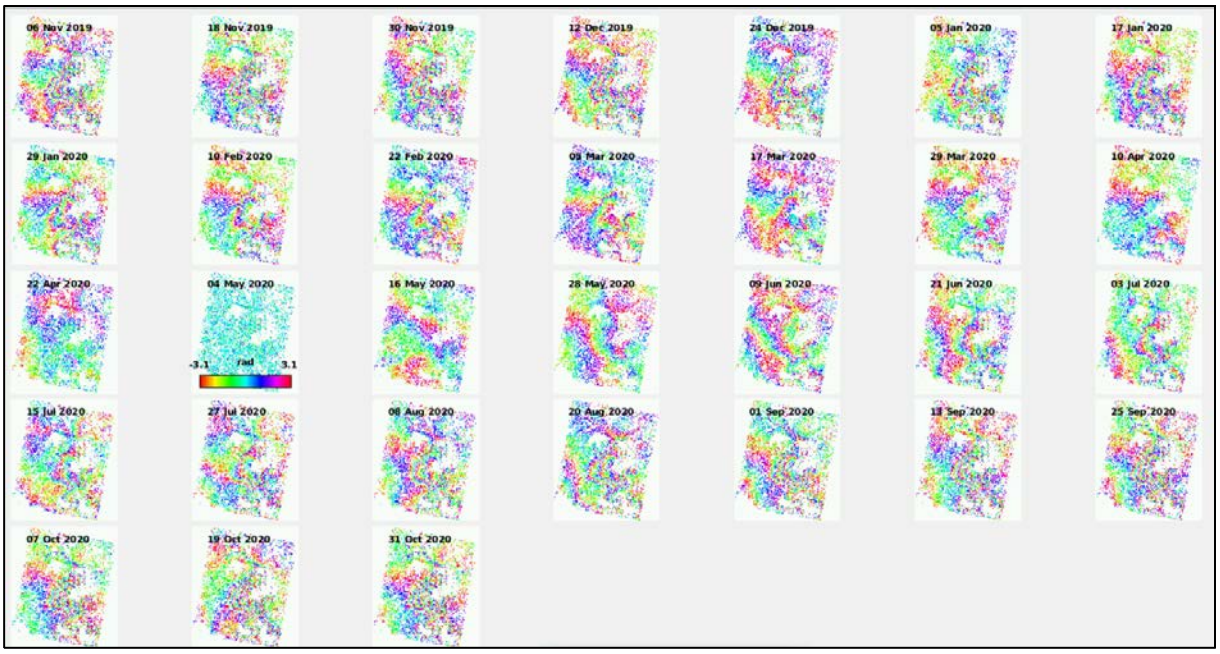
```
stamps(5,5)
```

The corrected phase information will be stored as ‘version 2’ (PS candidates were version 1). We can list the files by using the following command:

```
ls *2.mat
```

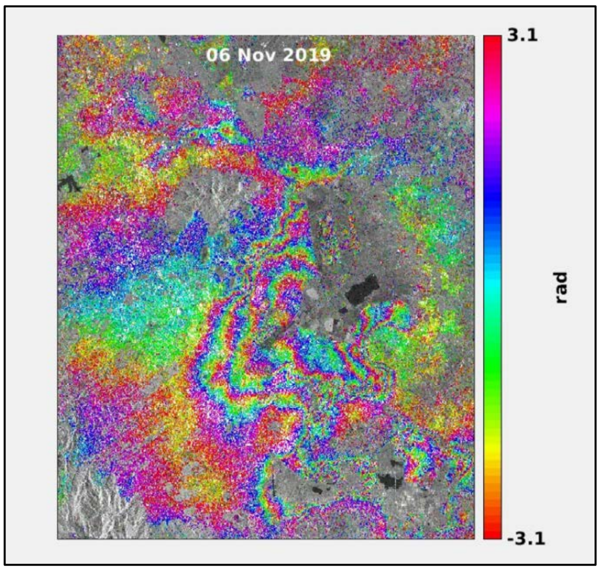
Now it is possible to plot the phase of the selected pixels for the full study area:

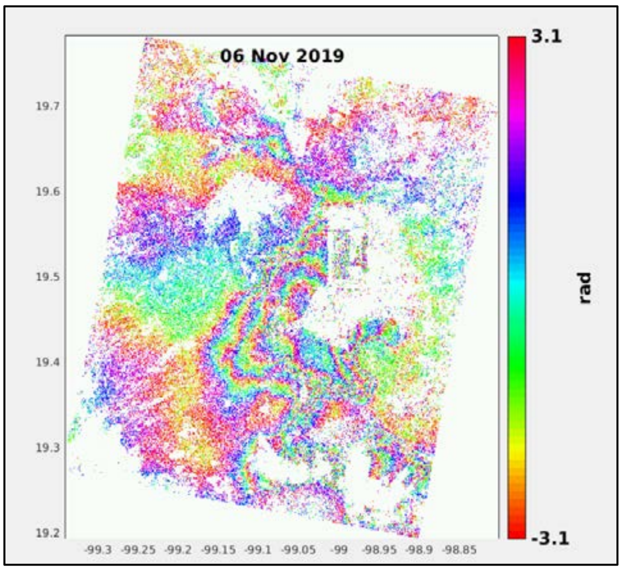
```
ps_plot(‘w’)
```



We can check the coherence by visually scanning the phase of each interferogram for this we need to plot the interferograms one by one. You can do this by running the command shown as bellow and changing the last number which represents the number of the interferogram in the time series.

```
ps_plot(‘w’,5,0,0,1)
```





It is possible to also change the background of the interferogram. So far, we have used value “5” which corresponds to the mean amplitude, other possible values are “0”- black, 1 – “white” (using the “0” or “1” will also result the image to be plotted in lat/lon instead of radar geometry “5”).

7.8 STEP 6: Phase unwrapping

When we measure phase, we can only measure it in the range 0 to 2π, when you consider the relationship of phase and wavelength you can imagine that there will be a lot of 2π cycles before the signal arrives to the ground as the wavelength, we use here is 5.5 cm for Senitnel-1. Therefore, the difference or interferometric phase at any point is the value between 0 to 2π plus any unknown number k of cycles.

The phase ambiguity is solved during the so-called phase unwrapping calculation which is the process of reconstructing the original phase shift from this "wrapped" representation. It consists of adding or subtracting multiples of 2 in the appropriate places to make the phase image as smooth as possible.

In this step the stamps will run the unwrapping using the snaphu SW that we have set up in the beginning.

Processing is controlled by the following parameters:

Parameter	Default	Description	Influence
unwrap_method	‘3D’	Unwrapping method, alternative '3D_QUICK' for SBAS. '3D_QUICK' is default for SBAS. Can be set to '3D' to potentially improve accuracy, although this will take longer. PS data sets are in fact three-dimensional, the third dimension being that of time, and unwrapping accuracy is improved by treating the problem as 3-D.	medium/ high
unwrap_grid_size	200	Resampling grid spacing. If unwrap_prefilter_flag is set to 'y', phase is resampled to this grid. Should be at least as large as merge_grid_size. High values reduce noise but may lead to undersampling of the signal	high
unwrap_gold_alpha	0.8	Value of α for Goldstein filter. The larger the value the stronger the filtering.	medium/ high
unwrap_gold_n_win	32	Window size for Goldstein filter. Optionally, if the data are very noisy, a pre-filtering step can be run to filter the data spatially before unwrapping, as is common in 2-D (Goldstein and Werner, 1998). For each time step, the complex phase data are first sampled to a grid using a grid spacing over which little variation in phase is expected (typically 40 to 100 m). Where multiple pixels fall in the same grid cell, the complex phase is summed. The gridded phase is then transformed using the fast Fourier transform and filtered in the frequency domain using an adaptive phase filter [Goldstein and Werner, 1998]. This preserves the dominant frequencies which are present in the data. After inverse transformation, the grid cells containing data are treated as the new data points for input into the unwrapping algorithm.	unknown
unwrap_hold_good_values	‘n’	-	-
unwrap_prefilter_flag	‘y’	Prefilter phase before unwrapping to reduce noise. SCLA and AOE subtraction is applied when step 6 is redone after step 7. To avoid subtraction, use scla_reset. Setting to 'y' is recommended.	high
unwrap_patch_phase	‘n’	Use the patch phase from Step 3 as prefiltered phase. If set to 'n' PS phase is filtered using a Goldstein adaptive phase filter. Setting to 'n' is recommended.	high
unwrap_time_win	730	Smoothing window (in days) for estimating phase noise distribution for each pair of neighbouring pixels (i.e. smoothing filter length in days for smoothing phase in time by estimate the noise contribution for each phase arc). The time series phase for each pair is smoothed using a Gaussian window with standard deviation of this size. Original phase minus smoothed phase is assumed to be noise, which	high

		is used for determining probability of a phase jump between the pair in each interferogram. With a Gaussian filter, the low frequencies of the time series are determined and subtracted, which leads to the noise measurements whose variance is estimated.	
drop_ifg_index	[]	ifg will not be included in unwrapping. This might be helpful if after unwrapping and testing parameters an ifg is still noisy. After omitting interferograms, steps from 2 onwards should be redone.	potentially high (if needed)

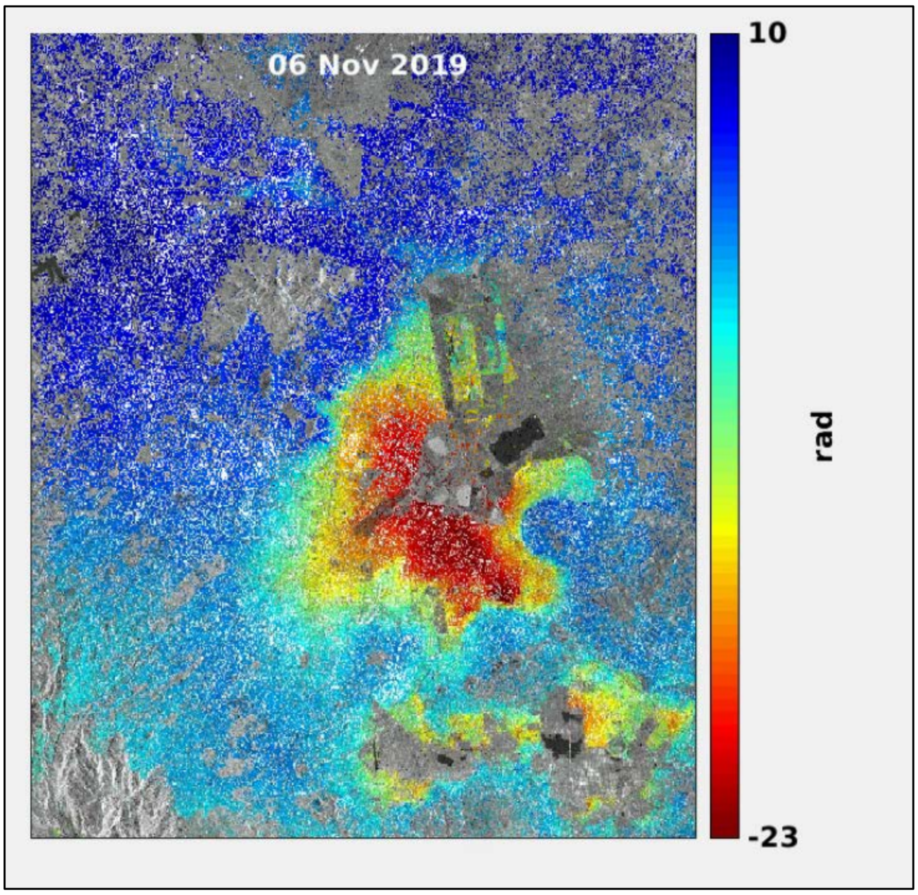
Table adapted from StaMPS/2-4_StaMPS-steps.md · master · Matthias Schlögl / gis-blog · GitLab & *StaMPS Manual*

We will keep all parameters as default and run:

```
stamps(6,6)
```

Once the processing finishes, we can visualize the one of the unwrapped interferograms by running:

```
ps_plot('u',5,0,0,1)
```



Here we have visualized only the first interferogram, but you can visualize all of them again using simply “ps_plot(‘u’)” command. This is our first unwrapping result. We will rerun this step later …

7.9 TRAIN: Linear tropospheric correction

A linear tropospheric correction can be computed based on the phase and topographic information. By default, the 'non_defo_flag' is set to 'n', which will force the computation of the tropospheric relation between phase and topography based on the whole interferogram.

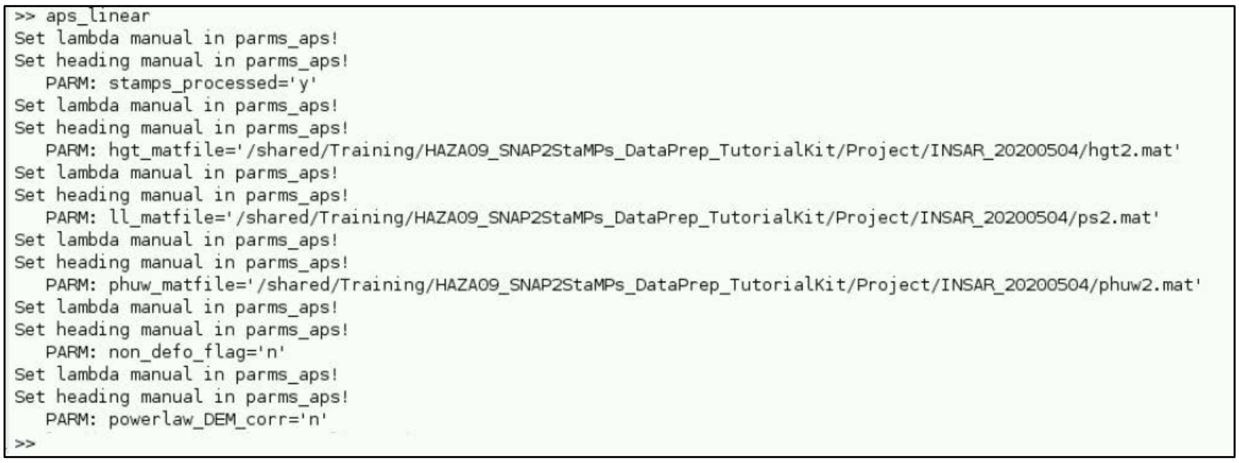
To compute the tropospheric correction type:

```
aps_linear
```

Output is saved into a “tca2.mat” or “tca_sb2.mat” as the “ph_tropo_linear” variable. The parameters which are used in aps_linear are:

Parameter	Default	Description	Influence
hgt_matfile	hgt2.mat	Full file path containing the topography, stored in a column vector of size [n_points x 1].	-
phuw_matfile	phuw.mat	Full file path of the unwrapped interferograms, stored as a matrix of size [n_points x n_ifgs].	-
ll_matfile	ps2.mat	Full file path of the geo-coordinates, stored in a matrix with size [n_points x 2] and in its columns the longitude and latitude.	-
non_defo_flag	‘n’	By default, the correction is performed based on the full interferogram. Put to 'y' to use the non-deforming estimation.	-

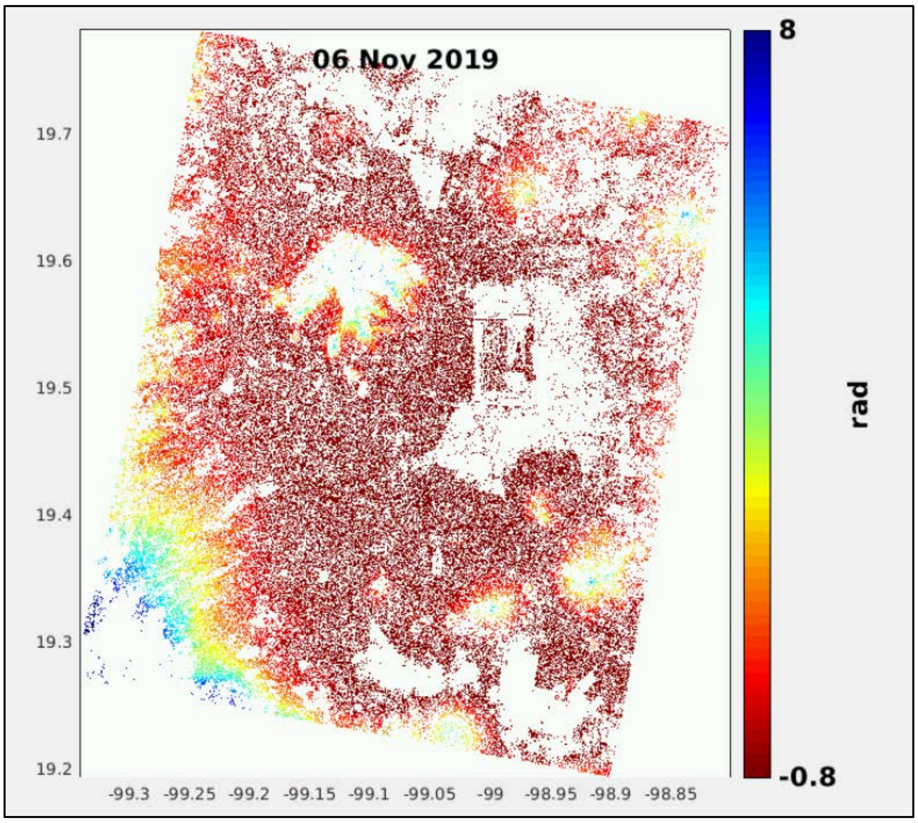
Table adapted from TRAIN Manual.

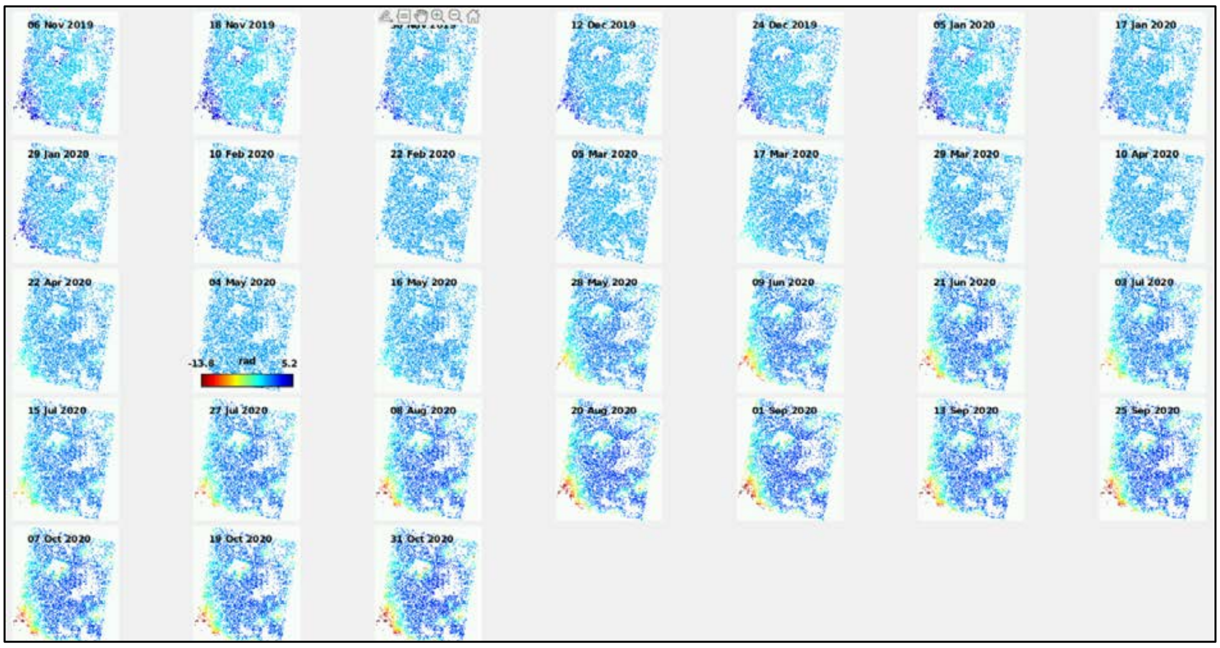


```
>> aps_linear
Set lambda manual in parms_aps!
Set heading manual in parms_aps!
   PARM: stamps_processed='y'
Set lambda manual in parms_aps!
Set heading manual in parms_aps!
   PARM: hgt_matfile='/shared/Training/HAZA09_SNAP2StaMPs_DataPrep_TutorialKit/Project/INSAR_20200504/hgt2.mat'
Set lambda manual in parms_aps!
Set heading manual in parms_aps!
   PARM: ll_matfile='/shared/Training/HAZA09_SNAP2StaMPs_DataPrep_TutorialKit/Project/INSAR_20200504/ps2.mat'
Set lambda manual in parms_aps!
Set heading manual in parms_aps!
   PARM: phuw_matfile='/shared/Training/HAZA09_SNAP2StaMPs_DataPrep_TutorialKit/Project/INSAR_20200504/phuw2.mat'
Set lambda manual in parms_aps!
Set heading manual in parms_aps!
   PARM: non_defo_flag='n'
Set lambda manual in parms_aps!
Set heading manual in parms_aps!
   PARM: powerlaw_DEM_corr='n'
>>
```

We can visualize the calculated tropospheric correction for any interferogram by running (for ifg 1):

```
ps_plot('a', 'a_linear', 1, 0, 0, 1)
```





7.10 Estimation of spatially correlated errors

Spatially uncorrelated look angle (SULA) error was calculated in Step 3 and removed in Step 5. In Step 7, spatially correlated look angle (SCLA) error is calculated which is due almost exclusively to spatially correlated DEM error (this includes error in the DEM itself, and incorrect mapping of the DEM into radar co-ordinates). Master atmosphere and orbit error (AOE) phase is also estimated simultaneously. (*StaMPS Manual*)

Processing is controlled by the following parameters:

Parameter	Default	Description	Influence
scla_deramp	'n'	If set to 'y', a phase ramp is estimated for each interferogram. The estimated ramp will be subtracted before unwrapping. This is useful if one is interested in local signals only.	medium/ unknown
drop_ifg_index	[]	Specified ifg will not be included in unwrapping. This might be helpful if after unwrapping and testing parameters an ifg is still noisy. Then the index can be set and steps from 2 should be redone.	high if needed

Table adapted from StaMPS/2-4_StaMPS-steps.md · master · Matthias Schlögl / gis-blog · GitLab & StaMPS Manual

If we want to estimate the phase ramp, we can change the parameter and then run the step 7:

```
setparm('scla_deramp','y')
stamps(7,7)
```

Then we can visualize:

1) the estimated the total DEM (look angle) error as radians per m of baseline:

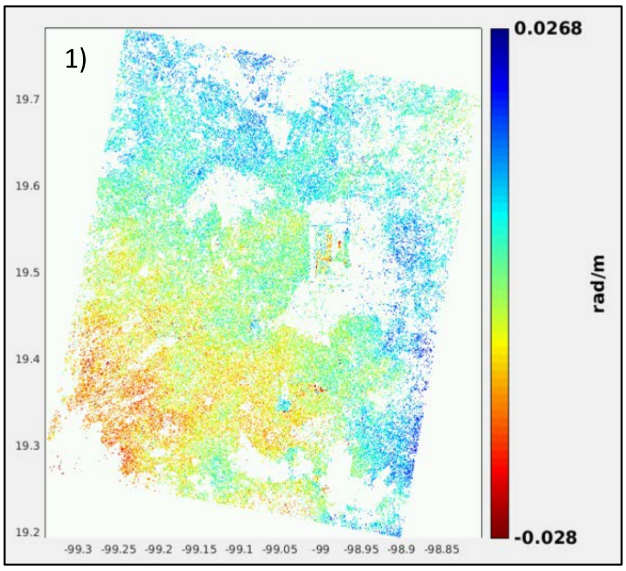
```
ps_plot('d')
```

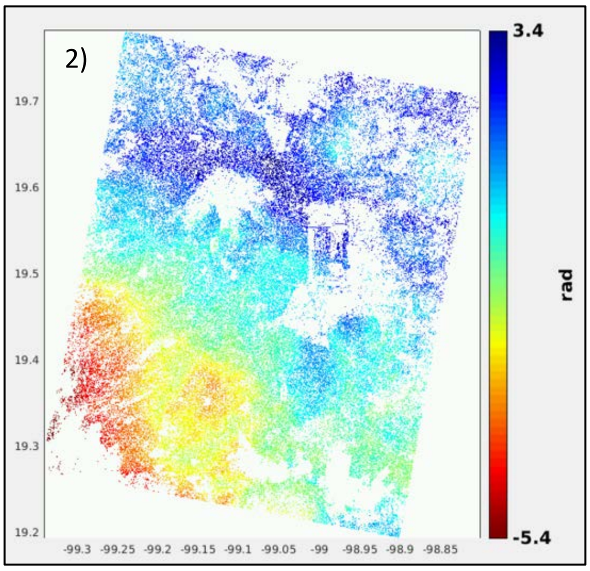
2) the master atmosphere:

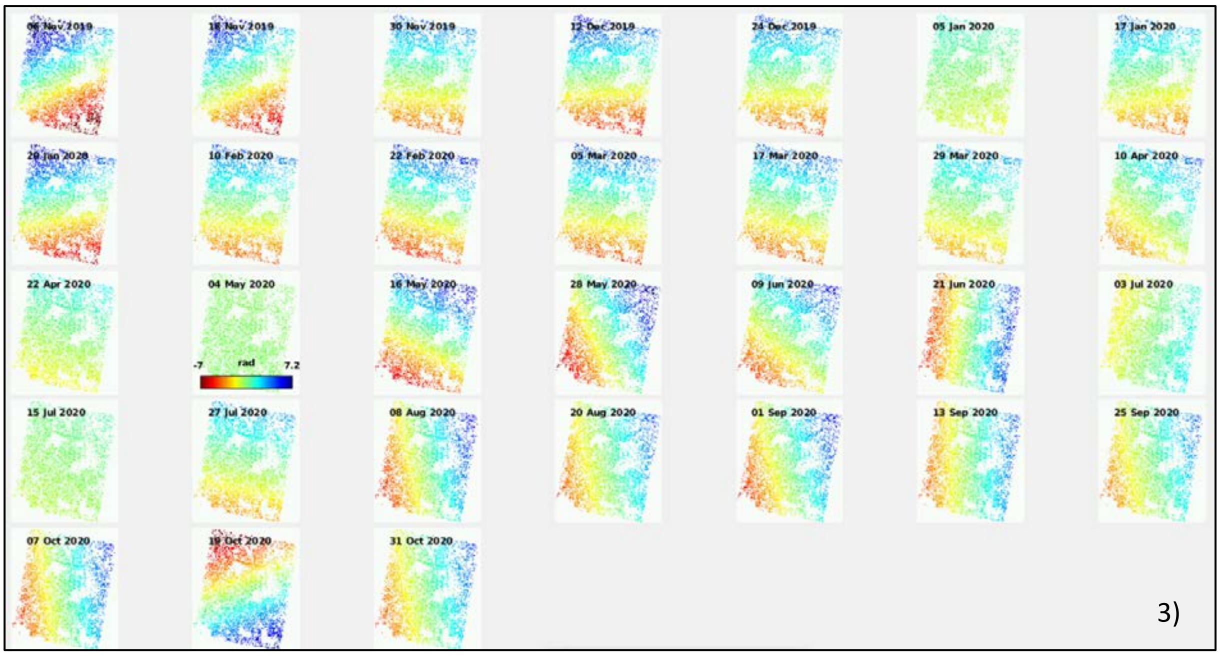
```
ps_plot('m')
```

3) estimated ramps

```
ps_plot('o')
```





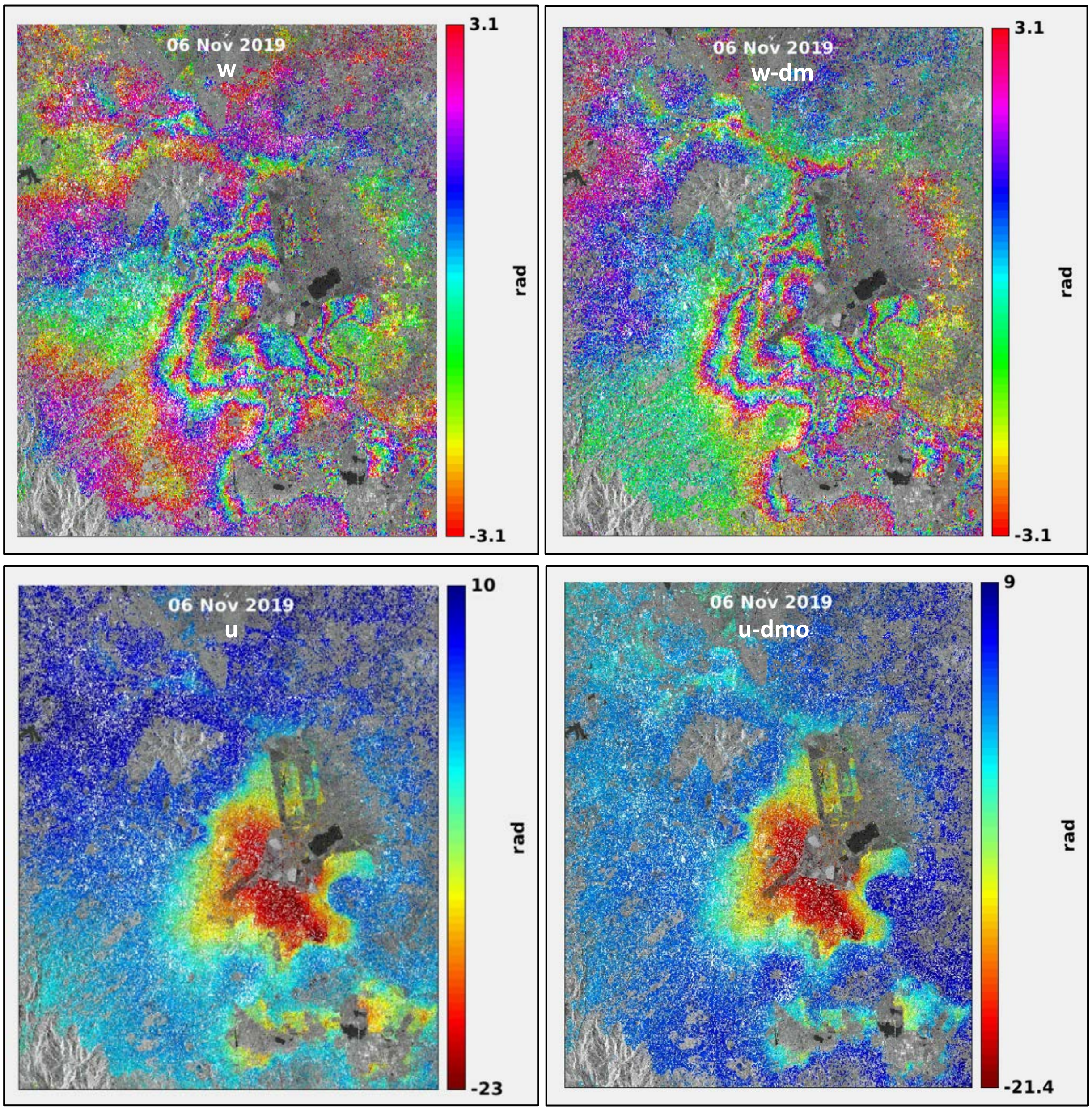


Now we can check for unwrapping errors (inconsistent jumps in phase). Unwrapping errors are more likely to occur in interferograms with longer perpendicular baselines due to the fact that there is more noise associated with each PS, and the phase due to any spatially correlated DEM error is larger, as it is proportional to perpendicular baseline (A.Hooper, 2018). To check the baselines, use the command:

```
ps_info
```

You can also see the estimated noise for each interferogram, which is calculated from the phase that is not correlated spatially and not correlated with baseline. Anything above approx. 80 degrees is likely to give trouble for unwrapping. In this case we are using Sentinel-1 data and the baselines are all small. (A.Hooper, 2018)

The plot for 'u-dm' should also be smoother than 'u'. Note that the default colour scales will be different and if smoother the phase range should be smaller. If not generally smoother, one or more interferograms have probably unwrapped incorrectly. It is also worth comparing the 'u-dm' to the wrapped phase corrected for 'd' and 'm'. Images below correspond to the first interferogram (highest estimated noise - 42.316 deg) with mean amplitude background e.g. `ps_plot(‘w-dm’, 5,0,0,1)`.



7.11 Re-run STEPS 6 and 7

In STEP 7 we have estimated spatially correlated errors and with TRAIN we have calculated the linear tropospheric correction. Removing these errors from the wrapped phase can aid the unwrapping step, for this reason we will now re-run STEPS 6 and 7. The estimates from STEP 7 are automatically subtracted from the wrapped phase before unwrapping.

```
stamps(6,6)
stamps(7,7)
```

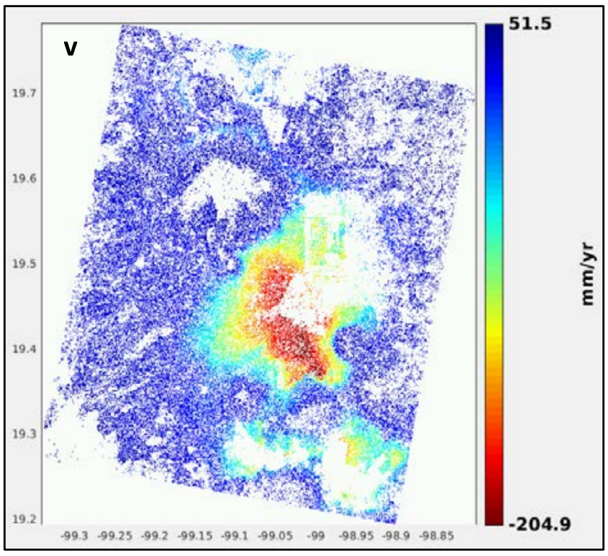
7.12 Velocity estimation, standard deviation, and time series

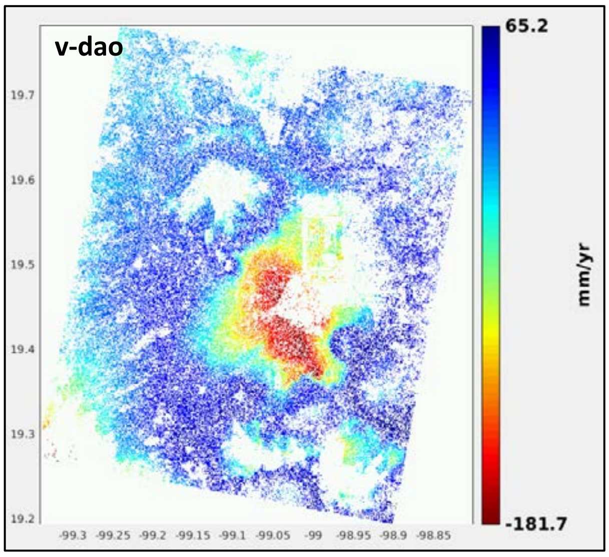
After running STEP 7 we can plot the estimated mean velocity:

```
ps_plot(‘v')
```

We can further subtract the DEM error, the orbital ramps and the tropospheric correction:

```
ps_plot('v-dao', 'a_linear')
```

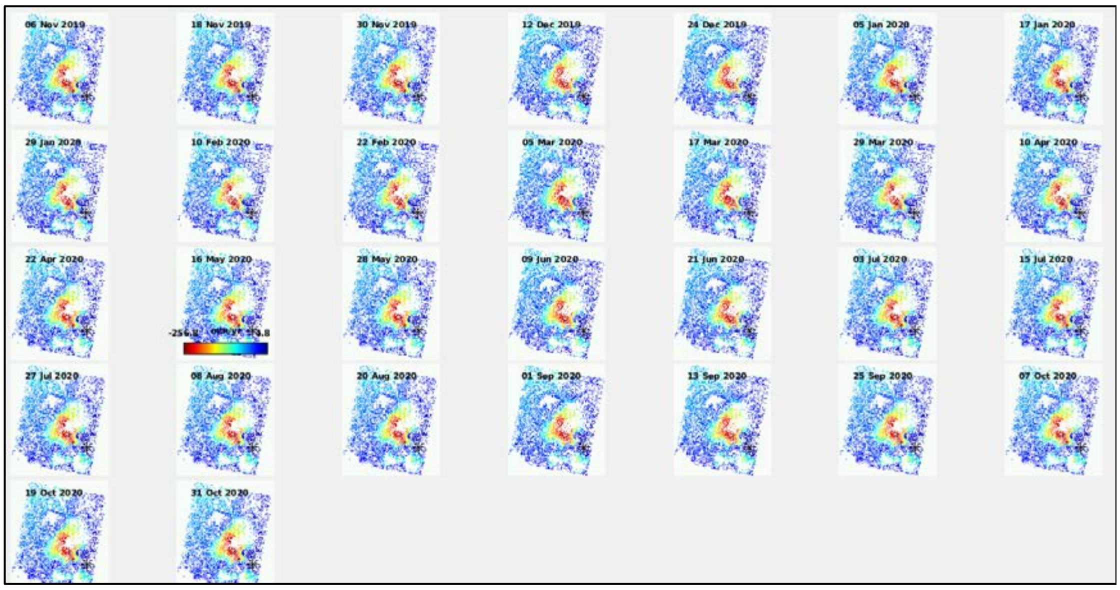




TIP: In our example the linear tropospheric correction seems to uncover OR introduce areas with higher velocity in the lover left corner and other spots always corresponding to higher altitudes. While it is possible that this represents a true deformation, it may also be erroneously introduced by the correction. We can check the effect of each interferogram on the resulting velocity estimate by running:

```
ps_plot('vdrop-do')
```

This step re-estimates the velocity recursively always removing one image. In the resulting plots you can then identify the date which had the greatest effect (if any) and remove it from the processing.



In our timeseries it appears that the effect of the atmosphere in any one image is very small as no differences can be easily seen.

To truly assess whether the deformation in the lover left corner is real or introduced we would need more information about the specific area and possibly also GPS measurements.

The velocity values that we see are relative to the mean velocity of the whole image. We would of course like to have a known velocity point so we could convert them to the absolute velocity values in the LOS direction. We can also set a reference area – with known velocity as our reference. In general, you will want to find an area where no change is expected (if you don't have GPS measurements).

For our study area we will select a circular reference area in the west of the city with the radius of 100m.

```
setparm('ref_centre_lonlat', [-99.280 19.446])
setparm('ref_radius', 100)
```

Note that the reference point here is selected randomly in an area on the slope of the mountain where less subsidence is expected. To have precise result we would need a GPS point or better knowledge of the area!

You can then visualize the 'v', 'v-do' and 'v-dao'.

Next, we can also estimate the standard deviation of the velocity estimates (relative to the reference area) with DEM errors and ramps subtracted, run:

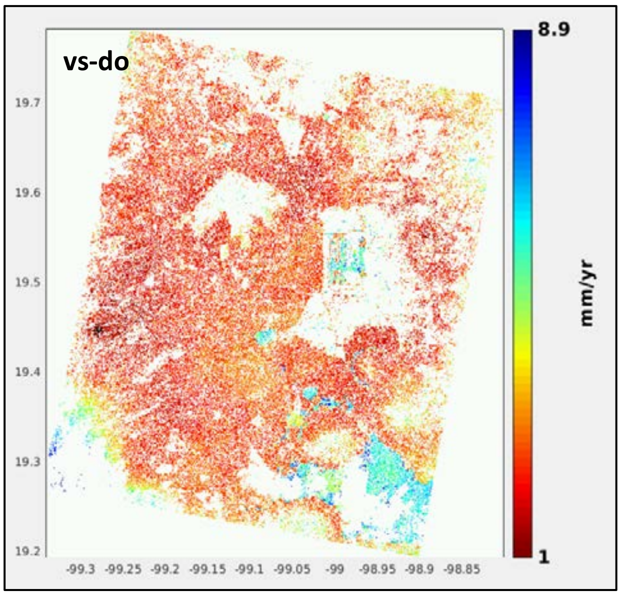
```
ps_plot('vs-do')
```

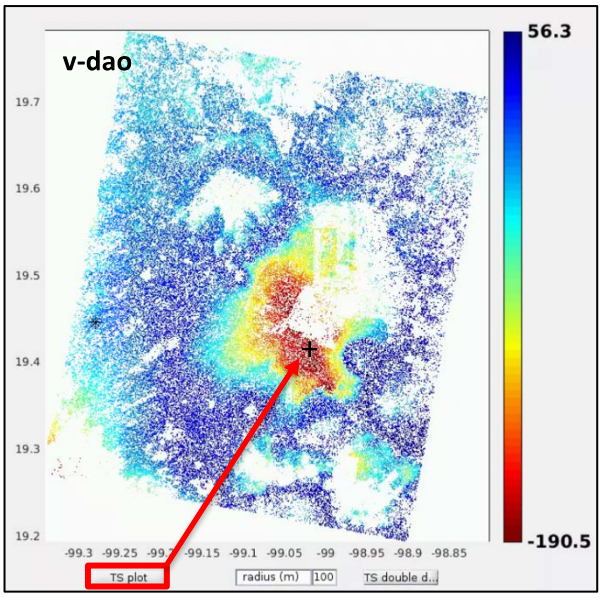
Areas with larger standard deviation indicate either larger errors (due perhaps to atmosphere or unwrapping errors) or deformation that is non-linear. We can plot the timeseries of points by using the 'ts'option.

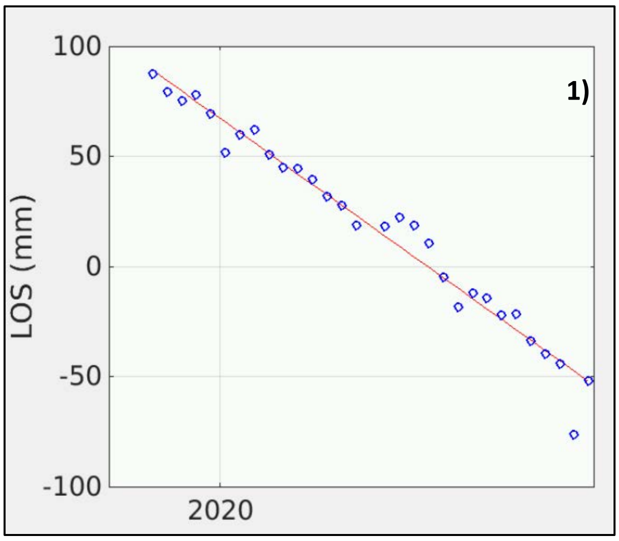
```
ps_plot('vs-dao', 'a_linear', 'ts')
```

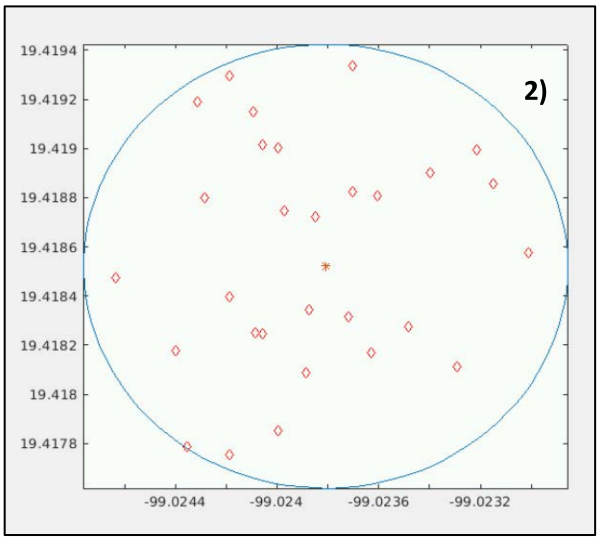
Then you can click at the TS plot button and select a point of interest in plot. Two additional plots will appear:

1) Time series plot for points in 100m radius around the selected point
2) Number and location of pints found in 100m radius









7.13 Export to StaMPSVisualizer

Apart from visualizing the data in MATLAB directly, we can also export them and use the StaMPS Visualizer tool created by Thorsten Höser and implemented using R and R Studio.

We can export the data by running the following code in MATLAB (code taken from the StaMPS Visualizer Manual):

```
ps_plot('v-dao', ‘a_linear’,'ts');
```

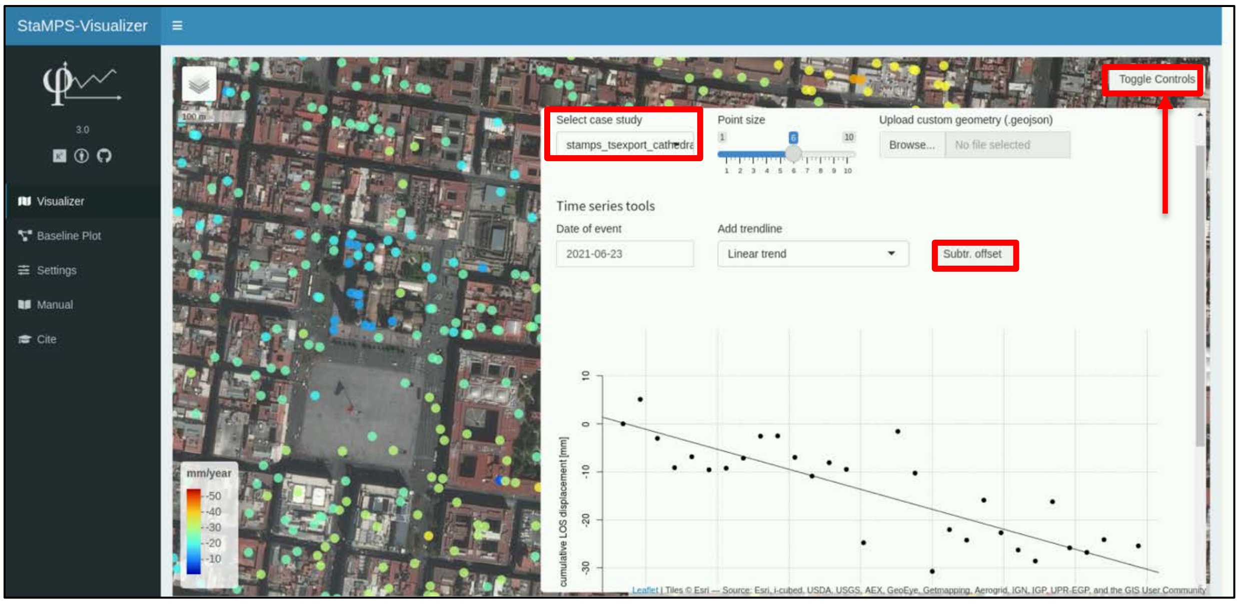
A new window will open; in the new window select a radius and location of the radius centre to select the PS to export. Then run

```
load parms.mat;

% the 'v-dao' parameter is an example you can change it to your needs
% but be sure that you use the same paramters as above in the ps_plot()!
ps_plot('v-dao', ‘a_linear’, -1);

load ps_plot_v-doa.mat;
lon2_str = cellstr(num2str(lon2));
lat2_str = cellstr(num2str(lat2));
lonlat2_str = strcat(lon2_str, lat2_str);

lonlat_str = strcat(cellstr(num2str(lonlat(:,1))),
cellstr(num2str(lonlat(:,2))));
ind = ismember(lonlat_str, lonlat2_str);

disp = ph_disp(ind);
disp_ts = ph_mm(ind,:);
export_res = [lonlat(ind,1) lonlat(ind,2) disp disp_ts];

metarow = [ref_centre_lonlat NaN transpose(day)-1];
k = 0;
export_res = [export_res(1:k,:); metarow; export_res(k+1:end,:)];
export_res = table(export_res);

% you can specify the location and name of the .csv export by renaming the second
parameter
writetable(export_res,'stamps_tsexport.csv')
```

Once the process finishes, you will find a file named **"stamps_tsexport.csv"** in the ***INSAR_20200504*** folder. Copy/Move this file into the ***input/stusi*** folder inside the StaMPS Visualizer folder. In case of RUS it will likely be located in ***/home/rus/StaMPS_Visualizer/input/stusi/.***

Then you can open the R Studio and load the ***ui.R*** file and in the upper left corner you can see a button **Run app**. Click on it to open the StaMPS Visualizer window.

Click on Toggle Controls, select the case study file we have just exported and then you can zoom in to an area of interest (e.g. the Mexico City Metropolitan Cathedral) and click on a point to visualize the timeseries. You can further customize the plot by subtracting the offset (displacement will be calculated from the start of the timeseries instead from the date of the master image).


Select case study: stamps_tsexport_cathedra

Point size: 6

Upload custom geometry (.geojson): No file selected

Time series tools

Date of event: 2021-06-23

Add trendline: Linear trend

StaMPS-Visualizer
3.0
Visualizer
Baseline Plot
Settings
Manual
Cite
Toggle Controls
Browse...
Subtr. offset
100 m
mm/year
-50
-40
-30
-20
-10
cumulative LOS displacement [mm]
10
0
-10
-20
-30
Leaflet | Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

7.14 Output to Google Earth

Finally, we can export our output velocity to be visualized in Google Earth. There is a dedicated tool in StaMPS called ps_gescatter('KML_FILE_NAME', 'INPUT_DATA, STEP, OPACITY).

If we export the entire dataset, we will inevitably crash the Google Earth application and the creation of the file will also take a very long time. If we want to see the overview of our entire study area, then we can set step to 50 for example – this will remove a lot of detail, however.

```
ps_plot('v-dao', 'a_linear', -1)
load ps_plot_v-dao
ps_gescatter('ps_velocity.kml', ph_disp, 50,1)
```

If we want to have a detailed view of a smaller area, we can use only a single patch (NOTE: you will need to re-run steps 6 and 7 inside the patch directory and then run the steps above also from inside the directory changing the step value to 1).

THANK YOU FOR FOLLOWING THE EXERCISE!

8 Extra steps

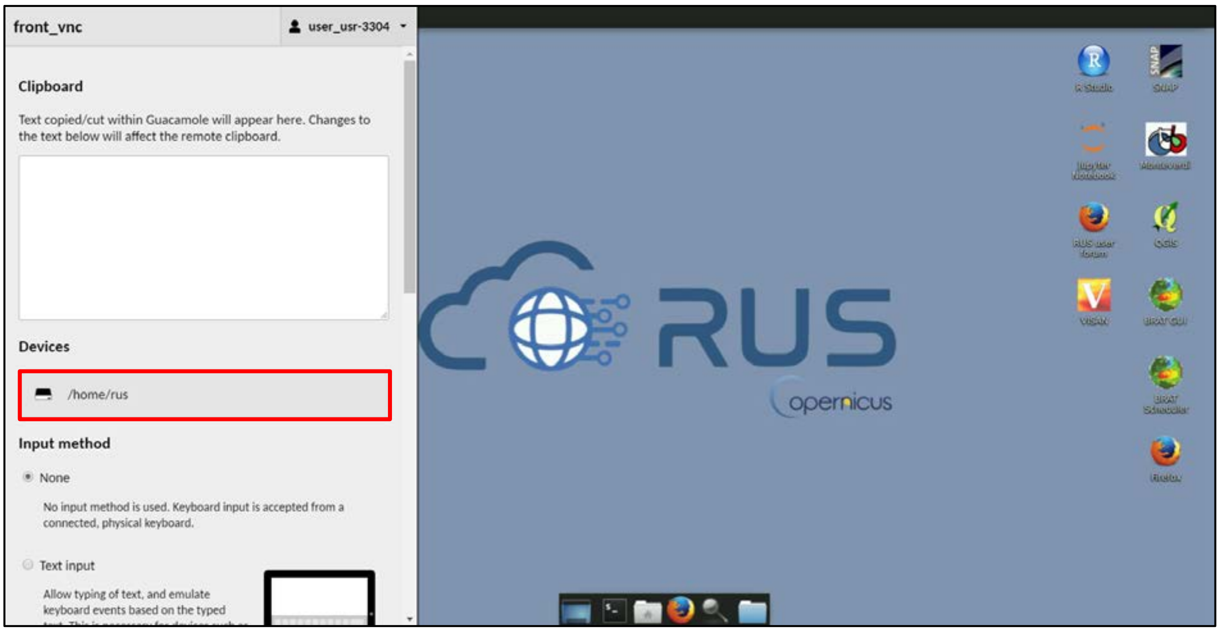
8.1 *Atmospheric correction*

8.2 *Downloading the outputs from the VM*

In your VM, press **Ctrl+Alt+Shift.**

A pop-up window will appear on the left side of the screen. Click on the bar below **Devices**, navigate to the folders you have saved the files you want to download and **double click** on them. The downloading process to your local computer will start automatically.

Once the KML files have been downloaded, you can load and visualize them in **Google Earth**.



front_vnc

user_usr-3304

Clipboard

Text copied/cut within Guacamole will appear here. Changes to the text below will affect the remote clipboard.

Devices

/home/rus

Input method

None

No input method is used. Keyboard input is accepted from a connected, physical keyboard.

Text input

Allow typing of text, and emulate keyboard events based on the typed

9 References

9.1 *PSI resources*

Crosetto, M., Monserrat, O., Cuevas-González, M., Devanthéry, 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

Delgado Blasco, J. M., Foumelis, M., Stewart, C., & Hooper, A. (2019). Measuring Urban Subsidence in the Rome Metropolitan Area (Italy) with Sentinel-1 SNAP-StaMPS Persistent Scatterer Interferometry. *Remote Sensing*, 11(2), 129. https://doi.org/10.3390/rs11020129

Jia, H., & Liu, L. (2016). A technical review on persistent scatterer interferometry. *Journal of Modern Transportation*, 24(2), 153–158. https://doi.org/10.1007/s40534-016-0108-4

9.2 *Tutorials*

GIS-Blog - Matthias Schlögl – Using StaMPS/MTI for PSI Analysis (post series) - https://www.gis-blog.com/stamps-1/
A. Hooper – StaMPS Persistent Scatterer Exercise (2015) – ESA Land Training Course
http://seom.esa.int/landtraining2015/files/Day_4/D4P2a_LTC2015_Hooper.pdf

FOLLOW US!!!