

1. Quick recap

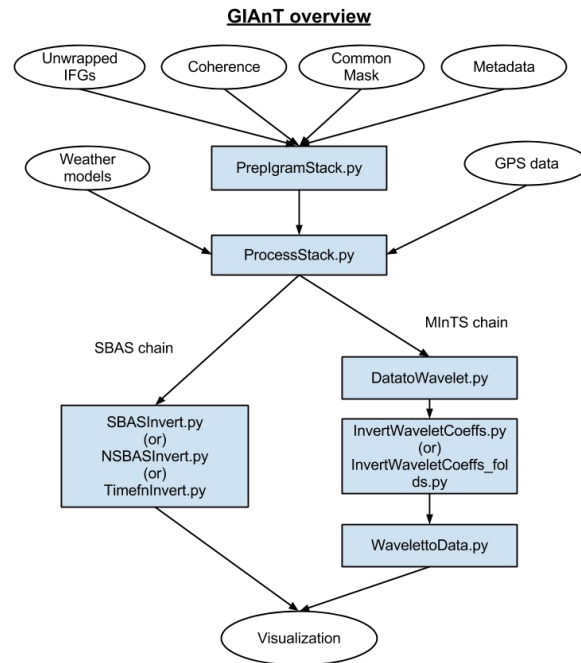
So far, we have gathered all the required network, unwrapped phase and coherence information into a HDF5 file using “PreplgramStack.py” in the previous tutorial.

```
> cd /home/ubuntu/data/giant/kilaeau/GIAnT
> h5ls Stack/RAW-STACK.h5
Jmat                Dataset {46, 17}
bperp               Dataset {46}
cmask               Dataset {1920, 2118}
dates               Dataset {17}
igram               Dataset {46, 1920, 2118}
tims                Dataset {17}
usat                Dataset {17}
```

In this tutorial, we will apply optional corrections to the ingested stack and estimate the deformation time-series using the SBAS technique. We will also teach users to interactively visualize some of the time-series results.

2. Setting up the processing parameters

In the previous tutorial, we described how the dataset parameters are controlled using “data.xml”. In this tutorial, we will learn to set up the processing parameters using a similar XML file - “sbas.xml”. This processing file is specific for the SBAS set of time-series inversions (See figure below).



In the example dataset directory, you will find a script named “prepsbasxml.py” .

```
> less prepsbasxml.py
#!/usr/bin/env python

import tsinsar as ts
import argparse
import numpy as np

if __name__ == '__main__':
    g = ts.TSXML('params')
    g.prepare_sbas_xml(nvalid = 30, netramp=True, atmos='',
                      demerr = False, filt=0.05)

    g.writexml('sbas.xml')
```

The complete list of all configurable parameters in “sbas.xml” can be found in the [GIAnt user manual](#). We describe the parameters that we have set up using prepsbasxml.py below:

| | |
|---------|--|
| nvalid | Used for NSBAS inversion. Determines the minimum number of interferograms that a pixel should be coherent to be considered for inversion. |
| netramp | Boolean parameter controlling the deramping of interferograms. In this case, the applied ramp corrections are consistent over the entire network. |
| atmos | ProcessStack.py can download weather model data and use that for stratified tropospheric phase delay correction. This is beyond the scope of this tutorial. We use a empty string to indicate that no weather model corrections are to be applied. |
| demerr | Boolean parameter indicating if a DEM error term needs to be estimated. The baseline information from ifg.list is used for DEM error estimation. |
| filt | Width of the Gaussian filter to applied to the raw time-series to obtain the smoothed estimates. The value of this parameter is in years. |

See [GIAnt user manual](#) for complete list of options and default values.

```
> python prepsbasxml.py
```

To view the generated “sbas.xml” file,

```
> less sbas.xml
<params>
  <proc>
    <nvalid>
      <value>30</value>
      <type>INT</type>
      <help>Minimum number of coherent IFGs for a single pixel. If
zero, pixel should be coherent in all IFGs.</help>
    </nvalid>
    <uwcheck>
      <value>False</value>
      <type>BOOL</type>
    </uwcheck>
```

```

    <netramp>
      <value>True</value>
      <type>BOOL</type>
      <help>Network deramp. Remove ramps from IFGs in a network
sense.</help>
    </netramp>
    <gpsramp>
      <value>False</value>
      <type>BOOL</type>
      <help>GPS deramping. Use GPS network information to correct
ramps.</help>
    </gpsramp>
    <stnlist>
      <value></value>
      <type>STR</type>
      <help>Station list for position of GPS stations.</help>
    </stnlist>
    ....
  </params>

```

Remember that the generated XML file can be modified in a text editor, and we again include a help string to describe each of the parameters in the file.

We are now ready to process our stack from the HDF5 file.

3. ProcessStack.py - Applying corrections

The first stage of processing, in which the data supplied by the users is modified, is accomplished using “ProcessStack.py”. The aim of this step is to

1. Correct for stratified troposphere artifacts, either
 - a. Empirically by looking at relationship between InSAR phase and DEM
 - b. Using weather models through the PyAPS package
2. Estimate ramps introduced due to orbital errors, either
 - a. Either empirically by fitting a predefined orbit error function to data
 - b. Using dense GPS observations

All the corrections are applied consistently across the interferogram network.

For this tutorial, we only choose to empirically deramping of interferograms. Details regarding other options and the associated fields in “sbas.xml” can be found in the [GIAnt user manual](#).

Run “ProcessStack.py”

(NOTE: The ProcessStack.py command needs to be run from the X11 windows in the Remote Desktop function of EarthKit.)

```
> pwd
/home/ubuntu/data/giant/kilaeau/GIAnt

> ProcessStack.py
logger - INFO - GIANT Toolbox - v 1.0
logger - INFO - -----
<module> - INFO - Input h5file: Stack/RAW-STACK.h5
<module> - INFO - Deleting previous Stack/PROC-STACK.h5
<module> - INFO - Output h5file: Stack/PROC-STACK.h5
deramp - INFO - PROGRESS: Estimating individual ramps.
[===== 98% =====> ]      17s /      0s
deramp - INFO - PROGRESS: Network deramp of IFGs.
[===== 98% =====> ]      27s /      0s
<module> - INFO - PNG preview of Deramped images: Figs/Ramp
[===== 11% =====> ]      42s /    342s
```

Outputs of “ProcessStack.py” include - a processed stack file “Stack/PROC-STACK.h5” and a directory of PNG previews of deramped interferograms “Figs/Ramp”.

```
> ls Stack
PROC-STACK.h5 RAW-STACK.h5
```

To preview the contents of the new stack file

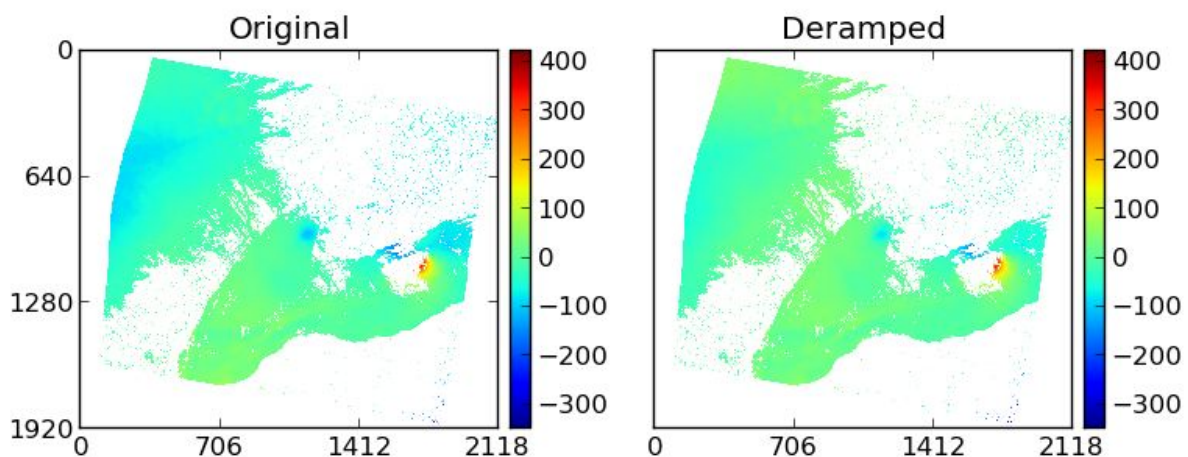
```
>h5ls Stack/PROC-STACK.h5
Jmat                Dataset {46, 17}
bperp               Dataset {46}
cmask               Dataset {1920, 2118}
dates               Dataset {17}
figram              Dataset {46, 1920, 2118}
ramp                Dataset {46, 3}
tims                Dataset {17}
```

To view the contents of the directory with the PNG previews.

```
> ls Figs/Ramp
I001.png I007.png I013.png I019.png I025.png I031.png I037.png
I043.png
I002.png I008.png I014.png I020.png I026.png I032.png I038.png
I044.png
.....
```

To see the effect of deramping on the 7th interferogram in the Stack: (NOTE: you need to run this from Remote Desktop for the graphical viewer)

```
> eog Figs/Ramp/I007.png
```



Our stack is now deramped and ready for the final time-series inversion.

4. NSBASInvert.py - Final inversion

In this tutorial, we demonstrate the simplest time-series inversion algorithm implemented in GIANt - the NSBAS algorithm. GIANt also implements two other algorithms in the SBAS chain - SBASInvert.py and TimefnInvert.py. The detailed discussion on the differences between these approaches can be found in the [GIANt user manual](#).

The NSBAS algorithm estimates the differential displacement between one SAR acquisition and the next using a simple least squares approach. Optionally, it uses a user-defined temporal form to connect disconnected components of an interferogram network. By default, NSBAS uses a quadratic polynomial for connecting disconnected components. Our implementation of the algorithm estimates the time-series only for the pixels that are considered coherent over at least “nvalid” interferograms in the entire stack.

```
> NSBASInvert.py -nproc 4
logger - INFO - GIANt Toolbox - v 1.0
logger - INFO - -----
Timefn - INFO - Adding 17 linear pieces (SBAS)
<module> - INFO - No modification in the NSBAS constraint functional
form
<module> - INFO - Assuming a quadratic polynomial form
Timefn - INFO - Adding order 0 at T = 0.000000
Timefn - INFO - Adding order 1 at T = 0.000000
Timefn - INFO - Adding order 2 at T = 0.000000
<module> - INFO - Output h5file Stack/NSBAS-PARAMS.h5
<module> - INFO - Number of parallel processes: 4
<module> - INFO - Relative weight of polynomial constraint :
1.000000e-04
[===== 99% =====> ]      800s /      0s
```

This script has been parallelized for performance, and “-nproc 4” on the command line instructions the program to launch the analysis on 4 threads.

“NSBASInvert.py” stores the inversion results in “Stack/NSBAS-PARAMS.h5”.

```
> h5ls Stack/NSBAS-PARAMS.h5
bperp          Dataset {46}
cmask          Dataset {1920, 2118}
dates          Dataset {17}
gamma          Dataset {SCALAR}
ifgcnt         Dataset {1920, 2118}
mName          Dataset {3}
```

| | |
|-----------|--------------------------|
| masterind | Dataset {SCALAR} |
| parms | Dataset {1920, 2118, 3} |
| rawts | Dataset {17, 1920, 2118} |
| recons | Dataset {17, 1920, 2118} |
| regF | Dataset {3} |
| tims | Dataset {17} |

Note that the HDF5 file contains the raw time-series estimates (rawts) as well as the filtered time-series estimates (recons). In the next couple of sections, we will describe the visualization tools that are included with GIANt.

5. plotts.py - Interactive visualization

GIANT includes a script called “plotts.py” for interactive visualization of the generated time-series products. “plotts.py” requires a graphical desktop to run. It also requires that you set the matplotlib environment for an X-windows environment. To set matplotlib to run successfully in X-windows, edit this file:

```
> nano ~/.matplotlib/matplotlibrc
```

and set this value:

```
backend : TkAgg
```

Now we are ready to run “plotts.py”. Running the script the “-h” option list all the input parameters that can be controlled from command line.

```
> plotts.py -h
logger - INFO - GIANT Toolbox - v 1.0
logger - INFO - -----
usage: plotts.py [-h] [-e] [-f FNAME] [-i TIND] [-m MULT] [-y YLIM
YLIM]
                    [-ms MSIZE] [-raw] [-model] [-mask MASK MASK] [-zf]
```

Interactive SBAS time-series viewer

optional arguments:

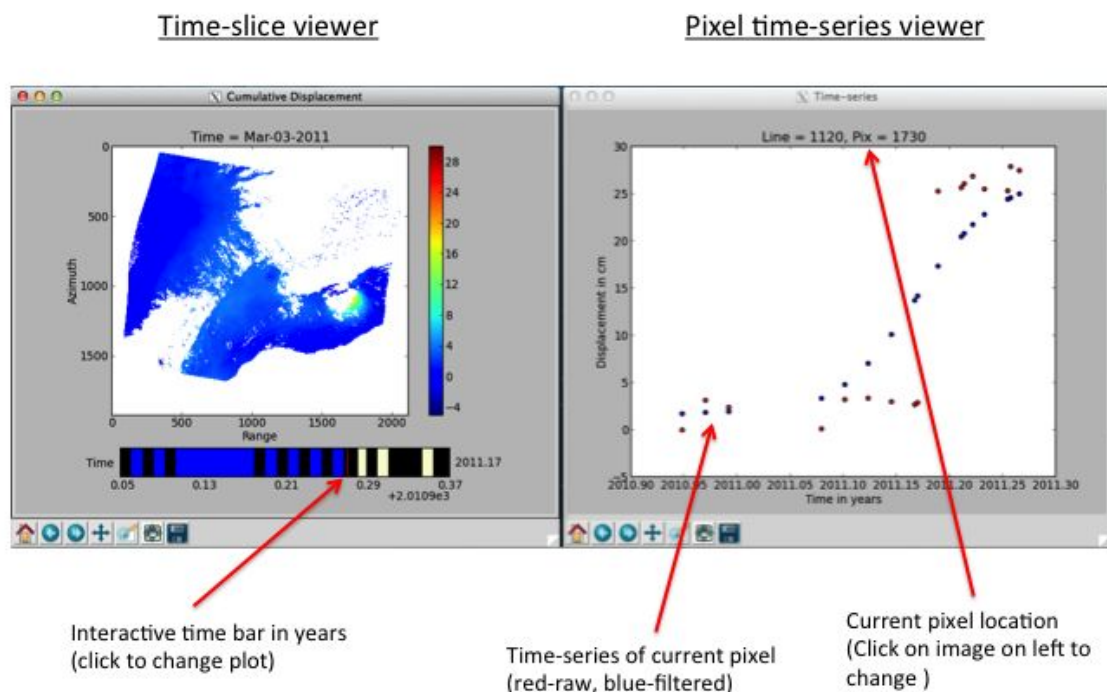
| | |
|--------------------------|--|
| -h, --help | show this help message and exit |
| -e | Display error bars if available. Default: False |
| -f FNAME | Filename to use. Default: Stack/LS-PARAMS.h5 |
| -i TIND | Slice to display. Default: Middle index |
| -m MULT | Scaling factor. Default: 0.1 for mm to cm |
| -y YLIM YLIM | Y Limits for plotting. Default: [-25,25] |
| -ms MSIZE | Marker size. Reduce if error bars are too small. |
| Default: 5 | |
| -raw | Plot Un-Filtered Time Series as well, if available |
| -model | Plot the individual model components as well. For |
| NSBAS, Timefn and MINTS. | |
| -mask MASK MASK | To mask out values. Need to provide 2 inputs - |
| Mask file in | |
| | float and xml file with dimensions. Default: None |
| -zf | Changes time-origin to first acquisition for |

showing time-series.

To visualize the output from “NSBASInvert.py”,

```
> plotts.py -f Stack/NSBAS-PARAMS.h5 -y -5 30 -raw
```

This will open two plot windows - an interactive time-slice viewer and a pixel time-series viewer as shown below. The colorbar for the slice viewer ranges from -5 to 30 cm, and the raw time-series (red dots) is also shown along with the filtered time-series (blue); as requested using the command line flags -y and -raw.



Users can now view different time-slices by clicking on the time-bar and can view the time-series for different pixels by directly clicking on the pixel of interest in the image.

Some observations:

1. Our analysis clearly captures the large offset associated with the Mar 6, 2011 dike injection on Kilauea volcano East Rift zone.
2. GIANt implements a simple Gaussian weighted moving average filter. Hence, the discrepancy between the raw displacement observations (red pixels) and the filtered observations (blue pixels).
3. Users can implement their own custom filtering with the raw time-series included in the HDF5 file.

Besides `plots.py`, GIANt can also export results as a movie through the “`make_movie.py`” script and as a Google Earth ready KML using “`make_kml.py`” scripts. For details and usage, refer to the [GIANt user manual](#). GIANt also includes tools to export these datasets into GMT's netcdf format and a GDAL compatible VRT.