1. Post-Processing UAVSAR Stack data with isceApp.py

In this lab, you will learn how to process UAVSAR Stack data, while learning about the ISCE application `isceApp.py`. Previous labs have used the application `insarApp.py` to process pairs of raw or single look complex data acquired on two different dates using spaceborne sensors into interferograms and geocoded products. In this lab we will work with the application `isceApp.py` to post-process several data sets from the same flight track acquired at different times with the UAVSAR radar flown on an airplane. The data downloaded from the UAVSAR website have already been processed to single look complex images by the UAVSAR team. Post-processing includes the following steps: forming interferograms from the slc data, removing topographic phase, filtering, unwrapping, and geocoding.

To get started change directory to the /data/lab11 directory (click the "Launch" button if you haven't already done so),

```
> cd /data/lab11
```

Take a look at the directory contents with the "ls -l" command,

```
> ls -l
demLat_N38_N39_Lon_W123_W121.dem.wgs84.xml
incoming -> /data/sites/Napa_uavsar_stack/incoming
isceApp.xml
precooked -> /data/sites/Napa_uavsar_stack/
SanAnd_05510_01_BC.dop -> incoming/SanAnd_05510_01_BC.dop
SanAnd_05510_12128_000_121105_L090HH_01_BC.ann ->
incoming/SanAnd_05510_12128_000_121105_L090HH_01_BC.ann
SanAnd_05510_12128_000_121105_L090HH_01_BC_s1_1x1.slc ->
incoming/SanAnd_05510_12128_000_121105_L090HH_01_BC_s1_1x1.slc
SanAnd_05510_13089_001_130508_L090HH_01_BC.ann ->
incoming/SanAnd_05510_13089_001_130508_L090HH_01_BC.ann
SanAnd_05510_13089_001_130508_L090HH_01_BC_s1_1x1.slc ->
incoming/SanAnd_05510_13089_001_130508_L090HH_01_BC_s1_1x1.slc
SanAnd_05510_13165_004_131031_L090HH_01_BC.ann ->
incoming/SanAnd_05510_13165_004_131031_L090HH_01_BC.ann
SanAnd_05510_13165_004_131031_L090HH_01_BC_s1_1x1.slc ->
incoming/SanAnd_05510_14068_000_140529_L090HH_01_BC_s1_1x1.slc
SanAnd_05510_14068_000_140529_L090HH_01_BC.ann ->
incoming/SanAnd_05510_14068_000_140529_L090HH_01_BC.ann
SanAnd_05510_14068_000_140529_L090HH_01_BC_s1_1x1.slc
SanAnd_05510_14128_003_140829_L090HH_01_BC.ann ->
incoming/SanAnd_05510_14128_003_140829_L090HH_01_BC.ann
SanAnd_05510_14128_003_140829_L090HH_01_BC_s1_1x1.slc ->
```

```
incoming/SanAnd_05510_14128_003_140829_L090HH_01_BC_s1_1x1.slc
```

You see that we have prepared this directory with some files and some symbolic links to data. We have provided a symbolic link named "`precooked`" to a directory (indicated with the `->` symbol following its name) where we have previously post-processed a stack of 12 SLCs from the San Andreas fault in the Napa area of California. In the interest of time, in this lab we will illustrate how to post-process these data using a short stack of only a few of those SLCs just before and after the recent earthquake in that area. The full stack will be used in Lab 12 on using GIAnT to create a time series of the deformation. The symbolic links in the current directory ending in "`.ann`" and "`.slc`" contain the meta data and the single look complex data downloaded from the UAVSAR website that we will use in the current lab.

We will start the processing of these data now while going on with the tutorial exposition because it will take a while for the data to process. You can keep an eye on the processing in the terminal pane while continuing to read the tutorial notes in this pane of your web browser window. If you haven't already done so, you can start the remote desktop now to have access to another terminal window and to display images.

To start the processing, you simply enter the following command,

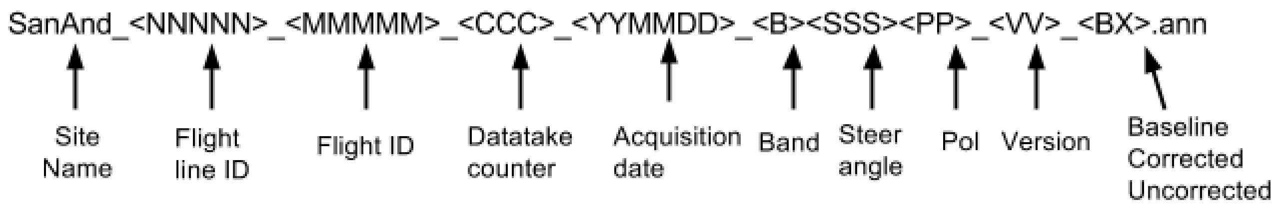
```
> isceApp.py
```

After you launch this command you should see a stream of information going to the terminal window.

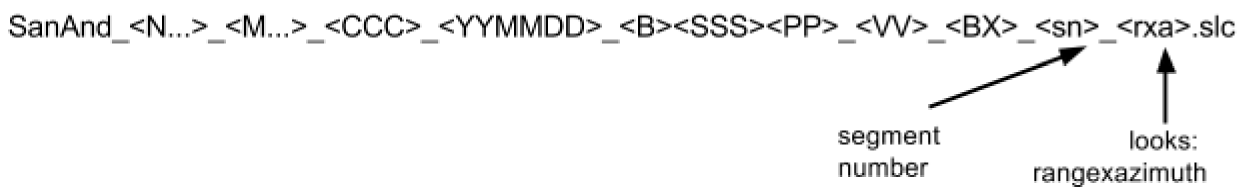
The remaining sections in this lab will explain how to understand the names of the downloaded UAVSAR files, the input files read by ISCE when you entered the command `isceApp.py`, and will lead you through exploring the data products with `mdx.py`.

2. Understanding UAVSAR Data Set Names

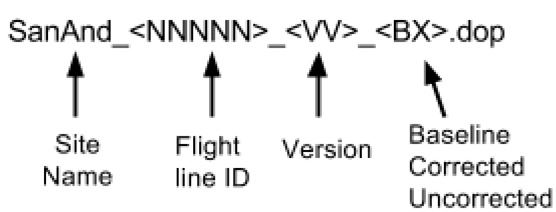
Annotation file:



Single look complex image file:



Doppler file:



The graphic above shows the standard naming conventions for UAVSAR data files. UAVSAR data come with a metadata file called an annotation file with extension "`.ann`", a set of single look complex files with extensions "`.slc`", and a file containing Doppler values as a function of range location with extension "`.dop`".

There is one annotation file for each Flight ID or acquisition date (except in the rare case of multiple flights on one date when only the Flight ID will be different). The SLC data are cut into multiple along track segments within each Flight ID. In this lab we are only working with segment 1 for each Flight ID. The annotation file for each Flight ID contains information on all of the SLC segments. There is one Doppler file for each Flight line. The UAVSAR team processes all SLCs in a Flight ID stack to the same Doppler and the same coordinate system (described by the Peg point contained in the annotation files). All the SLC images in a stack are either baseline corrected (BC) or baseline uncorrected (BU), depending on whether the residual baseline correction was applied.

3. Understanding the ISCE xml input files for stack processing

When you issued the `isceApp.py` command earlier in Section 1 you may have wondered how `isceApp.py` received input information on what to process and how to set processing options. ISCE will read configuration data from appropriately named files in the local directory. For the application, `isceApp.py` the name can either be `isce.xml` or `isceApp.xml`. In fact you could have two files using both of these names with either complimentary or conflicting information. In the case of complimentary information the union of the information in the two files will be used. In the case of conflicting information the information in the file named `isceApp.xml` will win. It is also possible to name an input file on the command line with any name desired (as was done in some of the earlier labs); the file on the command line will win in the case of conflicting information amongst the different configuration files. You can issue the following two commands to see that we have placed a file with name `isceApp.xml` in the directory, but not one with name `isce.xml`.

```
> ls isceApp.xml
isceApp.xml
> ls isce.xml
ls: cannot access isce.xml: No such file or directory
```

When we run `isceApp.py` ISCE loads the contents of the `isceApp.xml` file found in the local directory and configures the application.

Let’s look at the contents of the input file,

```
> cat isceApp.xml
<?xml version="1.0" encoding="UTF-8"?>

<isceApp>
  <component name="isce">
      <property name="sensor name">UAVSAR_Stack</property>
      <property name="resamp range looks">   6</property>
      <property name="resamp azimuth looks">16</property>
      <property name="do unwrap">True</property>
      <property name="unwrapper name">icu</property>
      <property name="output directory">.</property>

      <component name="stack">
<!--
      <component name="Scene1">
```

```
<property name="id">uav1</property>
<property name="hh">
            SanAnd_05510_09006_011_090218_L090HH_01_BC.ann
      </property>
</component>

<component name="Scene2">
<property name="id">uav2</property>
<property name="hh">
         SanAnd_05510_09091_005_091117_L090HH_01_BC.ann
      </property>
</component>

<component name="Scene3">
<property name="id">uav3</property>
<property name="hh">
          SanAnd_05510_10037_009_100511_L090HH_01_BC.ann
      </property>
</component>

<component name="Scene4">
<property name="id">uav4</property>
<property name="hh">
         SanAnd_05510_10077_010_101028_L090HH_01_BC.ann
      </property>
</component>

<component name="Scene5">
<property name="id">uav5</property>
<property name="hh">
          SanAnd_05510_11049_008_110713_L090HH_01_BC.ann
      </property>
</component>

<component name="Scene6">
<property name="id">uav6</property>
<property name="hh">
          SanAnd_05510_11071_012_111103_L090HH_01_BC.ann
      </property>
</component>

<component name="Scene7">
<property name="id">uav7</property>
```

```
    <property name="hh">
                SanAnd_05510_12017_007_120418_L090HH_01_BC.ann
          </property>
    </component>

    <component name="Scene8">
    <property name="id">uav8</property>
    <property name="hh">
               SanAnd_05510_12128_000_121105_L090HH_01_BC.ann
          </property>
    </component>
-->

    <component name="Scene9">
    <property name="id">uav9</property>
    <property name="hh">
               SanAnd_05510_13089_001_130508_L090HH_01_BC.ann
          </property>
    </component>

    <component name="Scene10">
    <property name="id">uav10</property>
    <property name="hh">
              SanAnd_05510_13165_004_131031_L090HH_01_BC.ann
         </property>
    </component>

    <component name="Scene11">
    <property name="id">uav11</property>
    <property name="hh">
               SanAnd_05510_14068_000_140529_L090HH_01_BC.ann
          </property>
    </component>

    <component name="Scene12">
    <property name="id">uav12</property>
    <property name="hh">
              SanAnd_05510_14128_003_140829_L090HH_01_BC.ann
          </property>
    </component>
    </component>

    <property name="selectPols">hh</property>
```

```
      <property
name="selectPairs">uav10-uav12,uav9/uav11,uav9/uav12</property>

      <property name="coregistration strategy">single
reference</property>
            <property name="reference scene">uav9</property>
      <property name="reference polarization">hh</property>

      <property name="geocode list">
          ["filt_topophase.flat", "filt_topophase.unw"]
      </property>

      <component name="dem">
          <catalog name="dem">
                    demLat_N38_N39_Lon_W123_W121.dem.wgs84.xml
               </catalog>
      </component>

   </component>
</isceApp>
```

This is an edited version of the file that was used in post-processing the data in the “precooked” directory. The file lets isceApp.py know that the “sensor name” is UAVSAR_Stack so that the appropriate code for handling the sensor meta data and image data will be used. It sets a few processing options such as the number of range and azimuth looks and the name of the unwrapper to use. Then it has a “component” section giving the names of the input annotation files for each image in the stack. In the precooked directory we used 12 input images. In this directory we have commented out (xml comment begins with “<!--” and ends with “-->”) all of the scenes except 9-12, the ones just before and after the recent Napa earthquake. After the scenes list in the stack, the reference scene is selected and the pairs to be post-processed are selected. The specification of the pairs using a “/” symbol simply means to form a single pair from the two named scenes. So, `uav9/uav11` means form an interferogram from the SLC labelled `uav9` and the SLC labelled `uav11` in the definition of the stack. The “-” symbol between two scenes is shorthand for all possible unique combinations between the first and last scene. So, `uav10-uav12` would expand into `uav10/uav11, uav10/uav12, uav11/uav12`.

The `isceApp.xml` gives the name of the annotation files provided by the UAVSAR project. The annotation file contains the names of the other inputs files used by ISCE, namely the single look complex (SLC) file and the doppler file. The annotation file is an rdf (radar data format) file, which is a text file consisting of lines of the form “keyword (units) = value”. Some example lines used in ISCE from one of the annotation files in this directory follows (`;` starts a comment,

& indicates a string value, − indicates a dimensionless entry),

```
slc_1_1x1 (&) = SanAnd_05510_11071_012_111103_L090HH_01_BC_s1_1x1.slc
dop       (&) = SanAnd_05510_01_BC.dop
slc_1_1x1 Columns   (pixels)  = 9900   ;samples in SLC 1x1 segment 1
slc_1_1x1 Rows      (pixels)  = 66664  ;lines in SLC 1x1 segment 1

1x1 SLC Range Pixel Spacing        (m) = 1.66551366
1x1 SLC Azimuth Pixel Spacing      (m) = 0.6
Global Average Altitude            (m) = 12495.755
Global Average Terrain Height      (m) = 4.61314579
Average Pulse Repetition Interval (ms) = 2.24897112
Peg Latitude                     (deg) =   38.2206738
Peg Longitude                    (deg) = -121.928086
Peg Heading                      (deg) =   55.2680562
Ellipsoid Semi-major Axis          (m) = 6378137.0
Ellipsoid Eccentricity Squared     (-) = 0.00669438
Segment 1 Data Starting Azimuth    (m) = -32011.8
Image Starting Slant Range        (km) = 13.4489379 ;for SLC 1x1 data
Average Along Track Velocity     (m/s) = 246.759825
Minimum Look Angle               (deg) = 21.54218775
Maximum Look Angle               (deg) = 65.29939711
Look Direction                     (&) = Left
Antenna Length                     (m) = 1.5
Polarization                       (&) = HH
Center Wavelength                 (cm) = 23.8403545
Bandwidth                        (MHz) = 80.0
Pulse Length                (microsec) = 40.0
Start Time of Acquisition          (&) =  3-Nov-2011 22:39:41 UTC
Stop Time of Acquisition           (&) =  3-Nov-2011 22:44:22 UTC
```

4. Understanding the output products

After isceApp.py has finished running, take a look at the list of directories and files in the current directory,

```
> ls
catalog
demLat_N38_N39_Lon_W123_W121.dem.wgs84.xml
dop.txt
incoming
isceApp.xml
isce.log
isceProc_20141017013039.xml
precooked
SanAnd_05510_01_BC.dop
SanAnd_05510_12128_000_121105_L090HH_01_BC.ann
SanAnd_05510_12128_000_121105_L090HH_01_BC_s1_1x1.slc
SanAnd_05510_13089_001_130508_L090HH_01_BC.ann
SanAnd_05510_13089_001_130508_L090HH_01_BC_s1_1x1.slc
SanAnd_05510_13089_001_130508_L090HH_01_BC_s1_1x1.slc.xml
SanAnd_05510_13165_004_131031_L090HH_01_BC.ann
SanAnd_05510_13165_004_131031_L090HH_01_BC_s1_1x1.slc
SanAnd_05510_13165_004_131031_L090HH_01_BC_s1_1x1.slc.xml
SanAnd_05510_14068_000_140529_L090HH_01_BC.ann
SanAnd_05510_14068_000_140529_L090HH_01_BC_s1_1x1.slc
SanAnd_05510_14068_000_140529_L090HH_01_BC_s1_1x1.slc.xml
SanAnd_05510_14128_003_140829_L090HH_01_BC.ann
SanAnd_05510_14128_003_140829_L090HH_01_BC_s1_1x1.slc
SanAnd_05510_14128_003_140829_L090HH_01_BC_s1_1x1.slc.xml
uav10
uav10__uav11
uav10__uav12
uav11
uav11__uav12
uav12
uav9
uav9__uav10
uav9__uav11
uav9__uav12
```

Starting from the top of this listing we see a new directory named `catalog`, which contains several text files with data indicating the state of several objects such as the orbit during the processing flow. These may be useful in debugging if something doesn't seem to work properly.

The next new file is the `dop.txt` file, which is a text file containing the input doppler samples from the UAVSAR input products and samples from a polynomial fit to these data done by `isceApp.py`.

The next new file is `isce.log`, which is a text file with logging information from the run of `isceApp.py`. Then there is the file `isceProc_<date-time>.xml` that contains detailed information relevant to the information used in post-processing the data (provenance), including the version of ISCE that was run, the input parameters, and much more. You can use list the contents of the file using the commands '`more`' or '`less`' or '`cat`'.

Next you see a block of files that were originally in the directory with new `slc.xml` files created by `isceApp.py` interspersed in the listing. These files contain useful metadata for the slc files that can be used in displaying those images with the command, `mdx.py`.

Then there are several directories named for the labels of the scenes (`uav9` for example) and the pairs indicated in the input file (*eg.*, `uav9__uav12`), `isceApp.xml`. Use the '`ls`' command to view the contents of those directories.

The reference scene (defined in the xml file) directory,

```
> ls uav9
uav9_hh.raw     uav9.lon.rdr    uav9.los.rdr.xml     uav9.z.rdr
uav9.zsch.rdr.xml
uav9.lat.rdr    uav9.lon.rdr.xml  uav9.simamp.rdr    uav9.z.rdr.xml
uav9.lat.rdr.xml  uav9.los.rdr       uav9.simamp.rdr.xml
uav9.zsch.rdr
```

The file with the extension `.raw` is not actually a raw unfocused file. It is a placeholder for a file expected in the normal flow of `isceApp.py`, which is actually just a symbolic link to the slc file. The `.rdr` files contain the digital elevation model (DEM) lattitude, longitude, and heights resampled in the radar coordinate system, a computed line of sight file to each output pixel in `los.rdr`, and a simulated ampitude file from the DEM in `simamp.rdr`.

One of the pair directories,

```
> ls uav9__uav12
uav9__uav12_hh.dem.crop
uav9__uav12_hh.resampImage.amp
uav9__uav12_hh.dem.crop.xml
uav9__uav12_hh.resampImage.amp.xml
```

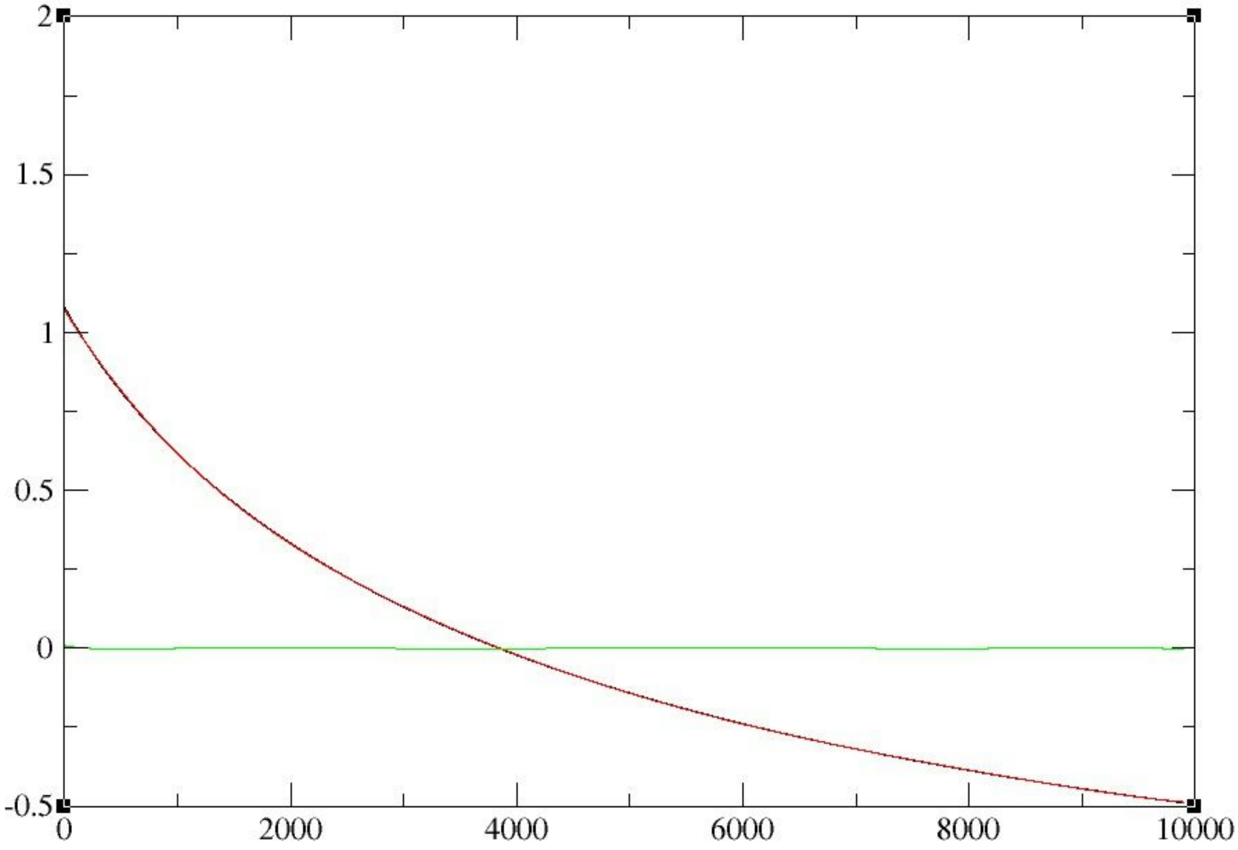
```
uav9__uav12_hh.filt_topophase.conncomp
uav9__uav12_hh.resampImage.int
uav9__uav12_hh.filt_topophase.conncomp.xml
uav9__uav12_hh.resampImage.int.xml
uav9__uav12_hh.filt_topophase.flat
uav9__uav12_hh.resampOnlyImage.amp
uav9__uav12_hh.filt_topophase.flat.geo
uav9__uav12_hh.resampOnlyImage.amp.xml
uav9__uav12_hh.filt_topophase.flat.geo.xml
uav9__uav12_hh.resampOnlyImage.int
uav9__uav12_hh.filt_topophase.flat.xml
uav9__uav12_hh.resampOnlyImage.int.xml
uav9__uav12_hh.filt_topophase.unw
uav9__uav12_hh.topophase.cor
uav9__uav12_hh.filt_topophase.unw.geo
uav9__uav12_hh.topophase.cor.xml
uav9__uav12_hh.filt_topophase.unw.geo.xml
uav9__uav12_hh.topophase.flat
uav9__uav12_hh.filt_topophase.unw.xml
uav9__uav12_hh.topophase.flat.xml
uav9__uav12_hh.geo.log
uav9__uav12_hh.topophase.mph
uav9__uav12_hh.phsig.cor
uav9__uav12_hh.topophase.mph.xml
uav9__uav12_hh.phsig.cor.xml
>
```

5. Visualizing the output products

You can look at a plot of the Doppler centroid as a function of range across the UAVSAR swath and the polynomial fit to it used in ISCE by using the following command in your remote desktop,

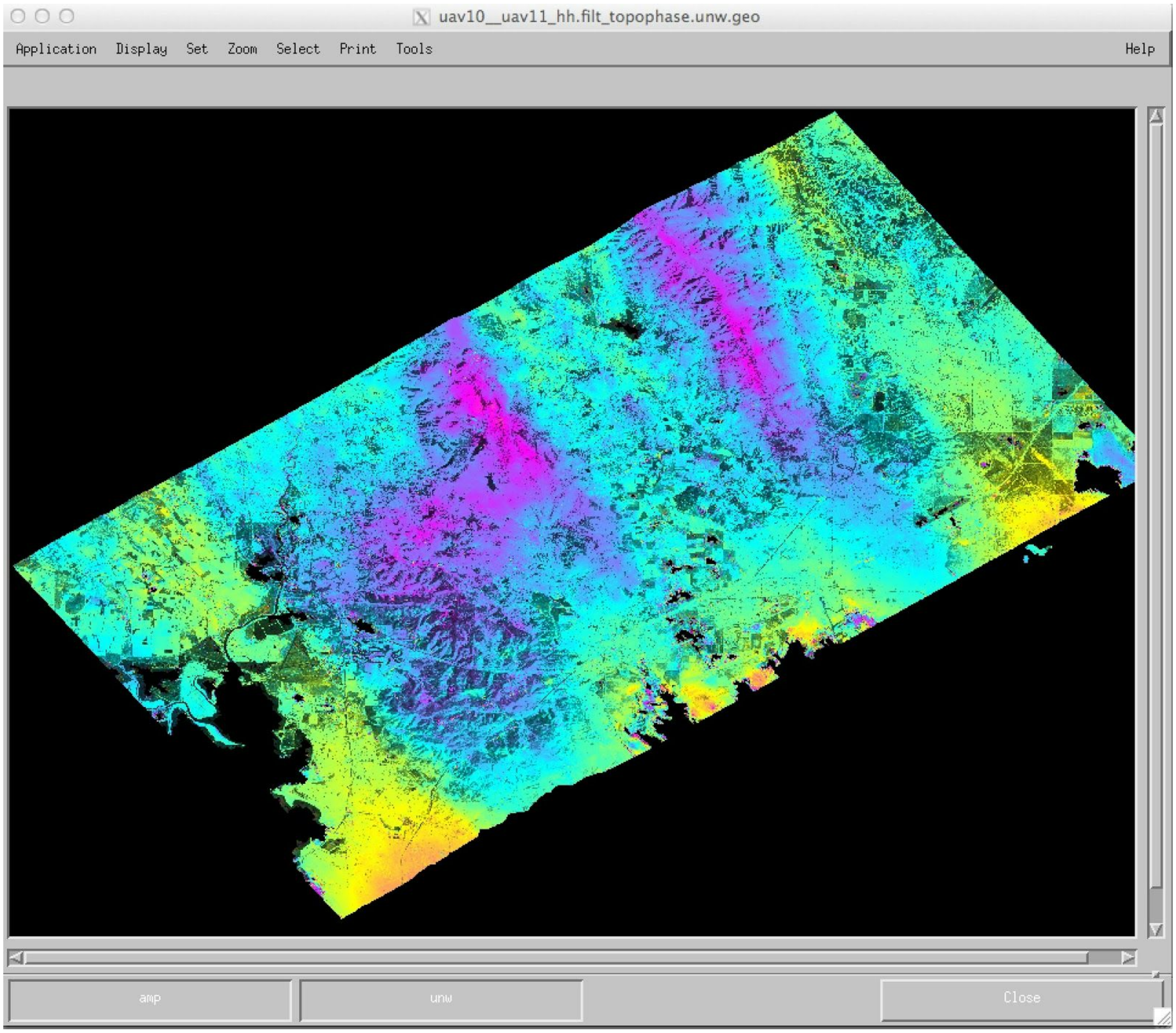
```
> cd /data/lab11
> xmgrace -nxy dop.txt
```

The abcissa is the range bin and the ordinate is the Doppler value normalized by the PRF.



There is a black curve, which is the Doppler data from the UAVSAR project input file, `SanAnd_05510_01_BC.dop`, which is hidden behind the red curve, which is the polynomial fit. The green curve is the residuals from the fit.

For those who are familiar with satellite SAR data, notice that there is a strong variation of the Doppler centroid across the UAVSAR swath caused by the large average yaw or squint of this stack (-1.9 degrees). You can check the yaw of the stack provided in the annotation file with this command:




```
grep Yaw SanAnd_05510_13089_001_130508_L090HH_01_BC.ann
Global Average Yaw                              (deg) = -1.90762926
```

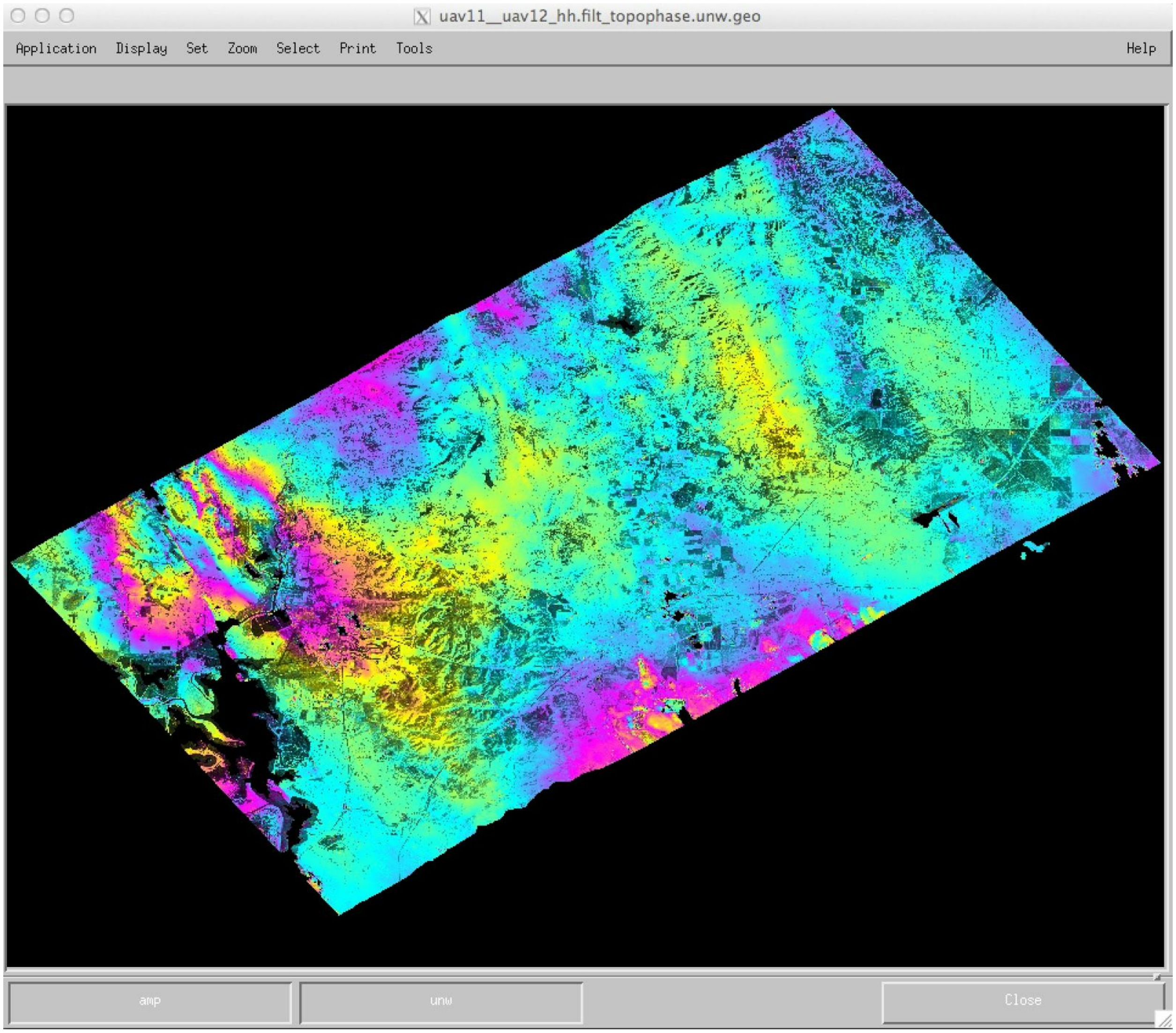
You can use `mdx.py` to visualize any of the output images. For example, the filtered, unwrapped, topophase corrected, geocoded interferogram for a pair before the earthquake,

```
> mdx.py uav10__uav11/uav10__uav11_hh.filt_topophase.unw.geo -z -2
```

Note that some of the phase in this interferogram before the earthquake (waves running across the swath) is likely due to aircraft motion that was not fully corrected, because this stack was processed without the residual baseline correction.

Similarly for a pair spanning the earthquake,

```
> mdx.py uav11__uav12/uav11__uav12_hh.filt_topophase.unw.geo -z -2
```



The August 24, 2014 M6.0 South Napa Earthquake caused the strong fringes and the sharp phase discontinuity near the west end of this UAVSAR interferogram where the fault ruptured the surface. The default phase unwrapping method was not able to estimate the large offset across the surface rupture, which is about 20 cm or nearly two fringes.

6. Notes about component configurability

Lab 7 illustrated a few of the component configurability options for controlling the processing of COSMO-SkyMed data. In the current lab we did not tell you earlier but we have used a feature of component configurability to provide additional input information to the processsing job. We have set certain global preferences for processing the UAVSAR data with `isceApp.py` so that the input `isceApp.xml` file we showed you in Section 3 was as simple and as specific to the current data as possible. ISCE uses an environment variable $ISCEDB to locate these global preferences. If you type the following commands you will see the name and contents of the directory that ISCE looks in for these global preferences,

```
> echo $ISCEDB
/home/ubuntu/.iscedb

>ls $ISCEDB
insar.xml  isce.xml
```

When you run an application the ISCE framework looks in this directory to see if any file is named appropriately for configuring an ISCE component or applications. The `isceApp.py` application can be configured with a file named either `isce.xml` or `isceApp.xml` (or both with the one named `isceApp.xml` having higher priority if there are conflicting information for any properties contatined in the two files). When configuring isceApp.py the global preferences can be overridden by settings in the input files in the processing directory or on the cammand line.

Let's look at the contents of the global preferences file for `isceApp.py`,

```
> cat ~/.iscedb/isce.xml
<?xml version="1.0" encoding="UTF-8"?>

<!-- NOTE: tag/attribute names must be in lower case -->
<isceApp>
  <component name="isce">
     <component name="stack">
     <component name="Raster1">
         <property name="ncol">1500</property>
         <property name="nlin">4000</property>
         <property name="datatype">float</property>
     </component>
     </component>
     <property name="doppler method">usedefault</property>
     <!-- Processors to run: True/False -->
```

```
    <property name="do preprocess">True</property>
    <property name="do verifyDEM">True</property>
    <property name="do pulsetiming">True</property>
    <property name="do estimateheights">True</property>
    <property name="do mocomppath">True</property>
    <property name="do orbit2sch">True</property>
    <property name="do updatepreprocinfo">True</property>
    <property name="do formslc">True</property>
    <property name="do multilookslc">True</property>
    <property name="do filterslc">False</property>
    <property name="do filter interferogram">True</property>
    <property name="do polarimetric correction">False</property>
    <property name="do calculate FR">False</property>
    <property name="do FR to TEC">False</property>
    <property name="do TEC to phase">False</property>
    <property name="do offsetprf">False</property>-->
    <property name="do outliers1">False</property>-->
    <property name="do prepareresamps">True</property>-->
    <property name="do resamp">False</property>-->
    <property name="do resamp image">False</property>-->
    <property name="do crossmul">True</property>
    <property name="do mocomp baseline">True</property>
    <property name="do set topoint1">True</property>
    <property name="do topo">True</property>
    <property name="do shadecpx2rg">True</property>
    <property name="do rgoffset">False</property>
    <property name="do rg outliers2">True</property>
    <property name="do resamp only">True</property>
    <property name="do set topoint2">True</property>
    <property name="do correct">True</property>
    <property name="do coherence">True</property>
    <property name="do unwrap">False</property>
    <property name="do geocode">True</property>
  </component>
</isceApp>
```

Most of the contents of this file tell isceApp.py whether to do a particular step in its flow. The False settings in this file are not really optional for processing UAVSAR stack data. The isceApp.py application is general enough to work with many different types of sensors, so it is necessary that it knows which steps to do for this particular sensor. The UAVSAR stack processing is relatively new and in future updates to ISCE would automatically set these options as the defaults when processing data from this sensor.