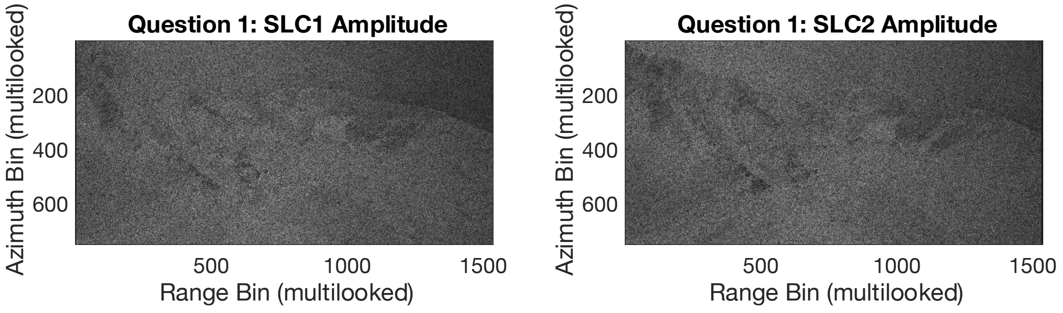
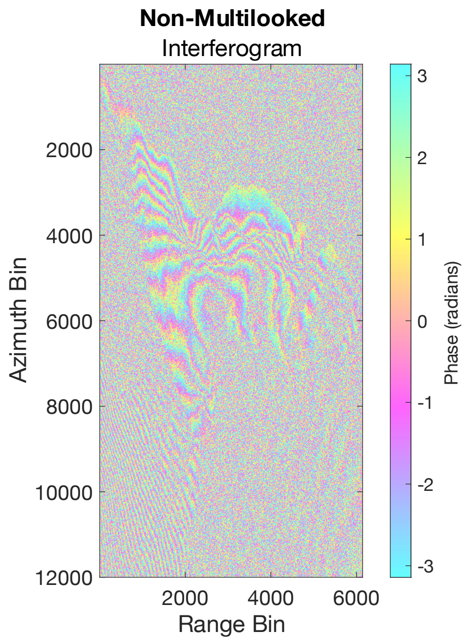
JPL/NISAR Radar Course
HW9 Solutions
November 3, 2023

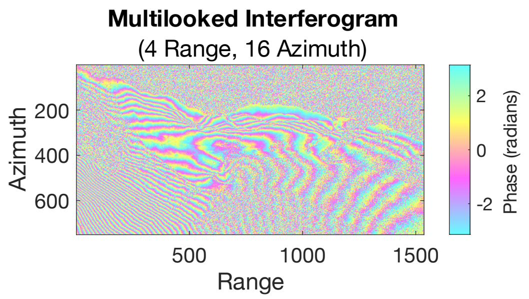
Problem 1

Download files slc1.dat and slc2.dat from class web area under homework 9. These images have been processed using motion compensation processor that removes the flat-Earth term automatically, and produces co-registered images. Cross multiply the two images and form the interferogram, take 4 looks in range and 16 in azimuth, and submit an image. Your interferogram should resembles topography of Hawaii with the flat (curved) Earth term removed.


Question 1: SLC1 Amplitude
Question 1: SLC2 Amplitude
Azimuth Bin (multilooked)
200
400
600
500
1000
1500
Range Bin (multilooked)

The amplitudes of the single-look complex images look very similar. We display multilooked SLCs here. You can make out some features – a coastline running roughly horizontal through the image, and some of the land features below it. The images are very similar but not identical, which we expect: they are in the same coordinate system from two different acquisition times, and the surface has changed some in that time.





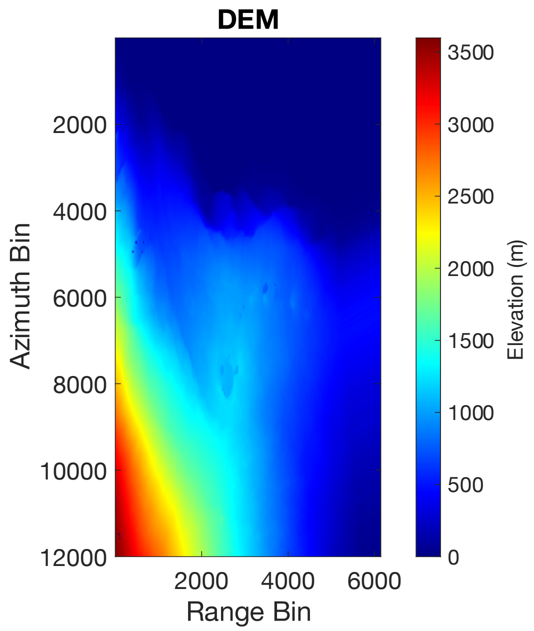
We display the full-resolution and multi-looked interferograms here. While both contain some noise, the phase signal is clearer to pick out in the right image after multilooking (spatial filtering).

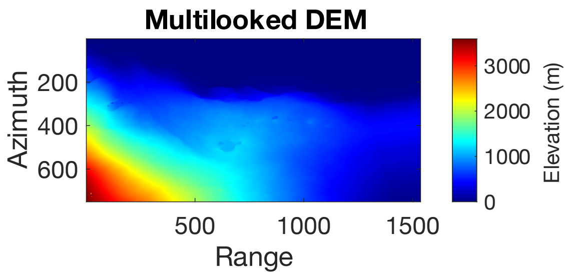
This interferogram contains two types of phase signals that we're identifying here: one is due to topography. Right now, we're assuming all the rays hit a smooth earth. In reality, some of them are travelling the distance to the top of the topography rather than to the smooth earth surface (a.k.a. approximately sea level). This signal will appear to be correlated with topography, and there will be dense fringes around steeper slopes.

The other signal contained here is the deformation, the result of the land surface changing due to e.g. a volcanic eruption. Right now, it's hard to pick it out because the signal due to topography is so strong.

The dimensions of the image after taking 4 range looks and 16 azimuth looks should be 1536 range bins and 750 azimuth bins.

Problem 2





The DEM file displays the elevation in meters.

The area of 0 elevation is the ocean in dark blue.

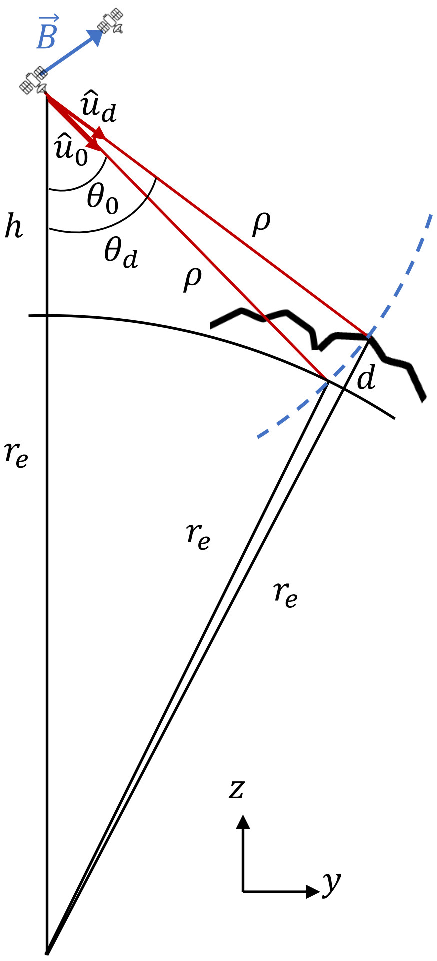
The area in bright red is the corner of Mauna Kea.

We multilook the DEM to match the size of the original image.

Problem 3
Compute the topographic phase term as follows. Referring to the figure above, calculate the unit look vectors to both the elevated point at each pixel location and also to the same pixel location but at zero elevation. (Hint: you will find it most accurate to use the curved- Earth geometry in your construction). The two unit vectors will be almost the same, but slightly different. Simulate the radar phase measurement by deriving the component of the baseline vector for each line in the line-of-sight direction to each pixel using the two unit vectors you just created. You will have two phases, one for the elevated pixel and one for the pixel at ground level. The difference of these is the topographic phase. Submit an image of the simulated topo phase.

We want to calculate the unit vectors $\hat{u}_0$ and $\hat{u}_d$ along the smooth-earth and DEM-earth look vectors. First, we want to calculate the set of ranges ρ in each of our range bins. We know the slant range bin spacing from our previous homework involving range: $\Delta_r = \frac{c}{2f_s}$. Then,

$$\rho = r_0 + (n-1)\Delta_r \ , \quad n = 0, 1, \dots nrange - 1$$



We calculate the angles θ_0, θ_d from the Law of Cosines. We do this for the inner triangle connecting the satellite, the smooth Earth, and the center of the Earth; and for the outer triangle connecting the satellite, the Earth surface with topography, and the center of the Earth.

$$\cos\theta_0 = \frac{\rho^2 + (r_e + h)^2 - r_e^2}{2\,(\rho)(r_e + h)}$$
$$\cos\theta_d = \frac{\rho^2 + (r_e + h)^2 - (r_e + d)^2}{2\,(\rho)(r_e + h)}$$

Next, we want to find the y and z components of ρ.

We can see that to project the directions of our look vectors $\vec{\rho}$ into the $y - z$ coordinate system, it's a simple matter of trigonometry using our angles θ_0, θ_d

Then our unit vector components are as follows:

$$\hat{u}_{0,y} = \sin\theta_0$$
$$\hat{u}_{0,z} = \cos\theta_0$$
$$\hat{u}_{d,y} = \sin\theta_d$$
$$\hat{u}_{d,z} = \cos\theta_d$$

Next, we want to project the baseline vectors onto the unit vectors. We have read in the baselines file: the first column is the number of lines, the second is the baseline component in the y direction B_y, and the third is the baseline component in the z direction B_z.
We want

$$\vec{B}_0 = \vec{B} \cdot \hat{u}_0 = B_y \hat{u}_{0,y} + B_z \hat{u}_{0,z}$$
$$\vec{B}_d = \vec{B} \cdot \hat{u}_d = B_y \hat{u}_{d,y} + B_z \hat{u}_{d,z}$$

Then we subtract the projected baseline vectors and convert to phase. We first find the phase that we'll put into our complex exponential. This is correct modulo 2pi but could be >2pi:

$$\phi_{topo,unbounded} = -\frac{4\pi}{\lambda}(\vec{B}_d - \vec{B}_0)$$
$$signal_{topo} = \exp(-j\phi_{topo,unbounded})$$

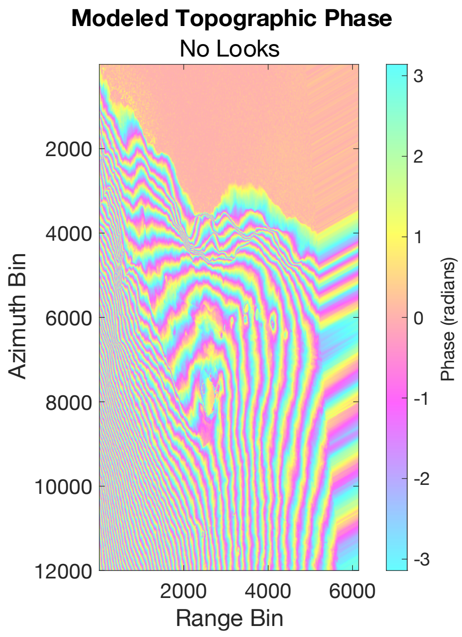
We put the phase of the signal in the exponent, then

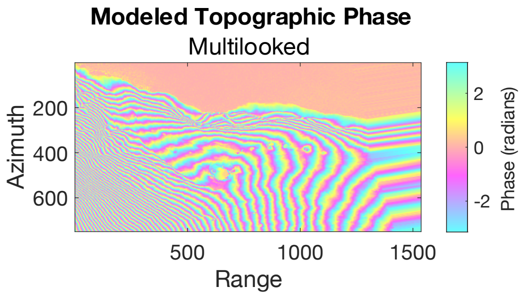
$$\phi_{topo} = angle(signal_{topo}) = angle\left(\exp\left(j \cdot \frac{4\pi}{\lambda}(\vec{B}_d - \vec{B}_0)\right)\right)$$

I use the MATLAB/Python angle function to find the phase from the complex exponential. This is also equivalent to:

$$angle(x) = \text{atan}\left(\frac{Im(x)}{Re(x)}\right)$$

We find the topographic phase:





How to tell if this is correct?

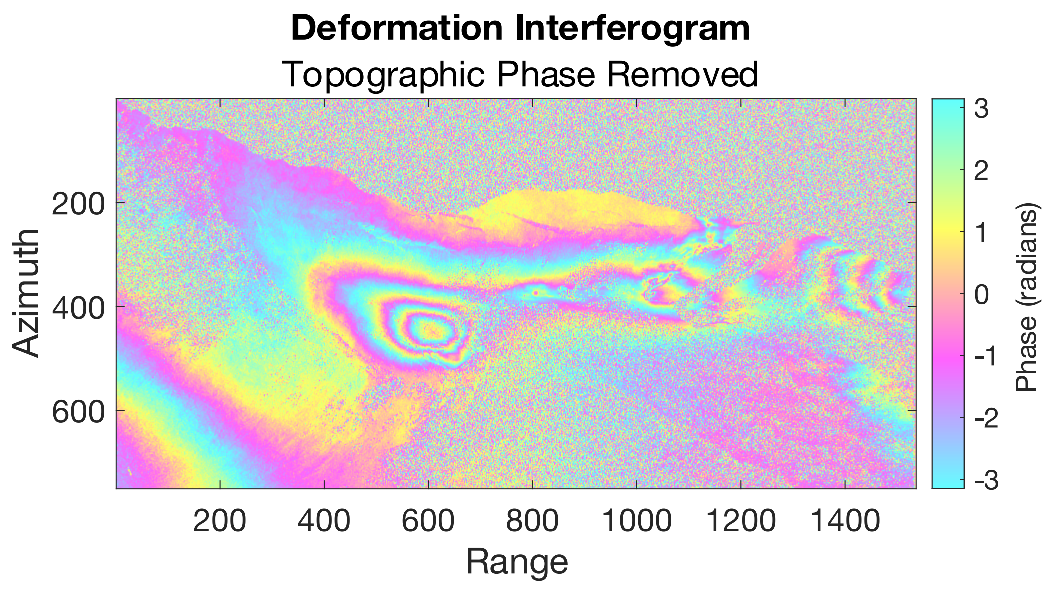
The lines should follow the topography of the DEM, and be closer together where the DEM is changing more rapidly and more spaced out over relatively unchanging (flat) parts of the DEM.

Problem 4
Using the simulated phase, correct the phase of each point in the interferogram for topography. This should yield a map of the deformation. Submit this image.

To compute the deformation map, we take the whole interferogram "int" and multiply it by the topographic phase correction, which I've been calling $signal_{topo}$.

I opted to do the corrections on the multilooked interferogram, so I have my interferogram ml_int and my multilooked topo signal ml_signal$_{topo}$. Then the deformation interferogram ml_defo_int is:

$$ml_{defo_{int}} = ml\_int \ * conj(ml\_signal_{topo})$$
$$ml_{defo_{int}} = ml\_int \ \exp\left(-j\ \phi_{topo}\right)^{*}$$



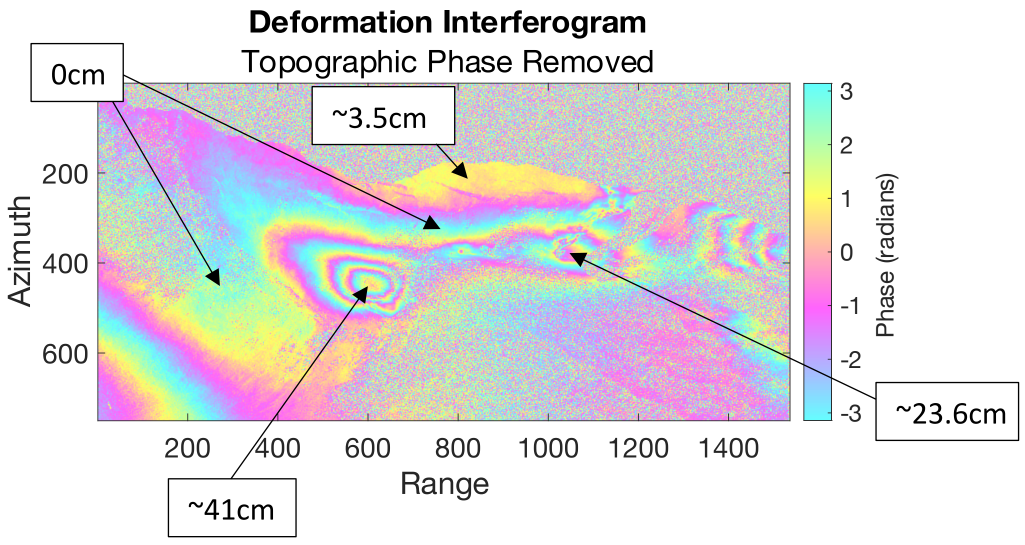
Problem 5

Now examine the deformation signature. Using the measured phases estimate the size of the deformation at several points where significant deformation is occurring. How much deformation in cm is occurring over the time of this interferogram? Is it related to features seen in the image?

The largest deformation signal is centered at (range, azimuth) = (600,400) in the multilooked image, with fringes around it in concentric contours. The best we can do is decide on a relative deformation magnitude at this point, relative to a point we assume has zero displacement. I decide on that point in the following image. This is a wide fringe, and in addition, it has symmetry traveling in the northeast to southwest direction, implying it may be some kind of minimum. From here, there are about 3-3.5 phase cycles to the point of maximum deformation. Each phase cycle represents 2π of phase change, which equals $\lambda/_2$ of deformation in the radar line-of-sight (LOS):

$$|\phi_{def}| = 2\pi = \frac{4\pi}{\lambda}\delta\rho v \rightarrow \ \delta\rho = \frac{\lambda}{2}$$

Thus, the maximum deformation that occurred between our two scene acquisitions is about 41 cm in the LOS direction, relative my zero-deformation point. In order to know if the ground moved up or down, we would need to be told whether the second slc was before or after the first slc. I have also identified a few other deformation magnitudes in the image below. Based on what we know about this area (a volcano in Hawaii), we can expect that the deformation is due to subsurface activity of the volcano. These points of deformation correspond well with what look like lava flows in the amplitude image. But note that the deformation is likely due to current subsurface pressure/stress changes (from magma activity), rather than "displacements" by the addition of surface material from a lava flow (which would likely cause decorrelation). Pretty cool stuff!



MATLAB code:

```matlab
clc; clearvars; close all; % Clear old variables
nr = 6144; % Number of range bins
naz = 12000; % Number of lines in azimuth

%% Howard's colormap
r = zeros(360,1);
g = r;
b = r;
for i=1:120
    r(i)=i*2.13*155/255+100;
    g(i)=(120-i)*2.13*155/255+100;
    b(i)=255;
end
for i=121:240
    ival=i-121;
    r(i)=255;
    g(i)=ival*2.13*155/255+100;
    b(i)=(239-i)*2.13*155/255+100;
end
for i=241:360
    ival=i-241;
    r(i)=(359-i)*2.13*155/255+100;
    g(i)=255;
    b(i)=ival*2.13*155/255+100;
end
r(r>255)=255;
g(g>255)=255;
b(b>255)=255;
map = [r/255 g/255 b/255];
% figure; image(1:360) % Optional code to display range of colors
% colormap(map)
% colorbar
%%

% open slc1 image file and create complex image
fid = fopen('slc1.dat','r');
slc1 = fread(fid,[nr*2, naz],'float');
fclose(fid);
slc1 = slc1(1:2:end,:) + 1i*slc1(2:2:end,:);

% open slc2 image file and create complex image
fid = fopen('slc2.dat','r');
slc2 = fread(fid,[nr*2, naz],'float');
fclose(fid);
slc2 = slc2(1:2:end,:) + 1i*slc2(2:2:end,:);

% Take looks to display SLCs
aslc1 = sqrt(multilook16x4(slc1,nr,naz));
aslc2 = sqrt(multilook16x4(slc2,nr,naz));

% Display the two SLCs
figure
subplot(1,2,1);
imagesc(abs(aslc1'));
```

```
set(gca,'fontsize',16);
title('Question 1: SLC1 Amplitude');
xlabel('Range Bin (multilooked)');
ylabel('Azimuth Bin (multilooked)');
colormap(gray);
caxis([0 mean(mean(abs(aslc1)))*3]);
axis image
subplot(1,2,2)
imagesc(abs(aslc2'));
set(gca,'fontsize',16);
title('Question 1: SLC2 Amplitude');
xlabel('Range Bin (multilooked)');
ylabel('Azimuth Bin (multilooked)');
colormap(gray)
caxis([0 mean(mean(abs(aslc2)))*3]);
axis image

% Form the interferogram
interferogram = slc1.*conj(slc2);

% Take Looks in the interferogram
az_looks = 16;
r_looks = 4;
ml_int = multilook16x4(interferogram,nr,naz);


figure;
subplot(1,2,1);
imagesc(angle(interferogram).');
set(gca,'fontsize',16)
title({'Non-Multilooked'},{'Interferogram'});
xlabel('Range Bin');
ylabel('Azimuth Bin');
colormap(map)
hC = colorbar;
ylabel(hC,'Phase (radians)')
axis image;
subplot(1,2,2)
imagesc(angle(ml_int).')
set(gca,'fontsize',16)
title({'Multilooked Interferogram'},{'(8 Range, 16 Azimuth)'});
xlabel('Range');
ylabel('Azimuth');
colormap(map)
hC = colorbar;
axis image
ylabel(hC,'Phase (radians)')


% Optional line: clears the full-res interferogram and SLCs
% to save space in your computer memory if needed
clear interferogram slc1 slc2

%% Load and multilook DEM
% open the DEM file and take looks to match interferogram
```

```
fid = fopen('slc.dem','r');
dem = fread(fid,[nr,naz],'float');
fclose(fid);

% Multilook the DEM
ml_dem = multilook16x4(dem,nr, naz);

% Plot the full-res and multilooked DEM
figure
subplot(1,2,1)
imagesc(dem.');
set(gca,'fontsize',16)
title('DEM');
xlabel('Range Bin');
ylabel('Azimuth Bin');
colormap(jet);
hC = colorbar;
ylabel(hC,'Elevation (m)');
axis image;
subplot(1,2,2);
imagesc(ml_dem.');
set(gca,'fontsize',16)
title('Multilooked DEM');
xlabel('Range');
ylabel('Azimuth');
colormap(jet);
h = colorbar;
axis image;
ylabel(h,'Elevation (m)')

% open the baseline file - we can use load to load a .txt file.
baselines = load('slc.baseline');
% Get the components on the Baselines for each pixel
By = baselines(:,2)';
Bz = baselines(:,3)';


% Radar Parameters
r_e = 6343837.1345648393; % radius of earth (m)
h = 700000; % altitude of the satellite (m)
r0 = 741489; % range to first pixel (m)
fs = 32e6; % range sampling frequency (Hz)
lambda = 0.236057; % wavelength of system (m)
c = 2.99792458e8; % speed of light (m/s)
Delta_r_bin_slant = c/(2*fs);

% Unit Vectors for each each pixel (smooth earth points)
% Each subsequent range bin is Delta_r further in slant range
rho = zeros(nr,1);
for n=1:nr
    rho(n) = r0 + (n-1)*Delta_r_bin_slant;
end

% Find theta_0 and theta_d from law of cosines
theta0  = acos(((r_e+h)^2 + rho.^2 - r_e^2)./(2*(r_e+h)*rho));
```

```
thetad = acos(((r_e+h)^2 + rho.^2 - (r_e+dem).^2)./(2*(r_e+h)*rho));
% Use trigonometry to find unit vectors along rho
u0_y = sin(theta0);
u0_z = cos(theta0);
ud_y = sin(thetad);
ud_z = cos(thetad);
% Project baselines onto unit vectors at each pixel
proj_B0 = (By.*u0_y + Bz.*u0_z);
proj_Bd = (By.*ud_y + Bz.*ud_z);

% Calculate topographic phase component from projected baselines
% Topographic phase is found from the difference between the lengths
% of the assumed (no-topography) and real paths
% This phase will not be wrapped between 0 and 2*pi
topo_phase = (proj_Bd-proj_B0)*-4*pi/lambda;

% Find the proportion of the signal due to topographic delays
% This appears as a phase change.
topo_signal = exp(-1i*topo_phase);

% Multilook the topographic phase signal
ml_topo_signal = multilook16x4(topo_signal,nr,naz);

% Display the topographic phase
figure
subplot(1,2,1)
imagesc(angle(topo_signal).');
set(gca,'fontsize',16);
title({'Modeled Topographic Phase'},{'No Looks'});
xlabel('Range Bin');
ylabel('Azimuth Bin');
colormap(map)
hC = colorbar;
ylabel(hC,'Phase (radians)');
axis image;
subplot(1,2,2)
imagesc(angle(ml_topo_signal).');
set(gca,'fontsize',16);
title({'Modeled Topographic Phase'},{'Multilooked'});
xlabel('Range');
ylabel('Azimuth');
colormap(map);
hC = colorbar;
ylabel(hC,'Phase (radians)');
axis image

% Optional: clear the full-res topo signal to save computer space
clear topo_signal;

% Adjust the deformation phase by the topo phase
deformation_phase = ml_int .* conj(ml_topo_signal);

% Show the interferogram of deformation
figure;
imagesc(angle(deformation_phase).');
```

```
set(gca,'fontsize',16);
title({'Deformation Interferogram'},{'Topographic Phase Removed'});
xlabel('Range');
ylabel('Azimuth');
colormap(map);
hC = colorbar;
ylabel(hC,'Phase (radians)');
axis image


%% Function to multilook image (16 looks in range, 4 in azimuth)
function ml_img = multilook16x4(image, width, nlines)
    img_temp=reshape(sum(reshape(image,4,width/4,nlines)),width/4,nlines);

ml_img=reshape(sum(reshape(img_temp',16,nlines/16,width/4)),nlines/16,width
/4)/4/16;
    ml_img = ml_img';
end
```

JPL-NISAR Radar Short Course HW10

Elizabeth Wig November 1, 2023

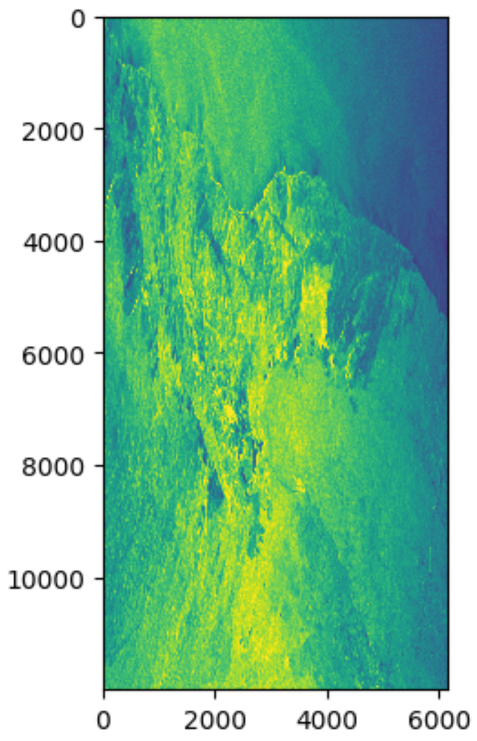
In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```python
# Howard's function to display visually appealing interferograms
def dismph(image, nr, naz): # Function takes complex image, size in range and azimuth
    upramp = np.linspace(100,255,120)/255
    dnramp = np.linspace(255,100,120)/255
    oneramp = np.ones(120)
    zeroramp = np.zeros(120)
    r = np.zeros(360)
    g = np.zeros(360)
    b = np.zeros(360)
    r[0:120]=upramp; r[120:240]=oneramp; r[240:360]=dnramp
    g[0:120]=dnramp; g[120:240]=upramp; g[240:360]=oneramp
    b[0:120]=oneramp; b[120:240]=dnramp; b[240:360]=upramp

    fig = plt.figure(figsize=(10, 10))
    phase = np.angle(image)*180./np.pi

    exponent=0.3
    scale = 1
    mag=np.power(np.absolute(image),exponent)
    ampsum=sum(sum(mag))
    ampi=naz*nr
    ampi=np.count_nonzero(mag)
    scalemag = (scale*150/(ampsum/ampi))/256

    mag = np.clip(mag*scalemag,0,1)

    icolor = phase.astype(int)
    icolor = np.where(icolor < 0, icolor+360, icolor)
    icolor = icolor.clip(max=359)

    # load color table values for image
    rgb = np.zeros((nr,naz,3))
    rgb[:,:,0] = r[icolor[:,:]]*mag
    rgb[:,:,1] = g[icolor[:,:]]*mag
    rgb[:,:,2] = b[icolor[:,:]]*mag

    plt.imshow(rgb)
```

In [3]:
```python
def multilook16x4(image,width, nlines): # Function to take 16 x 4 looks
    # There are many ways to do this: you can convolve with a rectangle and downsample
    # You can say for, e.g. two looks, img_ml_2 = (img[0::2] + img[1::2])/2
    # This method reshapes the image so that all the pixels within a multilook window
    # are elements in a 3rd dimension, and then finds the mean across that dimension.
    img_temp = np.sum(np.reshape(image, ( width//4,4, nlines)), axis=1)
    ml_img = np.sum(np.reshape(img_temp.T, (nlines//16,16, width//4)), axis=1)
    ml_img = ml_img.T /4/16
    return ml_img
```

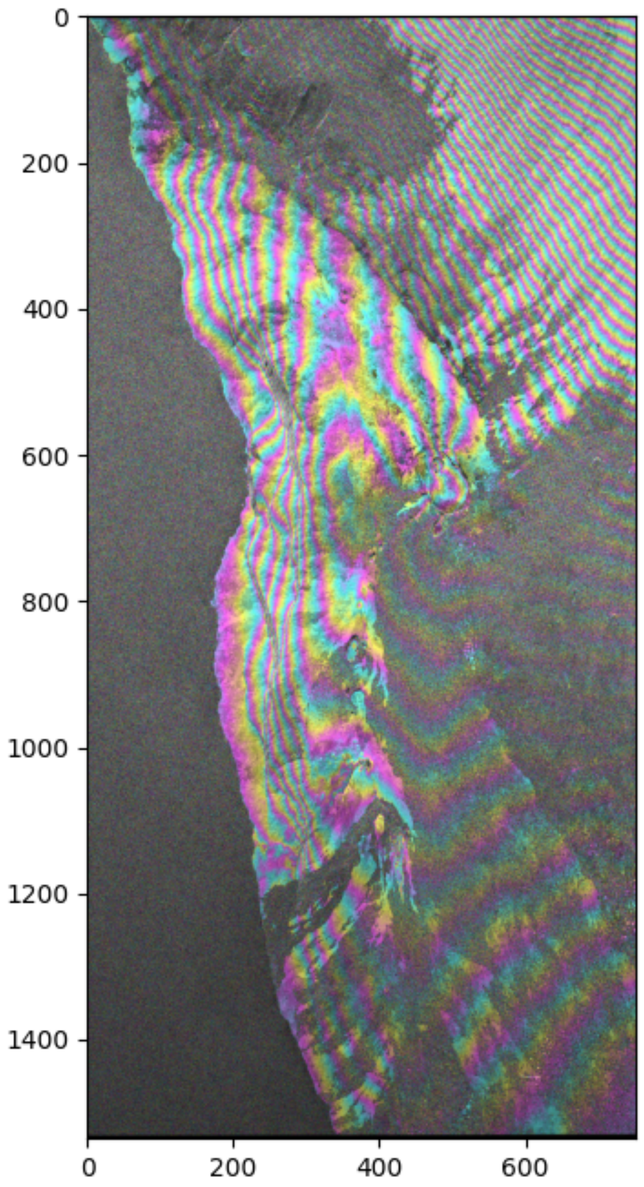
P1a: Read in the files and display interferogram

In [4]:
```
filepath = "/Users/elizabethwig/Library/CloudStorage/OneDrive-Stanford/Stanford_NonResearch/JPL-NISAR Radar Course/Day 09
slc1path = filepath+"/slc1.dat"
slc2path = filepath+"/slc2.dat"

nr = 6144       # Number of range bins
naz = 12000     # Number of azimuth bins

# file = open(slc1path,"rb")
slc1cpx = np.fromfile(slc1path,dtype=np.float32) # Read in file as float32
slc1vec = slc1cpx[0::2] + 1j * slc1cpx[1::2]     # Convert to complex numbers
slc1 = np.reshape(slc1vec,(naz,nr))              # Reshape to proper size
slc2cpx = np.fromfile(slc2path,dtype=np.float32) # Read in file as float32
slc2vec = slc2cpx[0::2] + 1j * slc2cpx[1::2]     # Convert to complex numbers
slc2 = np.reshape(slc2vec,(naz,nr))              # Reshape to proper size

plt.imshow(np.abs(slc1)) # Display single-look complex image
plt.clim(0,300) # Color limits so image shows up clearly
```



In [5]:
```
igram = (slc1 * np.conj(slc2)).T # Cross multiply the single-look complex to get interferogram
igram_ml = multilook16x4(igram,nr,naz) # Multilook interferogram
```

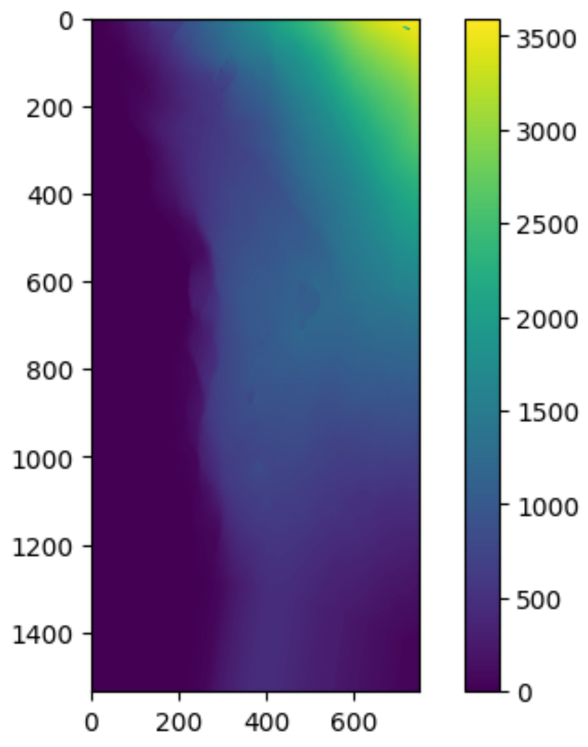
```
In [13]: dismph(igram_ml, nr//4, naz//16)  # Use Howard's code to display multilooked interferogram
```



P2: Load DEM and Baseline files

```
In [7]: demfile = filepath+"/slc.dem"
        dem = np.fromfile(demfile,dtype=np.float32)  # Read DEM. Elevation data is not complex
        dem = np.reshape(dem,(naz,nr)).T   # Reshape DEM to be size of image

        dem_ml = multilook16x4(dem,nr,naz) # Multilook DEM

        plt.imshow(dem_ml)
        plt.colorbar()
```

```
Out[7]: <matplotlib.colorbar.Colorbar at 0x7fe082352b50>
```

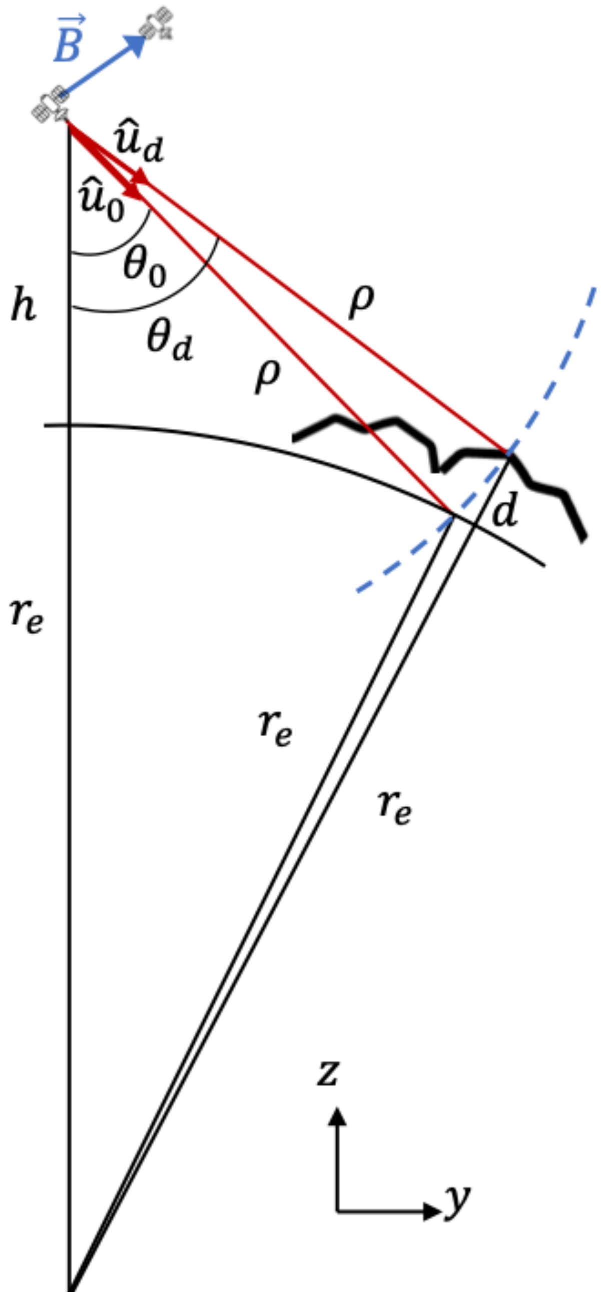


```
In [8]: baselinefile = filepath+"/slc.baseline"
        baselines = np.loadtxt(baselinefile)     # Load baseline file
        By = baselines[:,1]     # Find baseline y component
        Bz = baselines[:,2]     # Find baseline z component
```

P3: Calculate Difference in Path Length from Topography

Use geometry from the

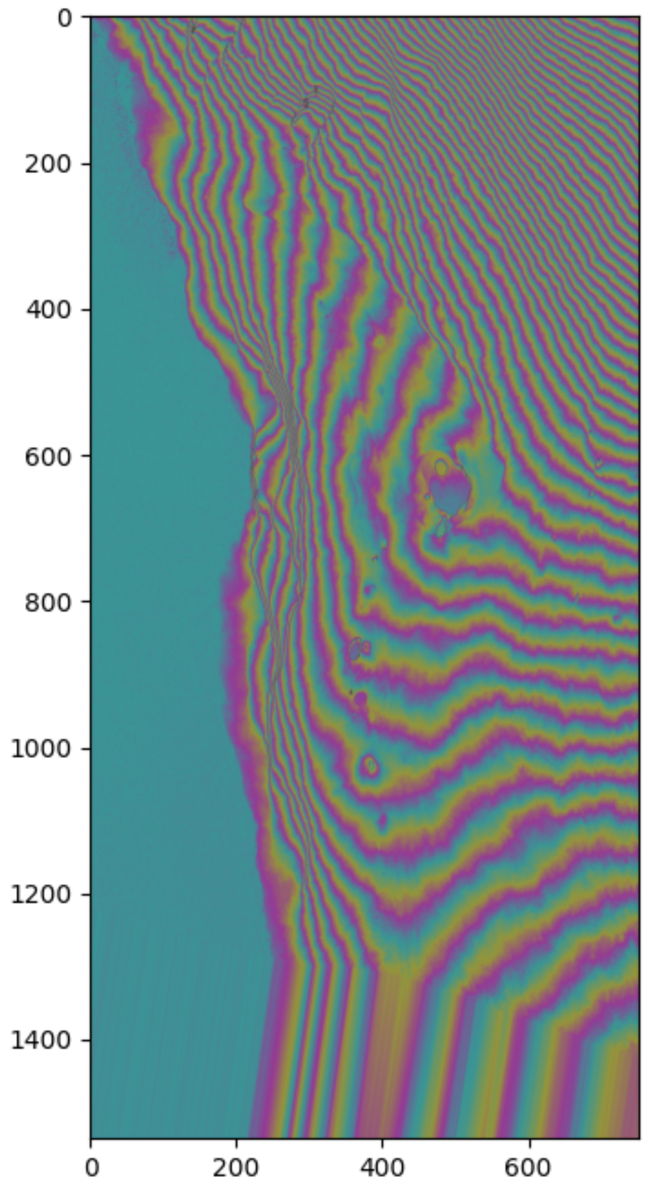
following image:


$\vec{B}$
$\hat{u}_d$
$\hat{u}_0$
θ_0
h
ρ
θ_d
ρ
d
r_e
r_e
r_e
z
y

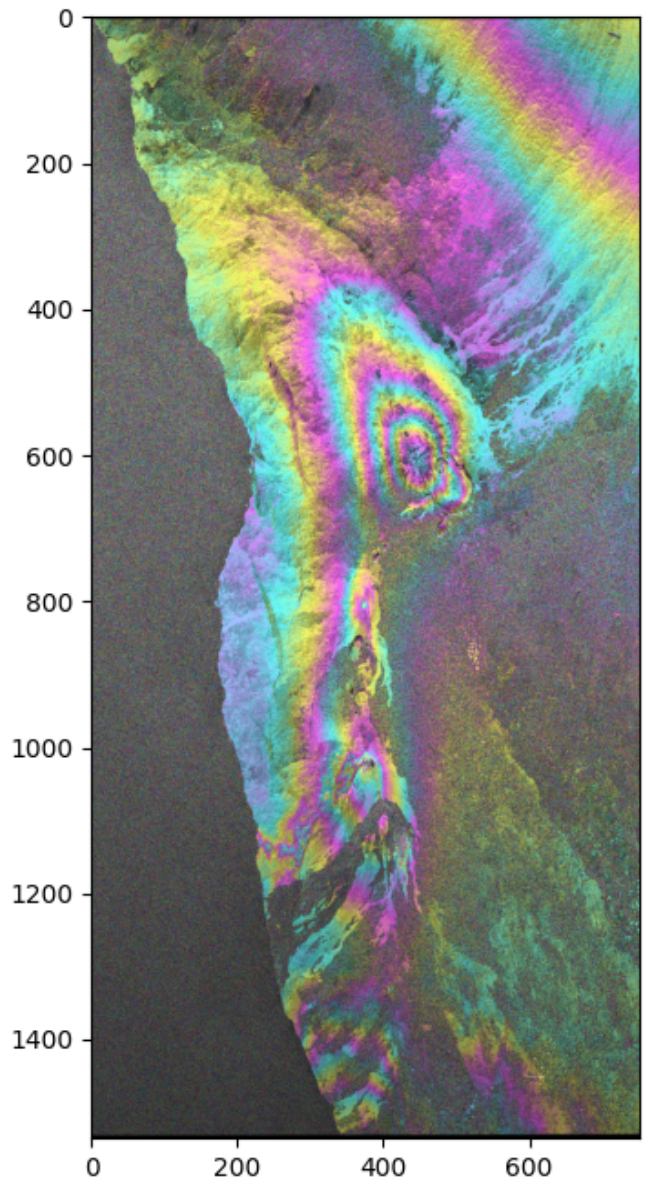
In [9]:
```
# Define radar parameters

r_e = 6343837.1345648393 # radius of earth (m)
h = 700000 # altitude of the satellite (m)
r0 = 741489 # range to first pixel (m)
fs = 32e6 # range sampling frequency (Hz)
lam = 0.236057 # wavelength of system (m)
c = 2.99792458e8 # speed of light (m/s)
Delta_r_bin_slant = c/(2*fs) # Slant range bin spacing
```

In [10]:
```
# First, find ranges rho
rho = np.arange(r0, r0+(nr-1)*Delta_r_bin_slant, Delta_r_bin_slant)
rho = np.expand_dims(rho, axis=-1) # Expand dimensions of rho so it can broadcast with 2D DEM
# Then, project the baseline vector in the direction of the look angle
theta0  = np.arccos(((r_e+h)**2 + rho**2 - r_e**2)/(2*(r_e+h)*rho));
thetad = np.arccos(((r_e+h)**2 + rho**2 - (r_e+dem)**2)/(2*(r_e+h)*rho));
# Use trigonometry to find unit vectors along rho
u0_y = np.sin(theta0);
u0_z = np.cos(theta0);
ud_y = np.sin(thetad);
ud_z = np.cos(thetad);
# Project baselines onto unit vectors at each pixel
proj_B0 = (By*u0_y + Bz*u0_z);
proj_Bd = (By*ud_y + Bz*ud_z);
```

In [11]:
```
# Calculate topo phase. This is accurate, though not constrained between 0 and 2pi
# So you can think of it as "not wrapped"
topo_phase = (proj_Bd-proj_B0)*-4*np.pi/lam;
# Find the signal used to correct for the topo phase
topo_signal = np.exp(-1j*topo_phase)
# Multilook the topo signal
ml_topo_signal = multilook16x4(topo_signal,nr,naz);

# Display the phase due to the topography
dismph(ml_topo_signal,nr//4,naz//16)
```



In [12]:
```
# Calculate topo-corrected interferogram -- phase only from deformation
deformation_int = igram_ml * np.conj(ml_topo_signal);

# Display phase due to deformation
dismph(deformation_int, nr//4, naz//16)
```



0
200
400
600
800
1000
1200
1400
0
200
400
600