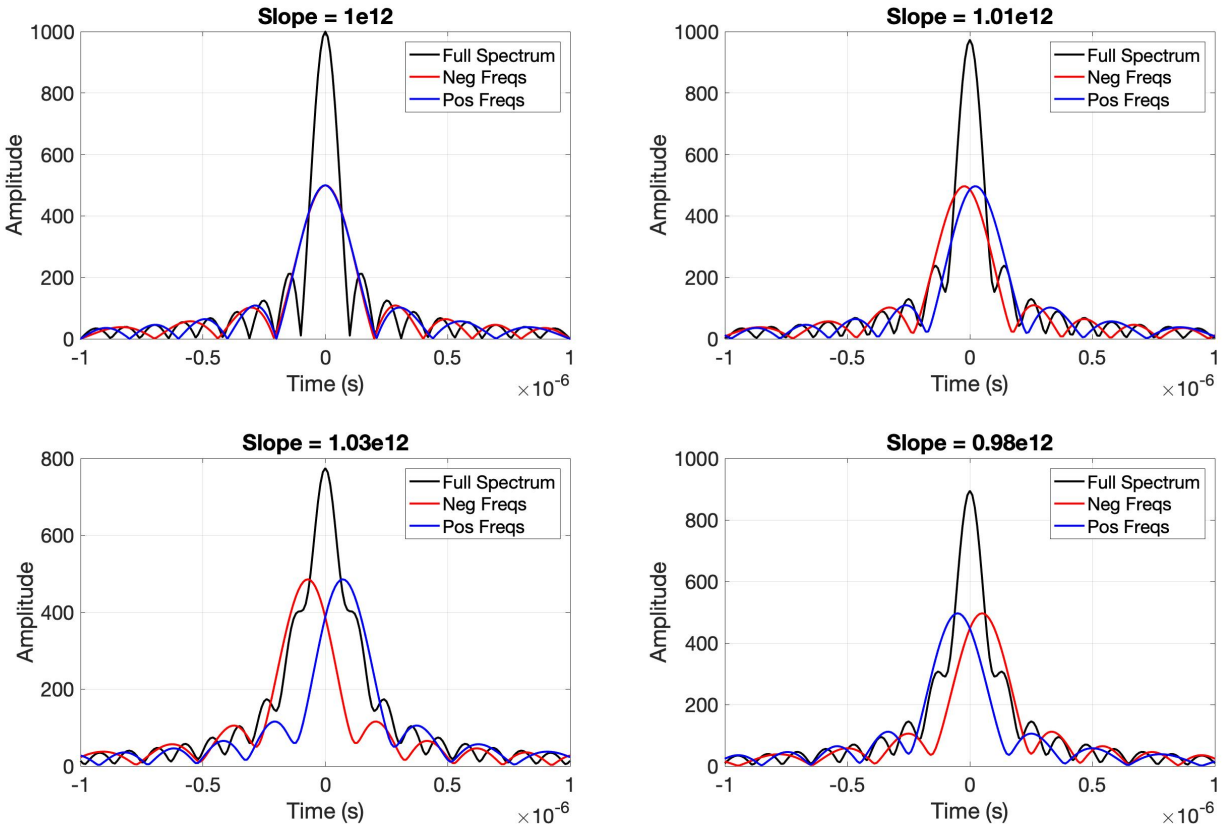
Imaging Radar HW5 Solutions
Winter 2020

Question 1: Write an autofocus program to implement the sub-aperture shift algorithm.

Correlate the chirp with reference chirps of slope 1e12, 1.01e12, 1.03e12, and 0.98e12 Hz/s. In each case analyze the resulting complex image and calculate

a) The offset in pixels of the subaperture images
b) The implied delta s for each case



a) For part (a), we compress the original signal, a chirp with slope of s = 10^{12} Hz/s, by correlating it with the reference chirp twice. The first reference chirp of slope s' consists of positive frequencies, half of the original bandwidth, and the second with the second subaperture consisting of only the negative frequencies (i.e. the other half of the bandwidth).

The compressed signals from the two subapertures will be slightly offset from each other, and this offset will determine Δt. We can find the offset by cross-correlating the amplitudes of the compressed signals to find the pixel offset Δp.

b) Once we have the shift in pixels, we can convert to the time shift Δt by dividing by the sample rate f_s:

$$\Delta t = \frac{\Delta p}{f_s}$$

Then, with the time shift we can compute the chirp slope correction Δs, which should match the expected slope correction (which we know because we know the true slope).

$$\Delta s = \frac{2s^2\Delta t}{BW} = \frac{2s^2\Delta t}{s\tau} = \frac{2s\Delta t}{\tau}$$

I get the following:

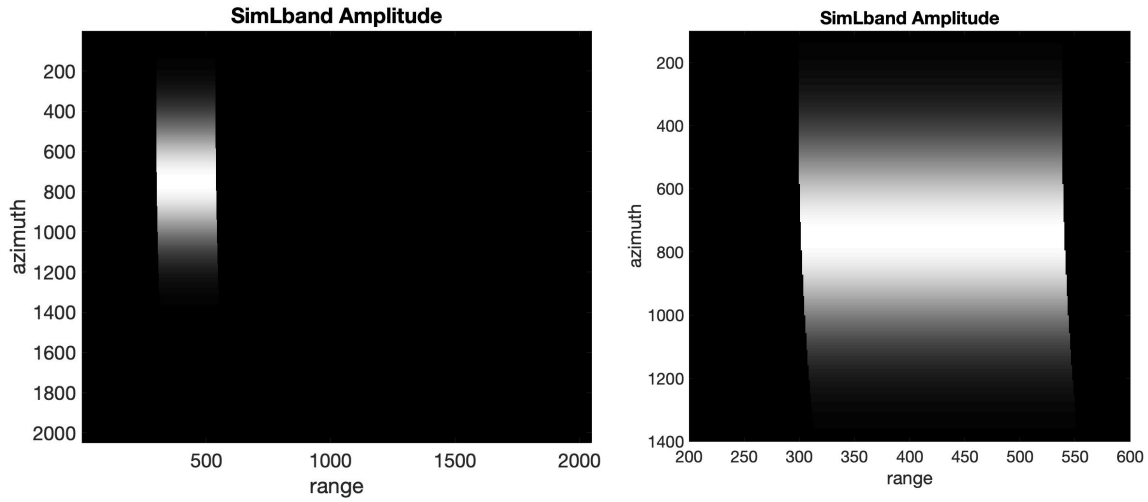
s' (Hz/s)	*(a) Δp*	*Δt (s)*	*(b) Δs (Hz/s)*
1e12	0	0	0
1.01e12	-5	-0.05e-6	-0.01e12
1.03e12	-15	-0.15e-6	-0.03e12
0.98e12	10	0.10e-6	0.02e12

The pixel offset will likely not be the exact offset we are expecting due to discretization, but even with rounding we do a pretty good job.

```
%% Question 1
% Write an autofocus program to implement the sub-aperture algorithm

clc; clearvars; close all;

% Chirp Parameters
s = 1e12; % chirp slope
tau = 10*10^-6; % pulse length
fs = 100*10^6; % sample rate
num = 2048; % number of samples

chp = makechirp(s,tau,fs,0,1,num); % Signal

chp_fft = fft(chp); % fft of signal chirp

s = [1e12 1.01e12 1.03e12 0.98e12];
strslp = {'1e12', '1.01e12', '1.03e12', '0.98e12'};

% Make the reference chirps
figure
t=1/fs*(-num/2:num/2-1);
for k = 1:length(s)
    ref1 = makechirp(s(k),tau/2,fs,-s(k)*tau/4,1,num);
    ref2 = makechirp(s(k),tau/2,fs,s(k)*tau/4,round(tau/2*fs)+1,num);
    refWH = makechirp(s(k),tau,fs,0,1,num);

    ref1_fft = fft(ref1);
    ref2_fft = fft(ref2);
```

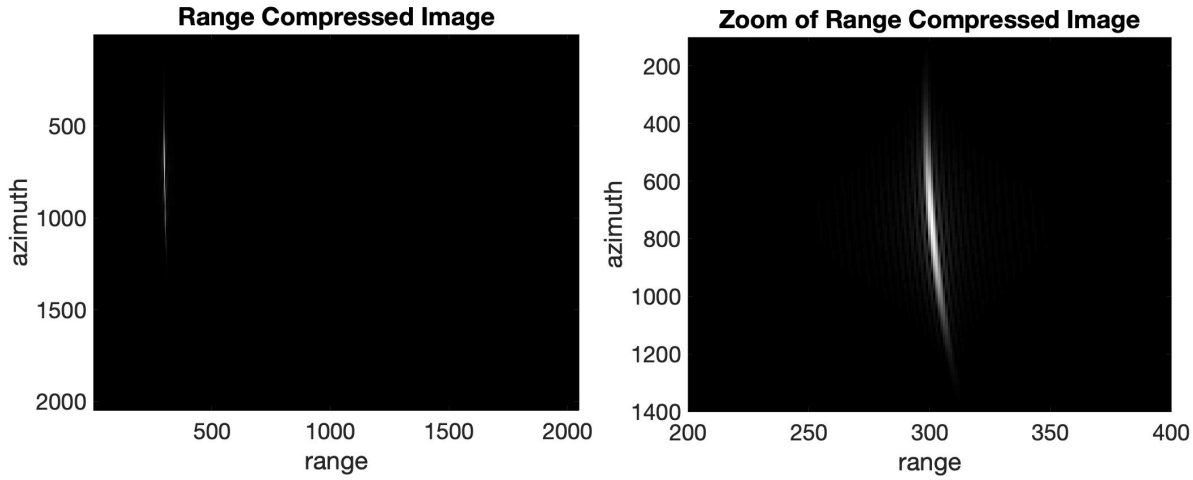
```
    refWH_fft = fft(refWH);

    cs1 = ifft(chp_fft.*conj(ref1_fft));
    cs2 = ifft(chp_fft.*conj(ref2_fft));
    csWH = ifft(chp_fft.*conj(refWH_fft));

    subplot(2,2,k)
    plot(t,fftshift(abs(csWH)),'k-','LineWidth',2)
    hold on
    plot(t,fftshift(abs(cs1)),'r-','LineWidth',2)
    plot(t,fftshift(abs(cs2)),'b-','LineWidth',2)
    hold off
    set(gca,'fontsize',20)
    title(['Slope = ' strslp{k}])
    xlabel('Time (s)')
    ylabel('Amplitude');
    xlim([-tau/10,tau/10])
    grid on
    legend('Full Spectrum','Neg Freqs','Pos Freqs')

    % Find the offset by correlating the two sub-aperture images
    shift = fftshift(ifft(fft(abs(cs1)).*conj(fft(abs(cs2)))));
    [maxnum,index] = max(abs(shift));
    dt(k) = (index-num/2-1)/fs;
    ds(k) = 2*dt(k)*s(k)^2/(tau*s(k))/1e12;
    pixeloffset(k) = (index-num/2-1);
end

function chirp = makechirp(s,tau,fs,fc,start,n)
%Function to compute chirp - reused in all problems
%s: slope
%tau: pulse length
%fs: sample rate
%fc: center frequency
%start: location of chirp
%n: the length of the chirp including zero
dt=1/fs;
npts=tau*fs;
t=[-npts/2:npts/2-1]*dt;
phase=pi*s*t.^2+2*pi*fc*t;
chirp=[zeros(1,start-1) exp(1i*phase) zeros(1,n-length(phase)-start+1)];
end
```

Question 2: simlband data



There is a distinctive curve to the returned signal. The boundaries are not confined to consistent range bins.

a) Range compress the data and examine the migration as a function of time.

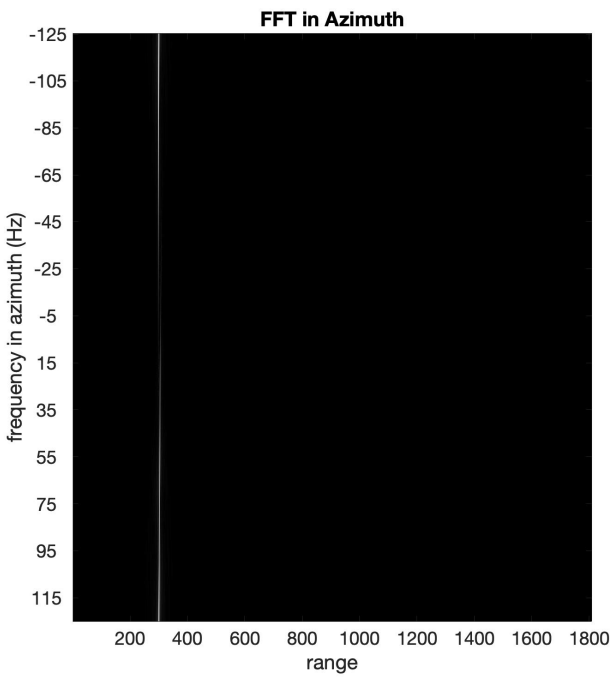


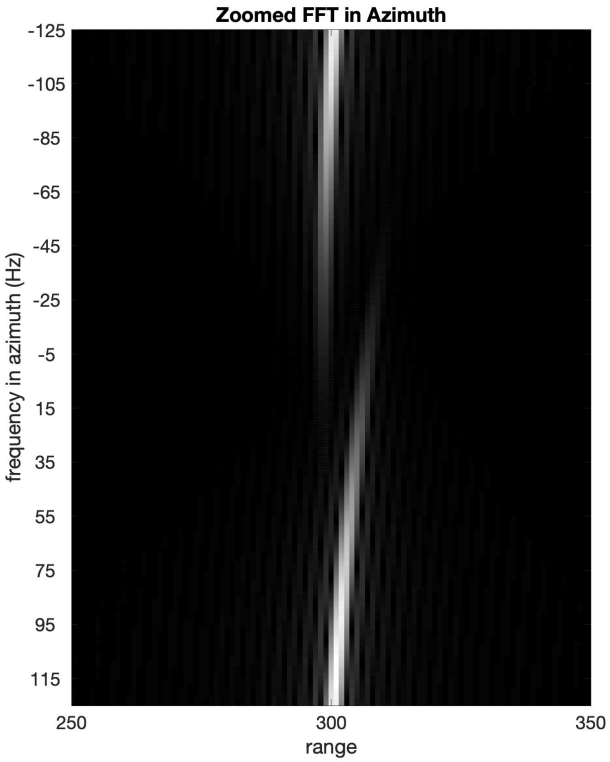
We have clearly succeeded in range-compressing the image, but the migration is still present, as better shown in the zoomed-in version.

```
%% Reference chirp parameters

num = 2048;
s = 1e12; % chirp slope (Hz/s)
tau = 10e-6; % pulse length (s)
fs = 24e6; % sample rate (Hz)

chp = makechirp(s,tau,fs,0,1,num);

chp_fft = fft(chp); % fft of reference chirp
```

```
signal_fft = fft(data,[],1);

for k = 1:num
    compress_fft(:,k) = signal_fft(:,k).*conj(chp_fft.');
end

sigcmp = ifft(compress_fft);

figure
subplot(2,1,1)
imagesc(abs(sigcmp)');
set(gca,'fontsize',20);
title('Range Compressed Image');
xlabel('range');
ylabel('azimuth');
colormap(gray)

subplot(2,1,2)
imagesc(abs(sigcmp)');
set(gca,'fontsize',20);
title('Zoom of Range Compressed Image');
xlabel('range');
ylabel('azimuth');
colormap(gray)
xlim([200 400])
ylim([100 1400])
```

b) Transform the compressed data in azimuth and display, again noting the migration path.

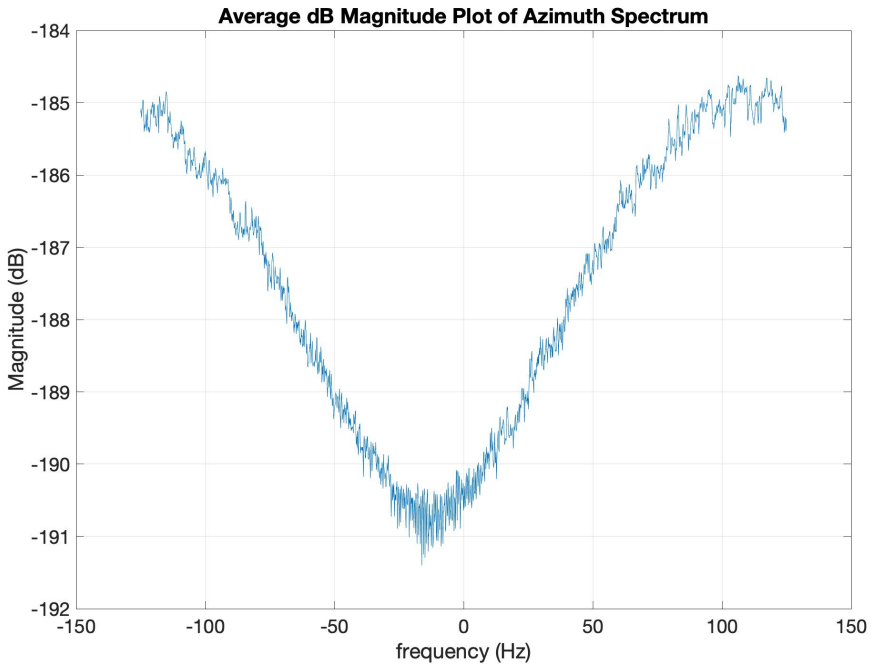




After transforming the range compressed data in the azimuth direction (left), we still see range migration between the two tail ends of the spectra (about 15-20 pixels), especially when we zoom in on range bins 250-350 (right). Notice that fDC is around prf/2 (part c), which explains why the "ends" of the spectra come together around -10 Hz. Also, the azimuth frequency range can have an ambiguity of n*prf, so we only plot the axis label as [-prf/2 prf/2].

```
%% calculate the fft in azimuth
nvalid = num-tau*fs;
cs_valid = sigcmp(1:nvalid,:);
azimuth_fft = fftshift(fft(cs_valid,[],2),2);
prf = 250;
freq = linspace(-prf/2,prf/2,num);

yticklabels = round(-prf/2):20:round(prf/2);
yticks = zeros(length(yticklabels),1);
for k = 1:length(yticklabels)
    indx = find(round(freq) == yticklabels(k));
    indx = round((indx(1) + indx(end))/2);
    yticks(k) = indx;
end

figure
subplot(1,2,1)
imagesc(abs(azimuth_fft)');
set(gca,'fontsize',20)
title('FFT in Azimuth');
colormap(gray)
xlabel('range');
ylabel('frequency in azimuth (Hz)');
axis image
set(gca, 'YTick', yticks, 'YTickLabel', yticklabels)

subplot(1,2,2)
imagesc(abs(azimuth_fft)');
set(gca,'fontsize',20)
title('Zoomed FFT in Azimuth');
colormap(gray)
xlabel('range');
ylabel('frequency in azimuth (Hz)');
xlim([250 350])
set(gca, 'YTick', yticks, 'YTickLabel', yticklabels)
```

c) Estimate the Doppler centroid of the data. List at least three possible Fdc's consistent with the ambiguous measurement of Fdc.



Since we really only have one scatterer, the average phase shift method couldn't be effectively used. Therefore, I estimated the Doppler centroid by finding the peak of the average spectrum and decided on Fdc = 106. To account for the ambiguity, I also added and subtracted multiples of the PRF=250 Hz. Some of the possible Fdc values are:

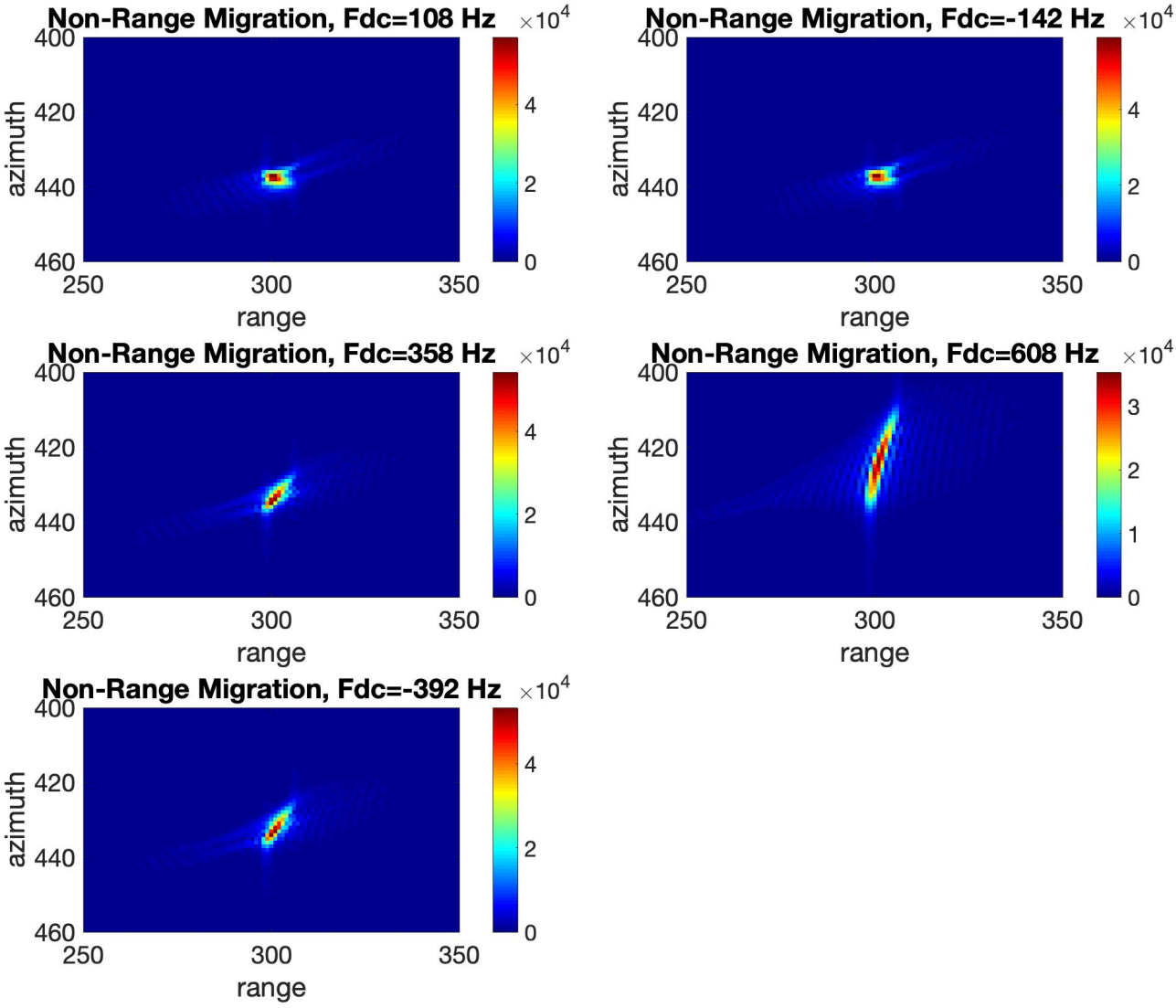
108 Hz -142 Hz 358 Hz 608 Hz -392 Hz

```
%% Find the Doppler Centroid

% Other radar parameters
prf = 250; %Hz
vel = 250; %m/s
l = 2; %m
lambda = 0.25; % m
r0 = 4653; % range to first bin (m)

% Calculate the dB magnitude
magnitude_azimuth = 20*log10(abs(azimuth_fft) + 1*10^-30);
%magnitude_azimuth = abs(azimuth_fft);
mag_az = mean(magnitude_azimuth,1);

figure
plot(freq,mag_az);
set(gca,'fontsize',20)
title('Average dB Magnitude Plot of Azimuth Spectrum');
xlabel('frequency (Hz)');
ylabel('Magnitude (dB)');
```

```
grid on

[val, indx] = max(mag_az);
Fdc = round(freq(indx));
Fdc = [Fdc Fdc-prf Fdc+prf Fdc+2*prf Fdc-2*prf]; % Doppler Centroid (Hz)
```

d) Process these data using the original (non-migrating) algorithm. Examine and plot the impulse response and describe the blurring. Process 80% of the azimuth bandwidth.

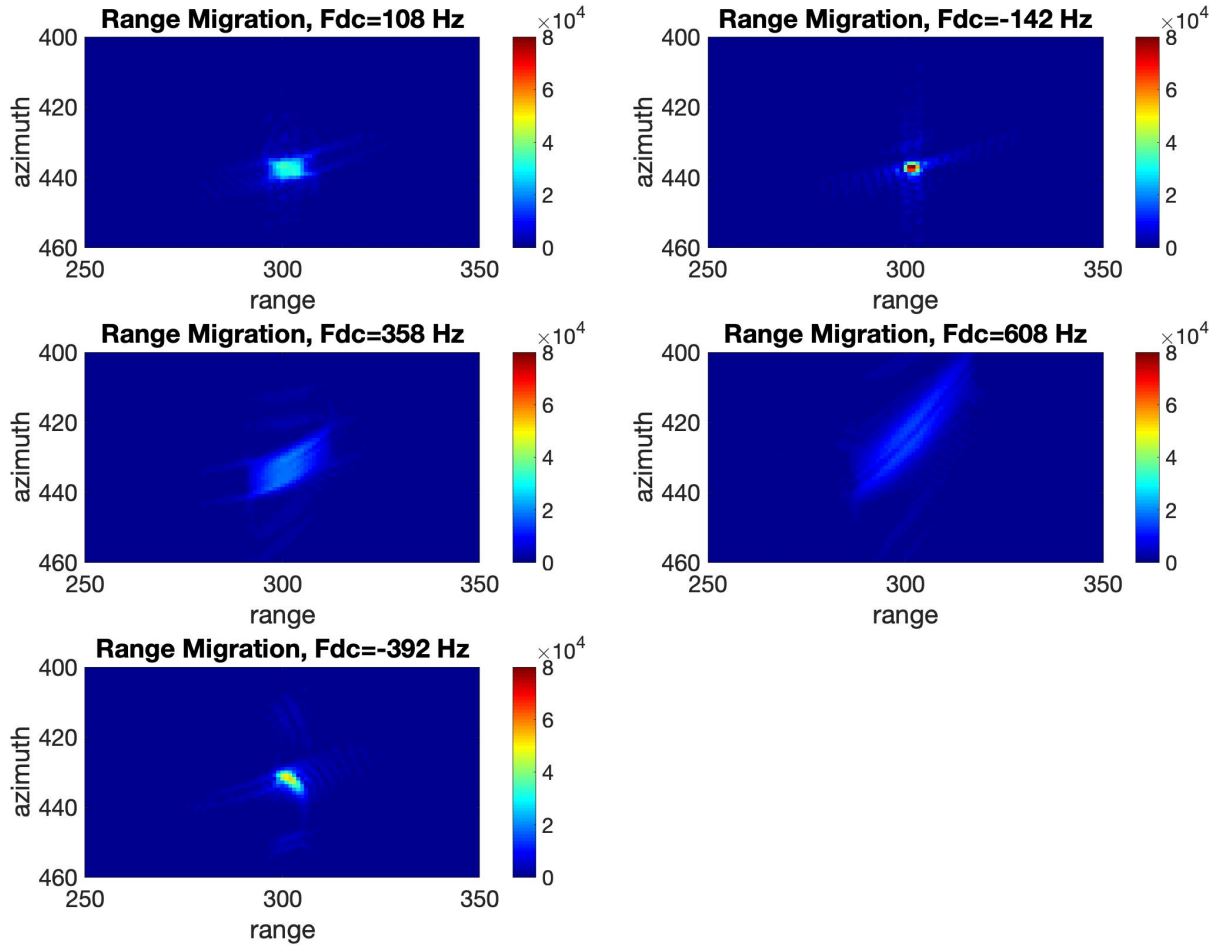
Note for part d and e, I am showing the zoomed-in version of the impulse response.



Although the point looks pretty good when zoomed out far, when we zoom in we can see some significant blurring, or spreading out of the signal over range bins. The best response is when we use a doppler centroid of -142 Hz, although our original pick of 108 Hz does almost just as well. 608 Hz does the worst, since it has the most smearing.

```
%% Question 2d
c = 3.0*10^8; %speed of light (m/s)
BW = s*tau; %Bandwidth

delta_slant=c/(2*BW); % slant range resolution

dr = c/fs/2; %bin spacing in slant range

figure
for n = 1:length(Fdc)
    fdc = Fdc(n); % The doppler centroid under consideration
    rmax = r0 + (nvalid-1)*dr; % Range to last bin
    rdc_max = sqrt(rmax^2 + (fdc*rmax*lambda/2/vel)^2); % Range DC
    taz_max = 0.8*rdc_max*lambda/l/vel; % Pulse length for ref chirp
    valid_az = floor(naz-taz_max*prf); % Valid azimuth bins in azimuth

    % Loop through each azimuth patch and match filter with range-specific
azimuth chirp
    focused = zeros(nvalid,naz);
    for bin=1:nvalid
        R = r0+(bin-1)*dr; % Get new r0 for each bin
        Rdc = sqrt(R^2 + (fdc*R*lambda/2/vel)^2); % Get new Rdc for each bin
        f_rate = -2*(vel^2)/lambda/Rdc; % Get new frequency rate
        tau_az = 0.8*Rdc*lambda/l/vel;

        ref_chp = makechirp(f_rate,tau_az,prf,fdc,1,naz);
        ref_fft = fft(ref_chp);

        cs_fft = fft(cs_valid(bin,:));

        signal = ifft(cs_fft.*conj(ref_fft));

        focused(bin,:) = signal';
    end

subplot(3,2,n)
imagesc(abs(focused)');
set(gca,'fontsize',20)
title(['Non-Range Migration, Fdc=' num2str(Fdc(n)) ' Hz'])
xlabel('range')
ylabel('azimuth')
colormap('jet');
colorbar
xlim([250 350]);
ylim([400 460]);
end
```

e) Apply the cut and paste algorithm assuming each Fdc found in (c) above. Which gives the best impulse response? Again use 80% bandwidth.



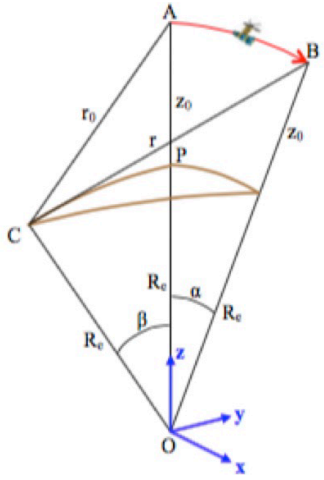
Using the cut and paste algorithm, we get the best concentration of the returned signal with a doppler centroid of -142 Hz. The others have a significant smearing of the returned energy. However, note that our returned signal using the cut-and-paste is still not perfect. We would likely see better results if we also added an interpolation to our cut-and-paste algorithm. Also note that the range migration process increased the sensitivity to the Doppler Centroid ambiguity.

```
%% Question 2e: Apply the cut and paste algorithm assuming each Fdc found
% in c above. Which gives the best impulse response? Again, use 80% of the
% bandwidth.

df = prf/naz;

AZ_fft = fft(cs_valid,[],2);
focused_fft= zeros(size(AZ_fft));
focused_rm = focused_fft;

figure
for n = 1:length(Fdc)
    fdc = Fdc(n); % The doppler centroid under consideration
    fmin = multi(n)*prf; % minimum frequency
```

```
    % Get the cut-and-past shift
    focused = zeros(nvalid,naz);
    for bin=1:nvalid
        for k = 1:naz

            f = fmin + (k-1)*df; % Get the frequency at current bin
            offset = (f^2 - fdc^2)*(r0/8)*(lambda/vel)^2; % (m)

            shift = floor(offset/dr); % number of offset range bins

            if bin+shift <= nvalid && bin+shift > 0
                focused_fft(bin,k) = AZ_fft(bin+shift,k);
            end

        end
    end

    % Get the matched filter for azimuth
    for bin = 1:nvalid
        R = r0+(bin-1)*dr; % Get new r0 for each bin
        Rdc = sqrt(R^2 + (fdc*R*lambda/2/vel)^2); % Get new Rdc for each bin
        f_rate = -2*(vel^2)/lambda/Rdc; % Get new frequency rate
        tau_az = 0.8*Rdc*lambda/l/vel;

        ref_chp = makechirp(f_rate,tau_az,prf,fdc,1,naz);
        ref_fft = fft(ref_chp);

        signal = ifft(focused_fft(bin,:).*conj(ref_fft));

        focused_rm(bin,:) = signal';
    end

subplot(3,2,n)
imagesc(abs(focused_rm)');
set(gca,'fontsize',20)
title(['Range Migration, Fdc=' num2str(Fdc(n)) ' Hz'])
xlabel('range')
ylabel('azimuth')
colormap('jet');
colorbar
xlim([250 350]);
ylim([400 460]);
caxis([0 8e4]);
end
```

Question 3: Effective Velocity

We can consider the radar geometry as two concentric spheres. The outer sphere is the orbital sphere of the satellite, the inner sphere is the Earth. Point A is the point of closest approach of the satellite to point C, and point B is some later position of the satellite. We want to find the range r from B to C.



We derived the effective velocity for the point P (directly under the satellite) in Handout 22. The relative height history z(t) between the point P and the satellite can be written as:

$$z(t) = z_0 - \frac{v^2 t^2}{2(z_0 + R_e)}$$

From geometry, the height difference between point P (directly under the satellite) and point C (not directly under the satellite) is:

$$\cos\beta = \frac{R_e - \Delta z}{R_e} \Rightarrow \Delta z = R_e - R_e \cos\beta$$

Now write the expression for range as a function of time, $r^2(t)$, but replace z(t) with $z(t) + \Delta z$:

$$r^2(t) = v^2 t^2 + y^2 + (z(t) + \Delta z)^2$$

$$r^2(t) = v^2 t^2 + y^2 + (z(t))^2 + 2z(t)\Delta z + (\Delta z)^2$$

Plug in the first equation into our new equation for $r^2(t)$ and simplify. Here, we neglect the higher order v^4t^4 term.

$$r^t(t) = v^2 t^2 + y^2 + \left(z_0 - \frac{v^2 t^2}{2(z_0 + R_e)}\right)^2 + 2\left(z_0 - \frac{v^2 t^2}{2(z_0 + R_e)}\right)\Delta z + (\Delta z)^2$$

$$r^2(t) = v^2t^2 + y^2 + z_0^2 - \frac{z_0 v^2 t^2}{z_0 + R_e} + 2\left(z_0 - \frac{v^2t^2}{2(z_0 + R_e)}\right)\Delta z + (\Delta z)^2$$

Gather all terms with v^2t^2, and call everything else constant (i.e. not changing with time):

$$r^2(t) = y^2 + z_0^2 + 2z_0\Delta z + (\Delta z)^2 + v^2t^2 - \frac{z_0 v^2 t^2}{z_0 + R_e} - 2\left(\frac{v^2t^2}{2(z_0 + R_e)}\right)\Delta z$$

$$r^2(t) = const + v^2t^2\left[1 - \frac{z_0}{z_0 + R_e} - \frac{\Delta z}{z_0 + R_e}\right]$$

The effective velocity v_{eff}^2 is the term in brackets, times v^2, so simplify:

$$v_{eff}^2 = v^2\left[1 - \frac{z_0}{z_0 + R_e} - \frac{\Delta z}{z_0 + R_e}\right]$$

$$v_{eff}^2 = v^2\left[\frac{z_0 + R_e}{z_0 + R_e} - \frac{z_0}{z_0 + R_e} - \frac{\Delta z}{z_0 + R_e}\right]$$

$$v_{eff}^2 = v^2\left[\frac{R_e - \Delta z}{z_0 + R_e}\right]$$

Plug in our knowledge of Δz and take the square root:

$$v_{eff}^2 = v^2\left[\frac{R_e - (R_e - R_e\cos\beta)}{z_0 + R_e}\right]$$

$$v_{eff}^2 = v^2\left[\frac{R_e\cos\beta}{z_0 + R_e}\right]$$

$$v_{eff} = v\sqrt{\frac{R_e\cos\beta}{z_0 + R_e}}$$