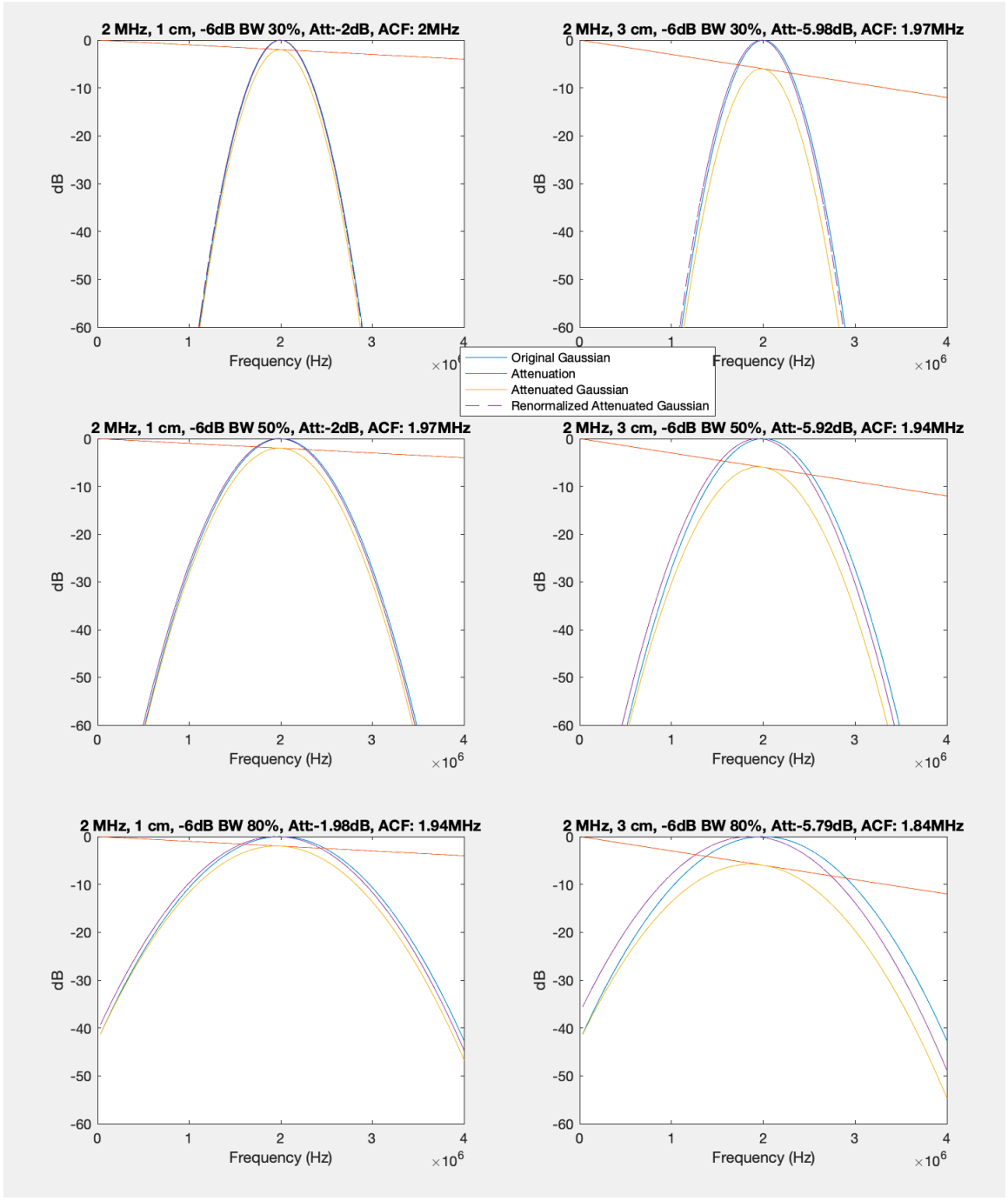
Homework: BME8730
Fall 2019

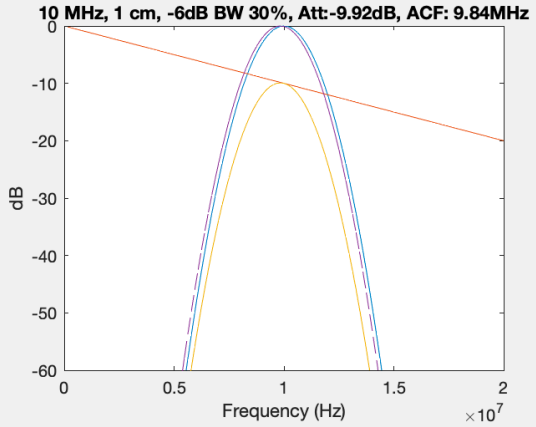
Samuel Sklar

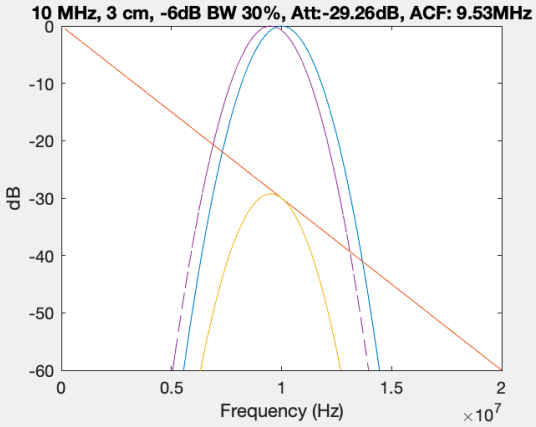
Due Monday 30th.
Please plan to discuss and ask questions Monday and/or Wednesday in class next week.
Code provided is outline only and is not working. You need to make it work. Feel free to adapt or re-write from scratch.
You may collaborate but must acknowledge.
100 points total.

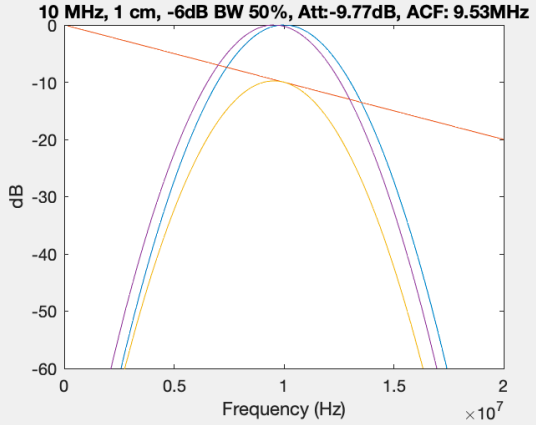
1. Frequency dependent attenuation.
Gaussian function generation in the frequency domain.
Create functions corresponding to 20%, 50% and 80% -6dB fraction bandwidth.
(We did this as part of the beam plot exercise in class)
Create functions at 2 MHz center frequency and 10 MHz center frequency
Consider imaging depths of 1 cm and 3 cm
Apply frequency dependent attenuation at the rate of 0.5 dB/cm/MHz – and account for the fact that we are dealing with two way propagation.
For this question, multiply the attenuation rate by the imaging depth and then make it frequency dependent – i.e. at each frequency increment in your code, recalculate the dB attenuation which at that point accounts for both frequency and depth. Attenuation is a multiplier /scaling effect. Since this is in the dB domain, it is additive / subtractive. In effect, in the dB domain, attenuation appears as a low pass filter with a straight line (versus frequency) downward pattern.

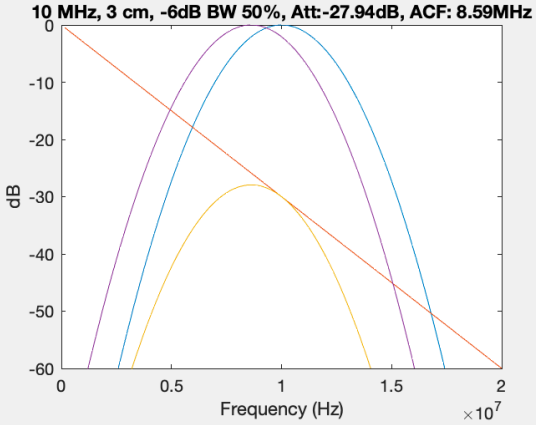
For each permutation of center frequency, bandwidth and imaging depth, plot
The original spectrum (magnitude) and the attenuated version. Measure off from the plot the reduction in amplitude at the peak level (in spectrum) and the attenuated versus original center frequency. Calculate the center frequency from the midpoint between the -6dB upper and lower cutoff frequencies.
High frequencies experience more absolute attenuation.
Higher bandwidth pulses experience greater frequency downshift. (12)

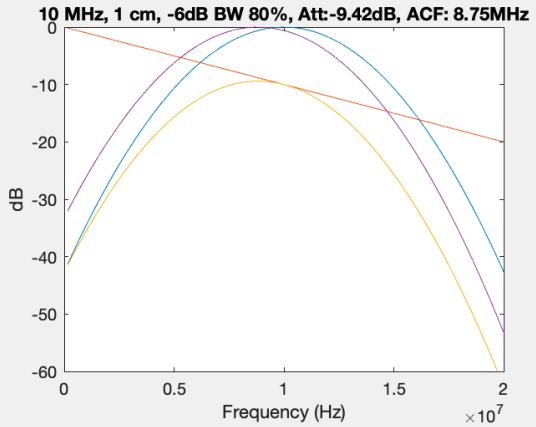

2 MHz, 1 cm, -6dB BW 30%, Att:-2dB, ACF: 2MHz
2 MHz, 3 cm, -6dB BW 30%, Att:-5.98dB, ACF: 1.97MHz
2 MHz, 1 cm, -6dB BW 50%, Att:-2dB, ACF: 1.97MHz
2 MHz, 3 cm, -6dB BW 50%, Att:-5.92dB, ACF: 1.94MHz
2 MHz, 1 cm, -6dB BW 80%, Att:-1.98dB, ACF: 1.94MHz
2 MHz, 3 cm, -6dB BW 80%, Att:-5.79dB, ACF: 1.84MHz
dB
Frequency (Hz)
×10^6
Original Gaussian
Attenuation
Attenuated Gaussian
Renormalized Attenuated Gaussian

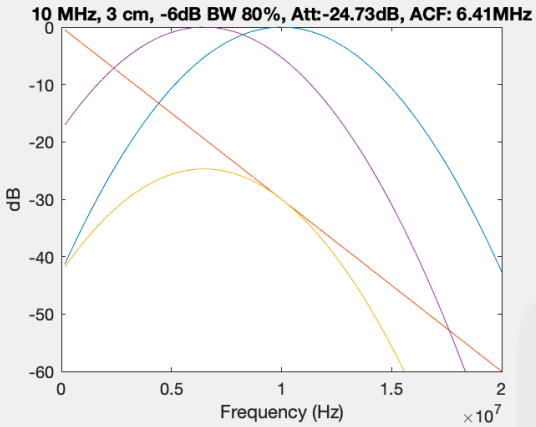

10 MHz, 1 cm, -6dB BW 30%, Att:-9.92dB, ACF: 9.84MHz
dB
Frequency (Hz)
×10^7


10 MHz, 3 cm, -6dB BW 30%, Att:-29.26dB, ACF: 9.53MHz
dB
Frequency (Hz)
×10^7


10 MHz, 1 cm, -6dB BW 50%, Att:-9.77dB, ACF: 9.53MHz
dB
Frequency (Hz)
×10^7


10 MHz, 3 cm, -6dB BW 50%, Att:-27.94dB, ACF: 8.59MHz
dB
Frequency (Hz)
×10^7


10 MHz, 1 cm, -6dB BW 80%, Att:-9.42dB, ACF: 8.75MHz
dB
Frequency (Hz)
×10^7


10 MHz, 3 cm, -6dB BW 80%, Att:-24.73dB, ACF: 6.41MHz
dB
Frequency (Hz)
×10^7

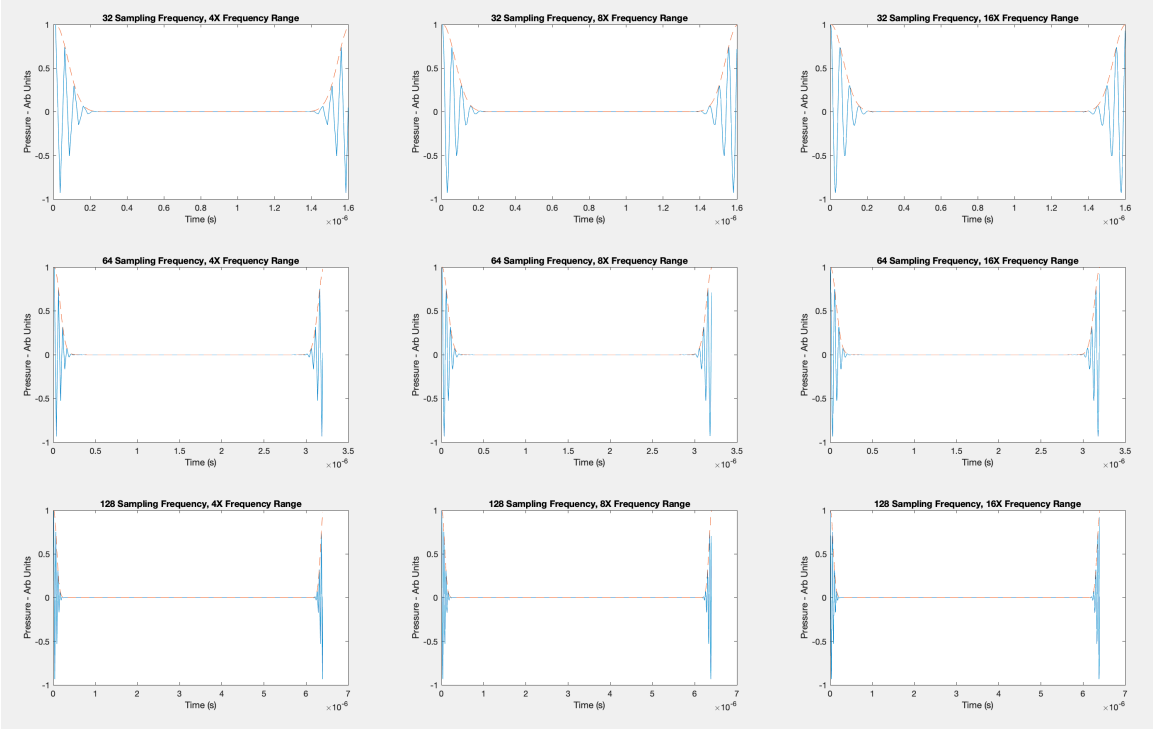
```
bws=[30,50,80];
fcs=[2e6,10e6];
z_foc=[1e-2,3e-2];
GAFout=cell(length(bws),length(fcs),length(z_foc));
attenuation=zeros(length(bws),length(fcs),length(z_foc));
acf=zeros(length(bws),length(fcs),length(z_foc));
count=0;
figure
for m=1:length(bws)

    for n=1:length(fcs)

       for k=1:length(z_foc)
           count = count+1;
           [f,att,gauss_pulse,gauss_pulse_att,gauss_t,env_gauss_t,t]=GauAttFn(bws(m),fcs(n),z_foc(k),64,8);
           attenuation(m,n,k) = max(20*log10(gauss_pulse_att));
           acf(m,n,k) = spline(20*log10(gauss_pulse_att(20:64)),f(20:64),attenuation(m,n,k));
            subplot(3,4,count);
            plot(f,20*log10(gauss_pulse));
            hold on
            plot(f,att);
            plot(f,20*log10(gauss_pulse_att));
            plot(f,20*log10(gauss_pulse_att)-attenuation(m,n,k),'--');
            ylim([-60,0])
            xlim([fcs(n)*0,fcs(n)*2])
            title([num2str(fcs(n)*10^-6),' MHz, ', num2str(z_foc(k)*100),...
                ' cm, -6dB BW ', num2str(bws(m)), '%, Att:',...
                num2str(round(attenuation(m,n,k),2)), 'dB, ACF: ',...
                num2str(round(acf(m,n,k)*10^-6,2)),'MHz'])
            xlabel('Frequency (Hz)')
            ylabel('dB')
            if count==1
                legend('Original Gaussian','Attenuation','Attenuated Gaussian'...
                    ,'Renormalized Attenuated Gaussian')
            end
            hold off
       end

    end
end
end
```

```
function [f,att,gauss_pulse,gauss_pulse_att,gauss_t,env_gauss_t,t]=GauAttFn(bw,fc,z_foc,sf,range)
fs=fc/sf;                                   % Define a sampling frequency (not critical)
                                            % Observe result when we choose
                                            % "64" to be "32" or "128".
                                            % This is sampling in the
                                            % frequency domain so it does
                                            % what when we convert to time
                                            % domain?
f=[fs:fs:range*fc];                             % Define an adequate frequency range (improve res
                                            % Observe the result when you
                                            % make the "8" a "4" or a "16"
                                            % -Plot spectrum and plot time
                                            % domain - essentially, the
                                            % maximum frequency is extended
                                            % and this is the Nyquist limit
                                            % so it does what in the time
                                            % domain?
                                            % calculation
w=2*pi*f;                                   % Angular frequency radians
ns=length(f);                               % Number of samples
tdel=0e-6;                                  % Use a fixed time offset so that initial waveform is
tdel2=1.0e-6;                               % Use a fixed time offset so that initial waveform is
                                            % Plot the time domain version
                                            % using each of these delays
                                            % and observe the effect
sig=bw*fc/100;                              % Width of Gausian
gauss_pulse=exp(-pi*((f-fc)/sig).^2);       % Generate Gaussian pulse (frequency domain)

gauss_pulse=gauss_pulse.*exp(-j*w*tdel);    % Apply time delay so 0 for t<0
att= -.00005*z_foc*f*2;

gauss_pulse_att=10.^(att./20).*gauss_pulse;         % Attenuation
gauss_t=real(ifft(gauss_pulse));            % Time domain of base waveform for reference
gauss_t=gauss_t./max(gauss_t);
env_gauss_t=abs(hilbert(gauss_t));          % Envelope calculation - i.e. peak value of sinusoida

tstep=1./max(f);                            % Time steps after using Inverst FFT
t=[1:ns].*tstep;                            % Define time axis
end
```

2. Using the code approximately developed in class, review and make sure that it is doing the following:
Create a frequency axis – to at least 8x center frequency and a well sampled frequency interval – e.g. 512 frequency samples
Create the Gaussian function – i.e. 10 MHz, 30% -6dB BW
Set up the field sampling points – either along a azimuthal oriented line at a fixed range or at a 2D matrix of points in both azimuth and range
Calculate desired focusing delays – i.e. “inverted” time delay from all elements to the focal point
Step through field points.
Step through the array element locations
Calculated propagation delays to field point from the element location to the field point
Calculated the net delayed response at the field point taking account of the propagation delay and the applied focal delay (noting that when at the focal point the propagation delay and applied focal delay will cancel – hence the “inverted” focal delay)
Sum the contributions from all elements at the field point
Take the IFFT of the summed value
Find the peak amplitude of the envelope of the IFFT waveform. Use the magnitude of the Hilbert transform.
Record the peak amplitude value for that field point location
Determine and apply any normalization
(Typically, values are normalized at each range increment when doing a beam plot through azimuth and depth. This is because in a real imaging system, depth (or focus)

dependent variations in peak amplitude are compensated out to produce a more even image intensity.)

Perform the interim steps suggested in the comments in the template code – i.e. changing the frequency axis sampling and extent to verify the expected impositions made on the time and frequency representations as you execute FFT and IFFT. Make plots of all results and use your own judgment as to whether to overplot lines or use separate figures. Differentiate and label lines as necessary. (9)



I did not get an appreciable difference in the frequency domain by changing these parameters.

```
bws=30;
fcs=20e6;
z_foc= 3e-2;
sfs=[32,64,128];
ranges=[4,8,16];
count=0;
 for n=1:length(sfs)
       for m=1:length(ranges)
           count = count+1;
           [f,att,gauss_pulse,gauss_pulse_att,gauss_t,env_gauss_t,t]=...
               GauAttFn(bws,fcs,z_foc,sfs(n),ranges(m));
            figure(1);
            subplot(3,3,count);
            plot(t,gauss_t)
            hold on
            plot(t,env_gauss_t,'--')
            title('Gaussian Pulse')
            xlabel('Time (s)')
            ylabel('Pressure - Arb Units')
            title([num2str(sfs(n)),' Sampling Frequency, ', num2str(ranges(m)),...
                'X Frequency Range'])
            hold off
       end

 end
```

2A. Simulate a beamplot (i.e. magnitude of waveform, in dB, at a range of X- offsets around X,Z=(0,50) (i.e. at Z=50mm)) for a 10 MHz, 30% BW transducer array of 128 elements and wavelength spacing focused at a point straight ahead at range 50 mm. Simulate for the case of no apodization (weighting = all 1's) and for Hann weighting. Verify the impact of apodization on main beam width and on presence of sidelobes. (Use your judgment to extend the range of the scale in both X direction (x-axis) and numbers of dBs (y-axis) in the plot to show this)
Use a narrow beam sampling so as to capture the main lobe and at least the first sidelobe. You may need to experiment with sampling. If the sampling is too coarse, you will not resolve the main beam and side lobes properly. +/- 2 mm is approximately sufficient for this question.

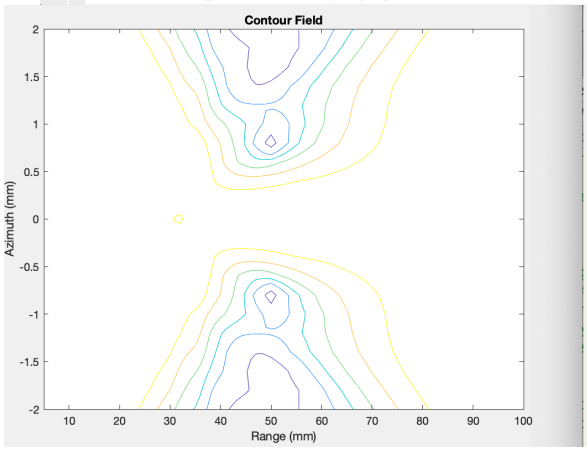
Your code should be able to create maps over 2D space (varying X and Z) over 1D space (varying X at a fixed Z – to get a beam cross-section although conventionally simply called a "beamplot")

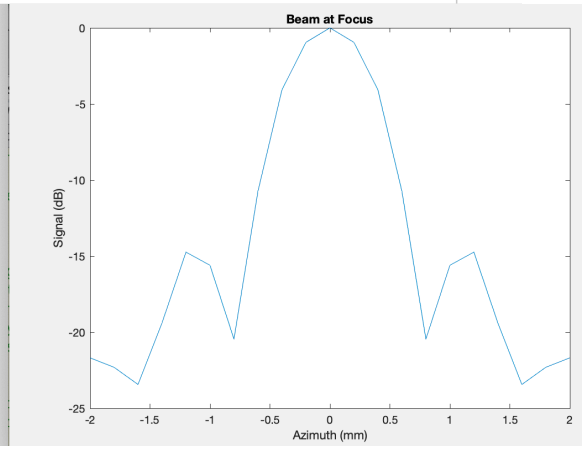
For this question and questions 3 and 4, you want to zoom in on the main lobe – plot over a range of approximate X = -2, + 2mm. These questions relate to main lone width and the presence of sidelobes so you need to resolve the main lobe properly.

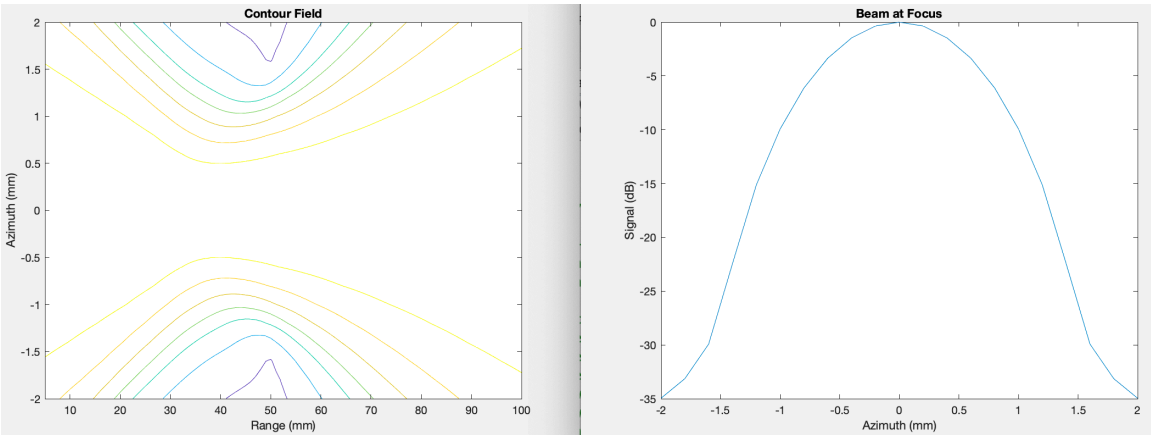
This code used for 2-4,6

```
z_foc=50e-3;                                % Range direction focal distance\
theta_steer=(45)*pi/180;
num_elems=64;                              % Number of array elements
fc=10e6;                                    % Center frequency
pitch=(.075e-3);
vel=1540;                                   % Speed of sound - all units MKS
fs=fc/64;                                   % Define a sampling frequency (not critical)
f=[fs:fs:8*fc];                             % Define an adequate frequency range (improv
i_cen=round(fc/fs);                         % Index of center frequency for single frequ
w=2*pi*f;                                   % Angular frequency radians
ns=length(f);                               % Number of samples
tdel=0e-6;                                  % Use a fixed time offset so that initial wa
tdel2=1.0e-6;                               % Use a fixed time offset so that initial wa
bw=30;                                      % Fractional bandwidth as percent
bw2=80;                                     % Fractional bandwidth as percent
sig=bw*fc/100;                              % Width of Gausian
gauss_pulse=exp(-pi*((f-fc)/sig).^2);       % Generate Gaussian pulse (frequency domain)
gauss_pulse=gauss_pulse.*exp(-j*w*tdel);    % Apply time delay so 0 for t<0
%att= .00005*f;
att=0;
%%
weight=ones(num_elems,1);                  % Define the weighting function (you can chan

%weight=hann(num_elems);
%x_pts=[40:0.2:60].*1e-3;
%z_pts=[5:1.0:2*z_foc*10^3].*1e-3;
x_pts=[-60:0.2:60].*1e-3;                     % Define X-direction field locations
z_pts=[5:1.0:60].*1e-3;

x_elem=zeros(num_elems,1);
foc_del=zeros(num_elems,1);
steer_del=zeros(num_elems,1);
for i=1:num_elems
    x_elem(i)=((i-1)-(num_elems-1)./2).*pitch;  % Calculate locations of each array elem
    steer_del(i) =  (((pitch*sin(theta_steer)*i)+sqrt(((x_elem(i))^2)+(z_foc^2)))./vel;
end
```

```
field_val= zeros(length(z_pts),length(x_pts));
field_val_fc= zeros(length(z_pts),length(x_pts));
field_db= zeros(length(z_pts),length(x_pts));
field_fc_db= zeros(length(z_pts),length(x_pts));
for j=1:length(z_pts)                       % Loop over field locations
    fprintf('%d ',j)                        % Short piece of code to provide reco
    if (j/20)==round(j/20)
            fprintf('\n')
    end
    for i=1:length(x_pts)
        sum_pulse=zeros(size(gauss_pulse));                % Initialize sum of v
        for k=1:num_elems
            r=sqrt((x_elem(k)-x_pts(i))^2+z_pts(j)^2);
            prop_del = (r./vel);
              sum_pulse=sum_pulse+weight(k).*10.^(-att.*r./20).*gauss_pulse.*...
                exp(-sqrt(-1)*w.*(prop_del-steer_del(k)));
        end
        sum_pulse_t=real(ifft(sum_pulse));                 % Convert to time dom

        field_val(j,i)= max(abs(hilbert(sum_pulse_t)));     % Field values in bea
        field_val_fc(j,i)=abs(sum_pulse(i_cen));           % Magnitude of summed

    end
    field_val(j,:)=field_val(j,:)./max(field_val(j,:));     % Normalize by peak v
    field_db(j,:)=20*log10(field_val(j,:));                % Convert to dB
    field_val_fc(j,:)=field_val_fc(j,:)./max(field_val_fc(j,:));
    field_fc_db(j,:)=20*log10(field_val_fc(j,:));
end
%%
figure;
contour(z_pts*1e3,x_pts*1e3,field_db',[-30 -20 -15 -12 -9 -6 -3]);
title('Contour Field')
xlabel('Range (mm)')
ylabel('Azimuth (mm)')
figure
mesh(z_pts*1e3,x_pts*1e3,field_db');
title('Mesh Field')
xlabel('Range (mm)')
ylabel('Azimuth (mm)')
zlabel('Signal (dB)')
figure
plot(x_pts*1e3,field_db(-5+z_foc*10^3,:))
title('Beam at Focus')
ylabel('Signal (dB)')
xlabel('Azimuth (mm)')
```

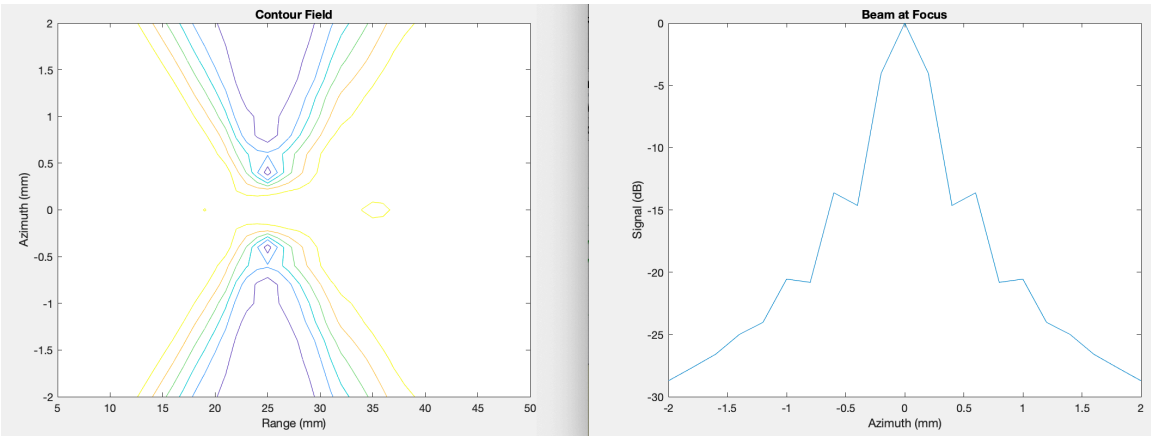






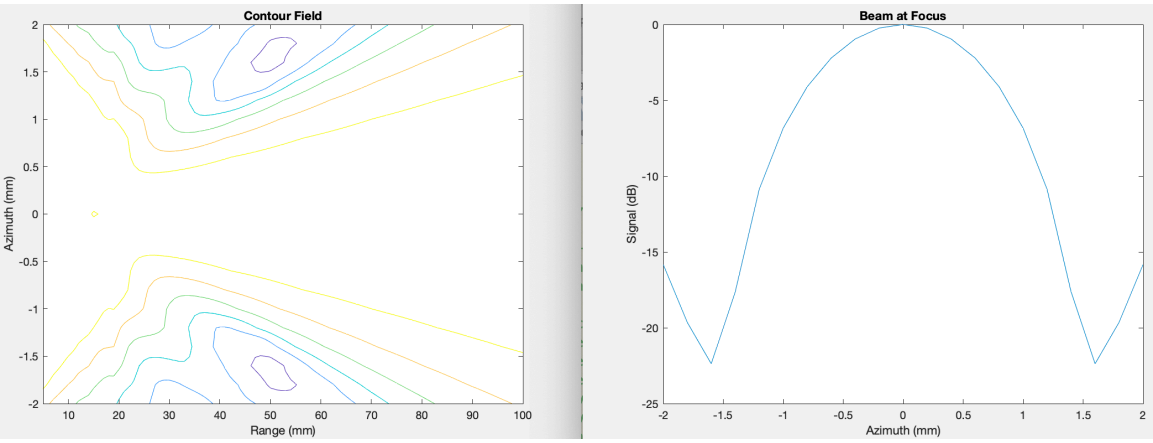
The Han weighting does kill the sidelobes and does increase the width of the main lobe

2B. Form a beamplot with a focus close in – X,Z=(0,25)mm. Verify that the apparent depth of focus is shorter. “Depth of focus” means the range in depth over which the beamwidth is still very narrow – e.g. <=200% of beam width at the true minimum. Just place annotation on your plot indicating this effect. Estimate the approx range (by extracting data or by measurement from plot) the range of depths around the focus for which the -6 dB width is no more than 200% that at the actual focus. i.e. if -6dB width (X dimension) is 1.0 mm at focus, over what range (Z) is the -6dB width less than 2.0 mm. (6)

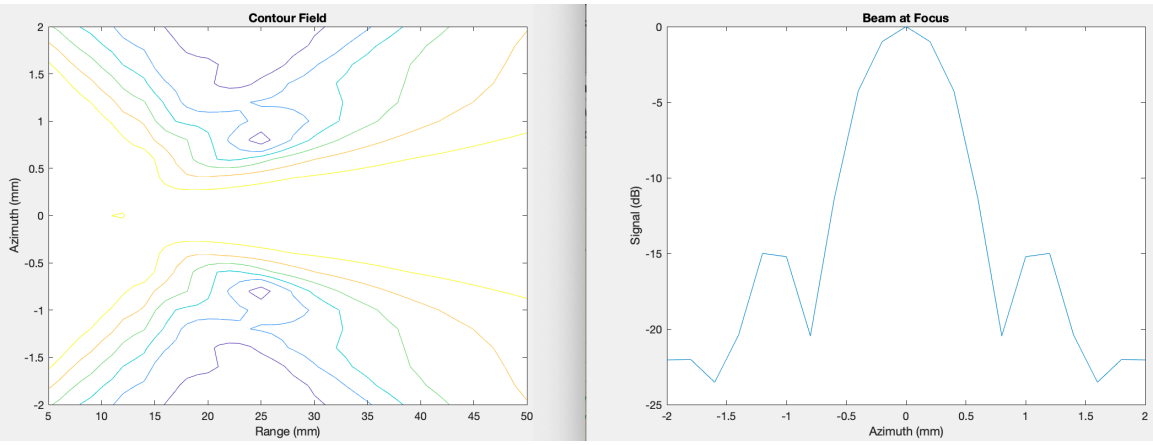


The main lobe is narrower at the focus but the narrowing only lets for about 5mm on either side of the focus where as with the 50mm focus, the narrow region lasts for around 10mm on either side of the focus. The ends of these regions are clearly apparent

3. Repeat 2 (no apodization) but with half wavelength spacing. i.e. shorter aperture. Verify that main lobe (at -6dB level) is
doubled in width. (4)
z_foc=50mm



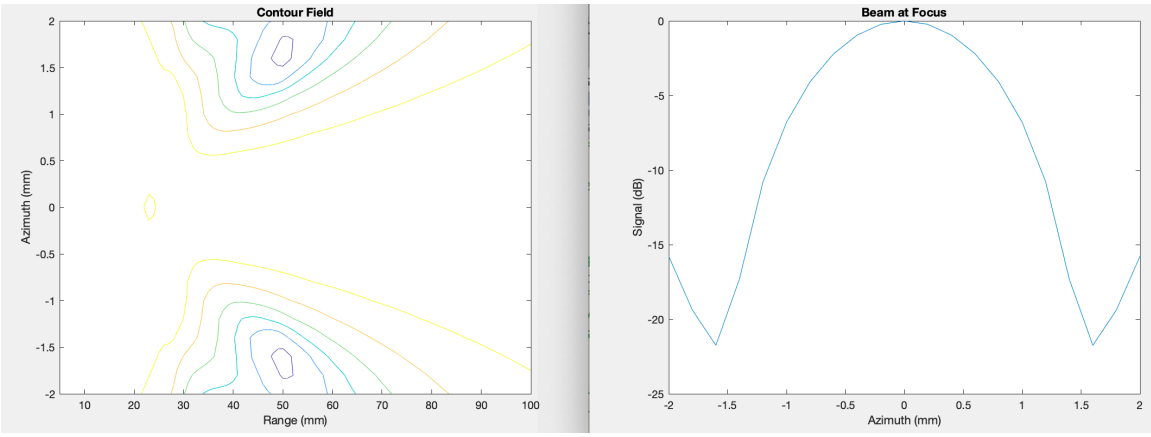
z_foc=25mm



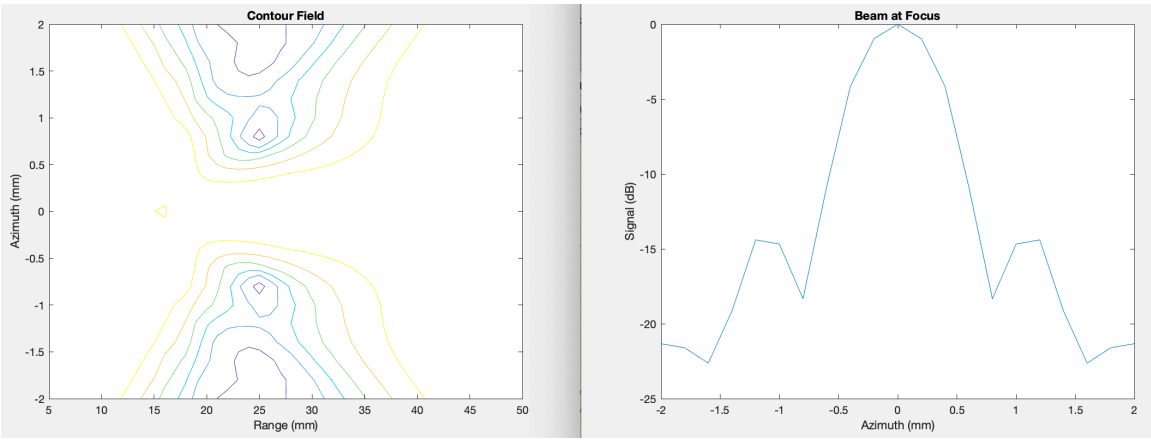
The main lobes have clearly doubled in width.

4. Repeat 2 (no apodization) with one wavelength spacing but the center frequency is now 5 MHz. Verify that main lobe (at -6dB level) is the same as in 2. Why isn't the main lobe wider because we are using a lower frequency? (4)

z_foc=50mm



z_foc=25mm



The beam plots look basically the same as from problem 3. By switching to a lower frequency we have offset the return to a wider aperture. A few differences can be seen in the contour plots though.

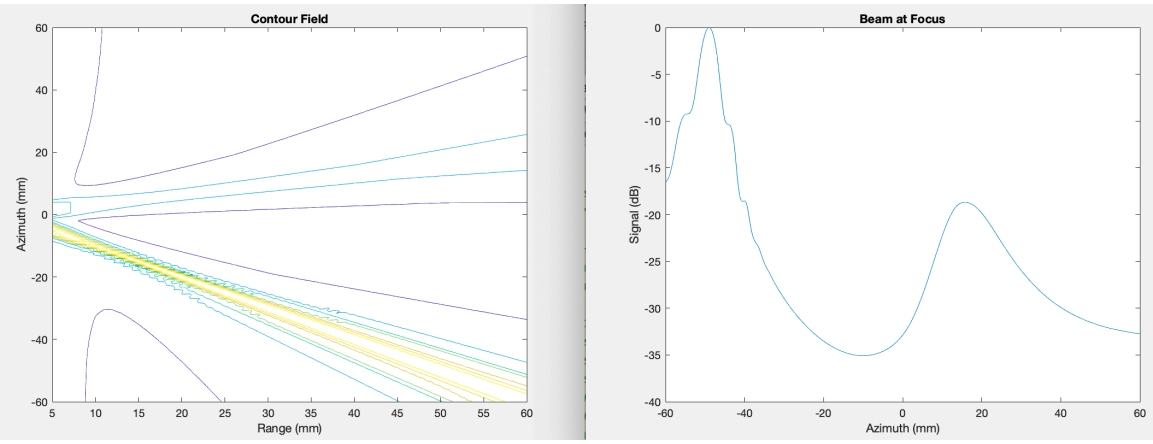
5. Measure from the pulse plot the code produces the -20dB (i.e. first crossing of amplitude over 0.1 of max value to last crossing of amplitude over 0.1 of max value) pulse duration for both the 10 MHz and 5 MHz pulses with 30% bandwidth. Convert these to axial resolution using speed of sound and factor of two for two way propagation. (6)
Code references the function from problem 1 in lines 8,9

```
1 -    bws=30;
2 -    fcs=[5e6,10e6];
3 -    z_foc= 50e-2;
4 -    sfs=64;
5 -    ranges=8;
6 -    rez=zeros(length(fcs),1);
7 -     for n=1:length(fcs)
8 -                [f,att,gauss_pulse,gauss_pulse_att,gauss_t,env_gauss_t,t]=...
9                      GauAttFn(bws,fcs(n),z_foc,sfs,ranges);
10 -                st=spline(env_gauss_t(10:40*n),t(10:40*n),.1);
11 -                ed=spline(env_gauss_t(50*n:70*n),t(50*n:70*n),.1);
12 -                rez(n) = (ed-st)*1540.*2
13 -    end
```

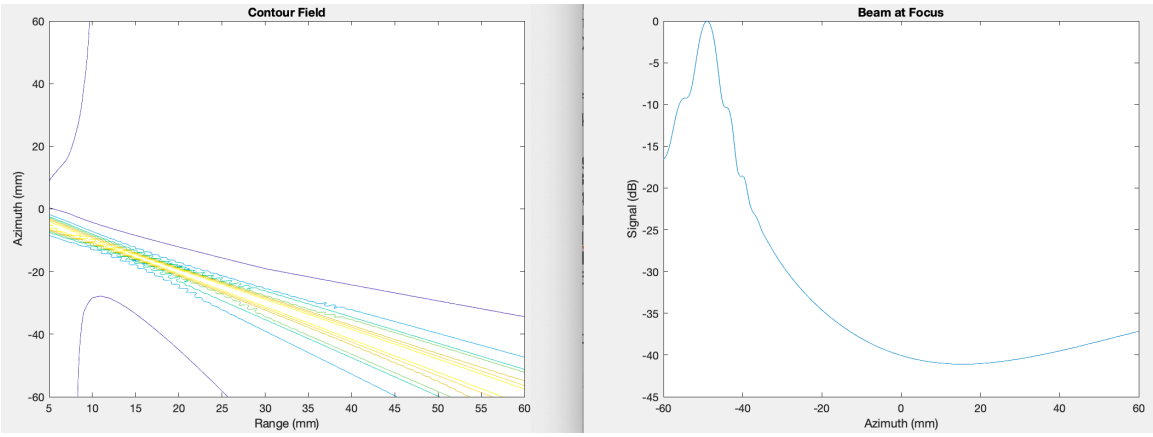
This code returned 3.5mm for the 5MHz pulse, and 1.8mm for the 10 MHz pulse so the doubled frequency also makes the resolution about twice as fine.

6. Steer the beam by 45 degrees. Plot an entire beam pattern using the contour plot option. Use 10 MHz, 30% BW, 50 mm focusing in front of the transducer, no apodization, and then superimpose the 45 degree steering. Do this for wavelength and half wavelength spacing and verify that you can see grating lobes (an artifactual sidelobe that is related to spatial undersampling – akin to sub-Nyquist sampling) at the opposite angle to the intended main beam for the wavelength spacing case. You should find that as you steer the beam to 45 degrees the apparent range to focus moves closer to the beam origin. Can you explain? It has to do with effective element spacing. (8)

With num_elem=32 and pitch=(.075e-3)*2



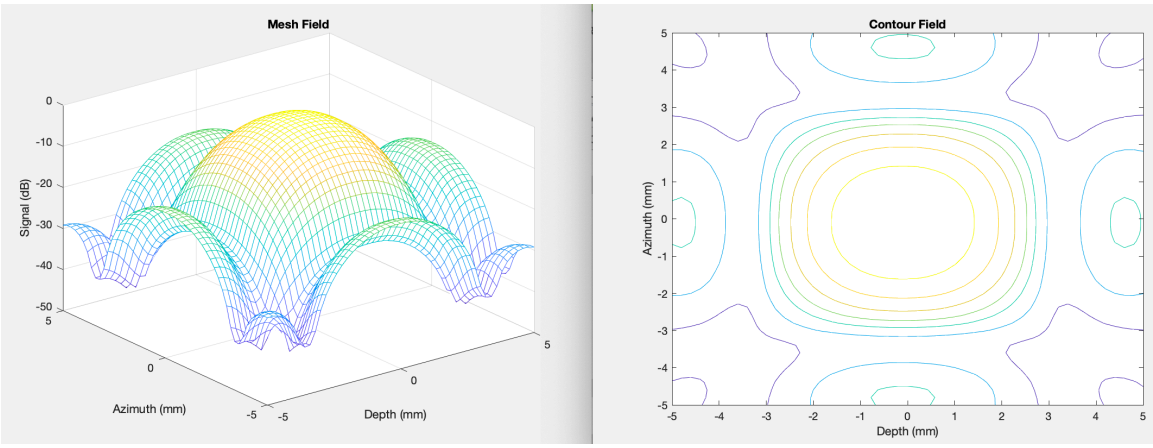
With num_elem=64 and pitch=(.075e-3)



The narrowest point in the main beam has moves back to around 17mm. When the spacing is halved the side lobe disappears.

7. Extend the program to simulate a 2D array with elements populating a 32 x 32 matrix of X,Y locations. Assume 10 MHz, 30% BW, wavelength spacing. Focus at Z=50 mm directly in front. Create a field plot over X (+/-5mm) and Y (+/-5mm) for range (Z) = 50 mm. You may use ¼ or 1/8 symmetry in the field to reduce the total calculation time. You may also change the sampling in the frequency domain or use any other method you like to speed up the calculation so long as your approaches do not measurably impact the quality of the result. (16)

(You can use symmetry anywhere in solving these problems – but it clearly doesn't help much when steering off axis)



```
z_foc=50e-3;                                % Range direction focal distance
%theta_steer_vert_deg=0;
%theta_steer=(90-theta_steer_vert_deg)*pi/180;                      % Steer angle - will use this later
theta_steer=(0)*pi/180;
num_elems=32;                               % Number of array elements
fc=10e6;                                    % Center frequency
pitch=(.075e-3);
vel=1540;                                   % Speed of sound - all units MKS
fs=fc/64;                                   % Define a sampling frequency (not critical)
f=[fs:fs:8*fc];                             % Define an adequate frequency range (improve resolution in time domain)
i_cen=round(fc/fs);                         % Index of center frequency for single frequency calc
w=2*pi*f;                                   % Angular frequency radians
ns=length(f);                               % Number of samples
tdel=0e-6;                                  % Use a fixed time offset so that initial waveform is 0 for t<0 (not critical)
tdel2=1.0e-6;                               % Use a fixed time offset so that initial waveform is 0 for t<0 (not critical)
bw=30;                                      % Fractional bandwidth as percent
bw2=80;                                     % Fractional bandwidth as percent
sig=bw*fc/100;                              % Width of Gausian
gauss_pulse=exp(-pi*((f-fc)/sig).^2);       % Generate Gaussian pulse (frequency domain)
gauss_pulse=gauss_pulse.*exp(-j*w*tdel);    % Apply time delay so 0 for t<0
%att= .00005*f;
att=0;
%%
weight=ones(num_elems,1);                   % Define the weighting function (you can change this)
%weight=hann(num_elems);

x_pts=[0:0.2:5].*1e-3;
y_pts=[0:0.2:5].*1e-3;
z_pts=[50].*1e-3;
%num_elems=64;
x_elem=zeros(num_elems,1);
y_elem=zeros(num_elems,1);
foc_del=zeros(num_elems);
for i=1:num_elems
    x_elem(i)=((i-1)-(num_elems-1)./2).*pitch;  % Calculate locations of each array element]
    for k=1:num_elems
        y_elem(k)=((k-1)-(num_elems-1)./2).*pitch;
        foc_del(i,k) =  (sqrt(((x_elem(i))^2)+(y_elem(k)^2)+(z_foc^2)))./vel;
    end
end

field_val= zeros(length(x_pts),length(y_pts));
field_val_fc= zeros(length(x_pts),length(y_pts));
field_db= zeros(length(x_pts),length(y_pts));
field_fc_db= zeros(length(x_pts),length(y_pts));

    for i=1:length(x_pts)
        for n=1:length(y_pts)
        sum_pulse=zeros(size(gauss_pulse));                 % Initialize sum of wavef
            for k=1:num_elems
                for m=1:num_elems
                r=sqrt(((x_elem(k)-x_pts(i))^2)+((y_elem(m)-y_pts(n))^2)+(z_pts^2));
                prop_del = (r./vel);
                  sum_pulse=sum_pulse+weight(k).*10.^(-att.*r./20).*gauss_pulse.*...
                    exp(-sqrt(-1)*w.*(prop_del-foc_del(k,m)));
                end
            end

        sum_pulse_t=real(ifft(sum_pulse));                  % Convert to time domain

        field_val(i,n)= max(abs(hilbert(sum_pulse_t)));     % Field values in beamplo
        end
    end
     field_val(:,:)=field_val(:,:)./max(max(field_val(:,:)));     % Normalize by peak
     field_db(:,:)=20*log10(field_val(:,:));                 % Convert to dB

%%
field_db=[flip(field_db(:,1:end-1),2),field_db];
field_db=[flip(field_db(1:end-1,:),1);field_db];
x_ptsp=[-5:0.2:5].*1e-3;
y_ptsp=[-5:0.2:5].*1e-3;
figure;
contour(y_ptsp*1e3,x_ptsp*1e3,field_db,[-30 -20 -15 -12 -9 -6 -3]);
title('Contour Field')
xlabel('Depth (mm)')
ylabel('Azimuth (mm)')
figure
mesh(y_ptsp*1e3,x_ptsp*1e3,field_db);
title('Mesh Field')
xlabel('Depth (mm)')
ylabel('Azimuth (mm)')
zlabel('Signal (dB)')
```

8. Single planar sources – model these as a finely sampled set of sources are simultaneous excited by a common function

In this question, assume no attenuation effects.

Planar
Define a 2D array of Huygen's sources spaced at 30 micron intervals
5 MHz
Assume a 1cm diameter flat piston transducer
Set 2D array source weights to all zeros
Define a mask of array weights equal one within the circular area of the transducer – i.e. R< 0.5 cm = 1.0
In at least one of the following examples, include a factor for spherical spreading – i.e. the energy from a point source is spread across the surface of a sphere as it propagates outward in all directions. Thus, it is spread over pi r^2. Therefore, your contribution for each source can be divided by the distance used in "prop_del" ^2.

```
z_foc=50e-3;                                % Range direction focal distance
theta_steer=(0)*pi/180;
num_elems=64;                              % Number of array elements
fc=5e6;                                    % Center frequency
pitch=(.075e-3);
vel=1540;                                   % Speed of sound - all units MKS
fs=fc/64;                                   % Define a sampling frequency (not
f=[fs:fs:8*fc];                             % Define an adequate frequency rang
i_cen=round(fc/fs);                         % Index of center frequency for si
w=2*pi*f;                                   % Angular frequency radians
ns=length(f);                               % Number of samples
tdel=0e-6;                                  % Use a fixed time offset so that
tdel2=1.0e-6;                               % Use a fixed time offset so that
bw=30;                                      % Fractional bandwidth as percent
bw2=80;                                     % Fractional bandwidth as percent
sig=bw*fc/100;                              % Width of Gausian
gauss_pulse=exp(-pi*((f-fc)/sig).^2);       % Generate Gaussian pulse (frequen
gauss_pulse=gauss_pulse.*exp(-j*w*tdel);    % Apply time delay so 0 for t<0
%att= .00005*f;
att=0;
%% set x_pts to [0] for axis view

x_pts=[-10:0.2:10].*1e-3;
%y_pts=[-5:0.2:5].*1e-3;
y_pts=0;
%x_pts=0;
z_pts=[10:.1:80].*1e-3;
x_elem=[-5:0.1:5].*1e-3;
y_elem=[-5:0.1:5].*1e-3;
w_elem=zeros(length(x_elem),length(y_elem));
foc_del=zeros(num_elems);
for i=1:length(x_elem)
    %x_elem(i)=((i-1)-(num_elems-1)./2).*pitch;  % Calculate locations of each
    for k=1:length(y_elem)
        %y_elem(k)=((k-1)-(num_elems-1)./2).*pitch;
        foc_del(i,k) =  (sqrt(((x_elem(i))^2)+(y_elem(k)^2)+(z_foc^2)))./vel;
        if sqrt(((x_elem(i))^2)+(y_elem(k)^2))<=5e-3
            w_elem(i,k)=1;
        end
    end
end
```

```
    weight=ones(length(x_elem),length(y_elem));
    %weight=hann(length(x_elem))*hann(length(y_elem))';

    field_val= zeros(length(z_pts),length(x_pts),length(y_pts));
    field_val_fc= zeros(length(z_pts),length(x_pts),length(y_pts));
    field_db= zeros(length(z_pts),length(x_pts),length(y_pts));
    field_fc_db= zeros(length(z_pts),length(x_pts),length(y_pts));
for q=1:length(z_pts)
        fprintf('%d ',q)                        % Short piece of code to provide record of progress
    if (q/20)==round(q/20)
        fprintf('\n')
    end
    for i=1:length(x_pts)
        for n=1:length(y_pts)
        sum_pulse=zeros(size(gauss_pulse));                 % Initialize sum of waveforms to zero before s
            for k=1:length(y_elem)
                for m=1:length(y_elem)
                r=sqrt(((x_elem(k)-x_pts(i))^2)+((y_elem(m)-y_pts(n))^2)+(z_pts(q)^2));
                prop_del = (r./vel);
                  sum_pulse=sum_pulse+(w_elem(k,m)*weight(k,m)*r^-2).*10.^(-att.*r./20).*gauss_pulse.*...
                    exp(-sqrt(-1)*w.*(prop_del-foc_del(k,m)));
%                   sum_pulse=sum_pulse+(w_elem(k,m)*weight(k,m)).*10.^(-att.*r./20).*gauss_pulse.*... % S
%                     exp(-sqrt(-1)*w.*(prop_del-foc_del(k,m)));
                end
            end

        sum_pulse_t=real(ifft(sum_pulse));                  % Convert to time domain

        field_val(q,i,n)= max(abs(hilbert(sum_pulse_t)));     % Field values in beamplot - Hilbert finds e
        field_val_fc(q,i,n)=abs(sum_pulse(i_cen));            % Magnitude of summed single frequency (cent
        end
    end
    %% Typical normalization
    %       field_val(q,:,:)=field_val(q,:,:)./max(max(field_val(q,:,:)));     % Normalize by peak value at ea
    %       field_db(q,:,:)=20*log10(field_val(q,:,:));                  % Convert to dB
    %       field_val_fc(q,:,:)=field_val_fc(q,:,:)./max(max(field_val_fc(q,:,:)));
    %       field_fc_db(q,:,:)=20*log10(field_val_fc(q,:,:));
end
    %% Axis FC normalization
          field_val=field_val./max(max(max(field_val)));     % Normalize by peak value at each range
          field_db(q,:,:)=20*log10(field_val(q,:,:));                % Convert to dB
          field_val_fc(:,:)=field_val_fc(:,:)./max(max(field_val_fc(:,:)));
          field_fc_db=20*log10(field_val_fc);
end
%% Axis FC normalization
      field_val=field_val./max(max(max(field_val)));     % Normalize by peak value at each range
      field_db(q,:,:)=20*log10(field_val(q,:,:));                  % Convert to dB
      field_val_fc(:,:)=field_val_fc(:,:)./max(max(field_val_fc(:,:)));
      field_fc_db=20*log10(field_val_fc);
%%
figure;
contour(x_pts*1e3,z_pts*1e3,squeeze(field_db(:,:)),[-30 -20 -15 -12 -9 -6 -3]);
%contour(squeeze(field_db(:,:,26)),[-30 -20 -15 -12 -9 -6 -3]);
title('Contour Field')
ylabel('Range (mm)')
xlabel('Azimuth (mm)')
figure
mesh(x_pts*1e3,z_pts*1e3,squeeze(field_db(:,:)));
%mesh(squeeze(field_db(:,:,26)));
title('Mesh Field')
ylabel('Range (mm)')
xlabel('Azimuth (mm)')
zlabel('Signal (dB)')
figure
mesh(x_pts*1e3,z_pts*1e3,squeeze(field_fc_db(:,:)));
%mesh(squeeze(field_db(:,:,26)));
title('Mesh Field at FC')
ylabel('Range (mm)')
xlabel('Azimuth (mm)')
zlabel('Signal (dB)')
figure
plot(z_pts*1e3,squeeze(field_fc_db(:,:)));
title('Center axis at FC')
xlabel('Range (mm)')
ylabel('Signal (dB)')
```

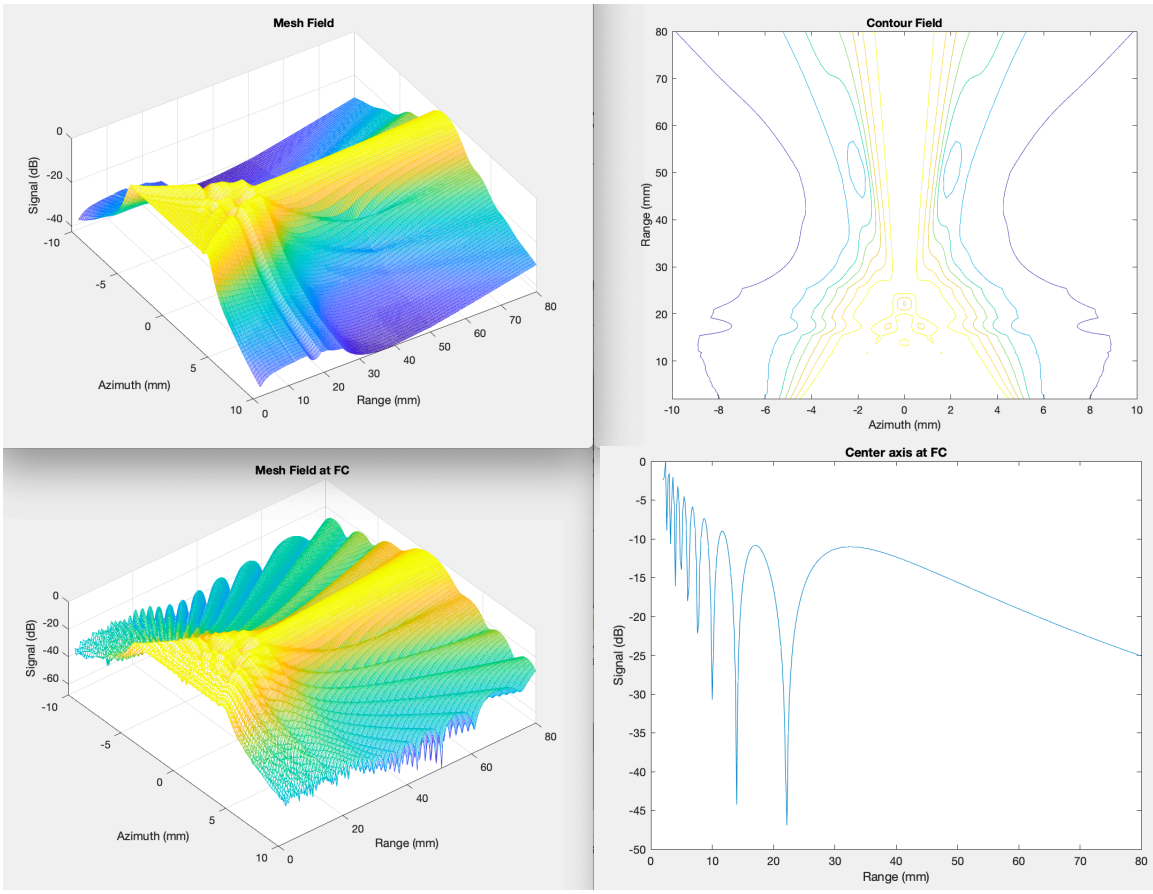
Create a meshplot (in dB domain) +/- 10 mm in azimuth (x) and 2-60 mm in range (Z) for each of:

a) Gaussian pulse 30% -6dB fractional BW (10)

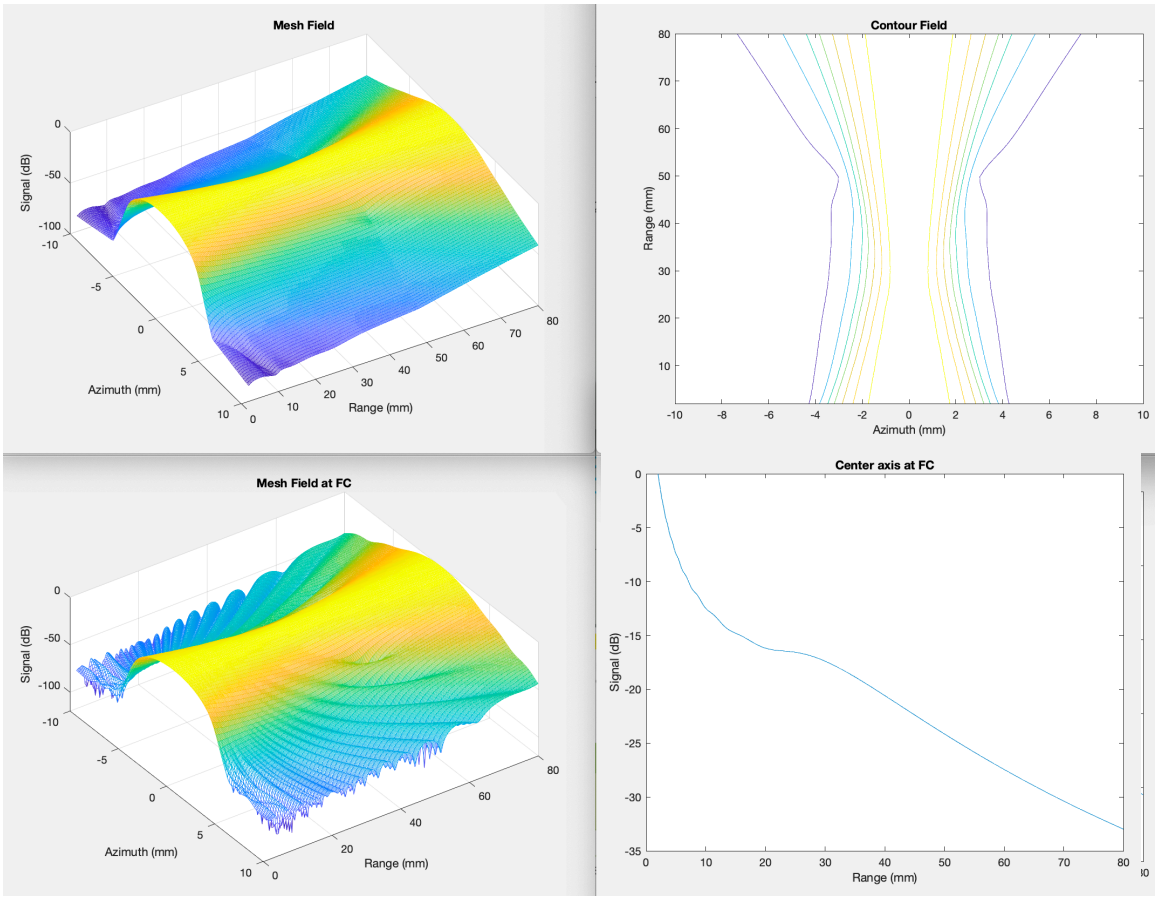
See part b

b) CW at 5 MHz – per in class discussion. Evaluate at single frequency.

a. For this case, also produce a finely sampled plot along the central axis (X=Y=0) to Z=80 mm (5)

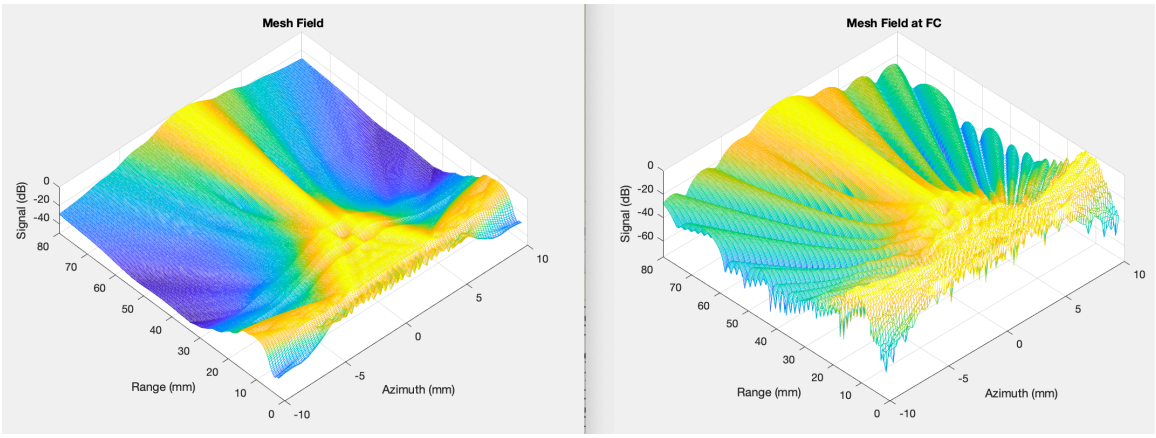


c) Repeat a) and b) after imposing a Hann apodization function across the aperture with circular symmetry. Very the reduction in sidelobe activity. (10)



The side lobes has been outright removed in the normal mesh field but reduced in the FC mesh. Also the axis FC plot does not have the same kind of interference as in part b.

d) Experiment with increasing source spacing (i.e > 30 microns). Verify that source spacing is more critical in near field. (10)



I set the spacing to 50 microns and it resulted in a near field with increased interference from the part a plots but in the far field it looks approximately the same. Also there are side lobes now.