EECS 316

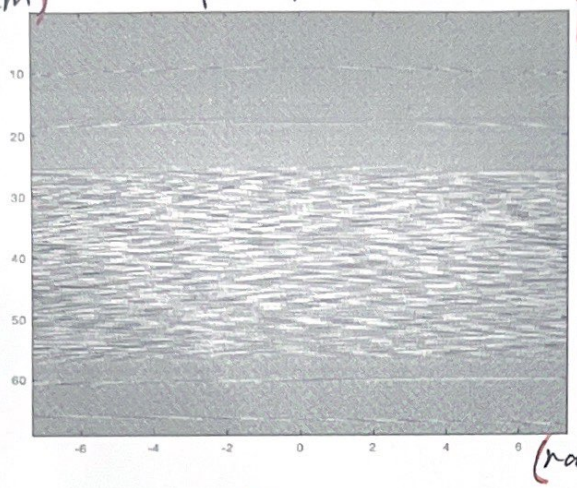
us project , YUZHOU CHEN.

YZC.

with partner YIHANG REN.

Part A: the answer with part N.

Plot for part A.



Part B:

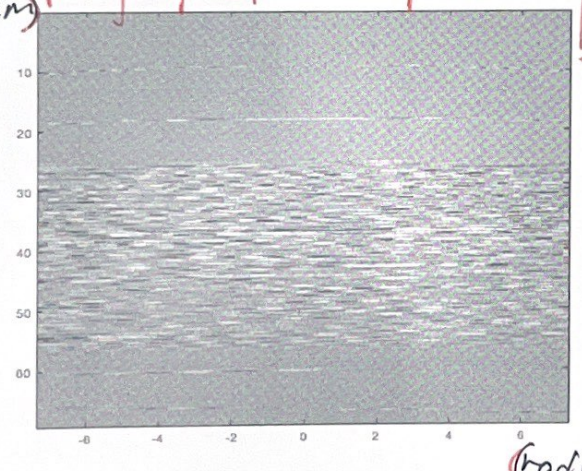
$$\#\text{beams} = \frac{\max(\sin\theta) - \min(\sin\theta)}{\Delta(\sin\theta)}$$

$$= \frac{\frac{\sqrt{2}}{2} - \left(-\frac{\sqrt{2}}{2}\right)}{\lambda/4a}$$

$$= \frac{4\sqrt{2}a}{\lambda} = 133$$

θ = [$\frac{\lambda}{2}$ divided by 133 times equally. every piece is $\frac{\pi}{2\times 133}$ length, start from $-\frac{\pi}{4}$ to $\frac{\pi}{4}$.

Plot for part F: real part.



Part C:

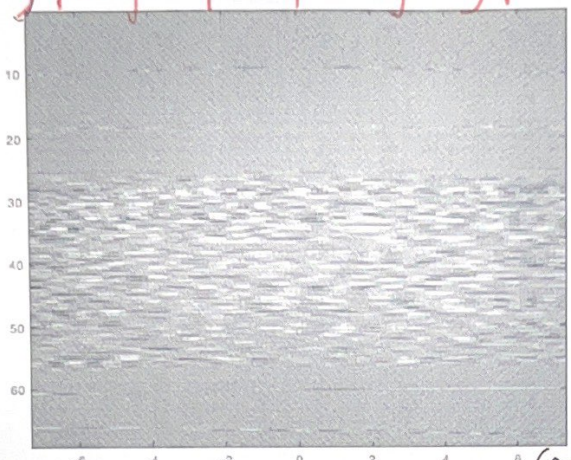
$$\tau_n' = \tau'(x_n, r_0, \theta_0) = \frac{x_n \sin\theta_0}{c} + \frac{x_n^2 \cos^2\theta_0}{2cr_0}$$

$\Leftrightarrow$ mat.

τ_n' and mat are 3-dimensional arrays.

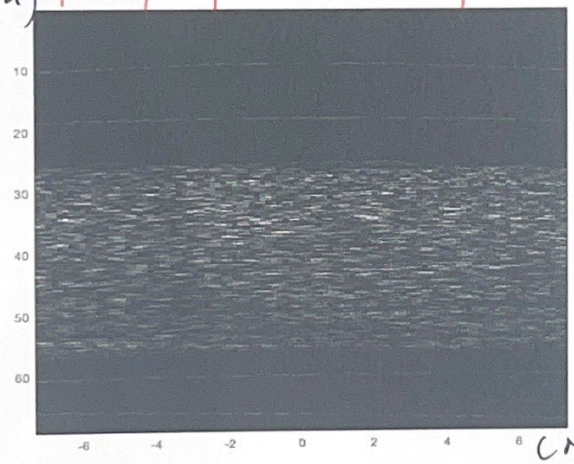
Part D:

(mm) Plot for part F: imaginary part.



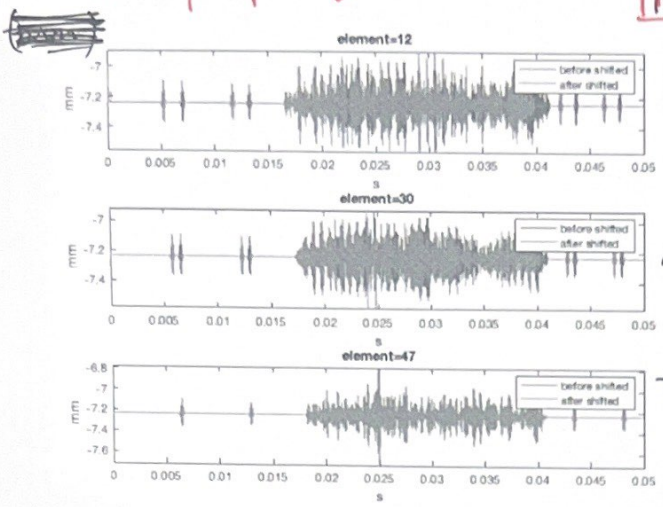
(radius)

(mm) Plot for part F: whole part.



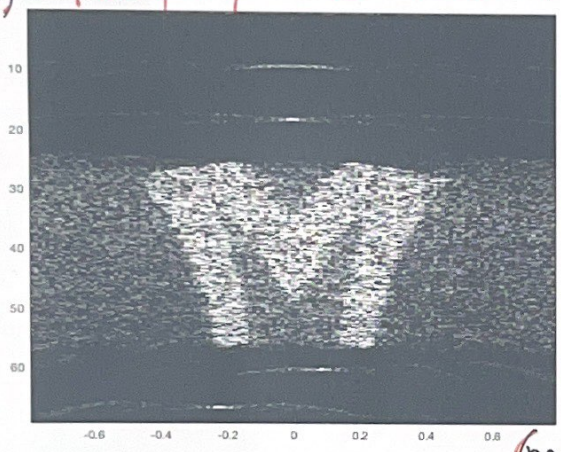
(radius)

Plot for part D



Part D. : As the time vector shown in the plot, values are collected from element 12, 30 and 47. As the number of element increase, the signal before and after shifting are increasingly become equal. And in the same time, the pulas and waves are add constructly together.

(mm) Plot for part E. (As data).



(radius)

Part E : By using equation: $V(t) = \frac{1}{N}\sum_{n=1}^{n} V_n(t+[\tau'])$, the plot was generated. It can be seen as a letter "M", but still need some ajustion to see much more clearly.

Part F By implement the complex number and rectangle filter, from plot real part and imaginary part, we can see that the signal from r=26 to r=60 looks like the mixture of gray and white, others part are totally gray except r=9 and r=18, which are 2 solitary reflectors ~~located at~~. That is almost I expect.

Part H Yes, by checking elements 12, 30 and 47, the signals at location r=9 and r=18 looks added constructively. And it seems like some reflectors at r=62 and r=68.

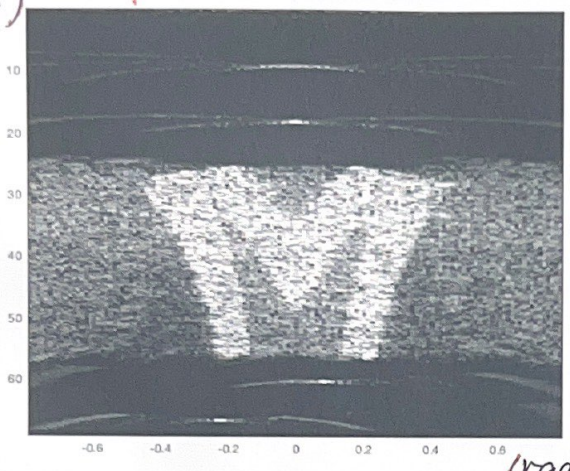
Part @G

phase term = $-2\pi f_0 \tau_i$

the equation above was implemented for this part.

This phase term will ensures that the wavefronts will add constructively for the focal point and deaostructively for wavefront coming from other directions.

(mm) Plot for Part I : rsdata_i



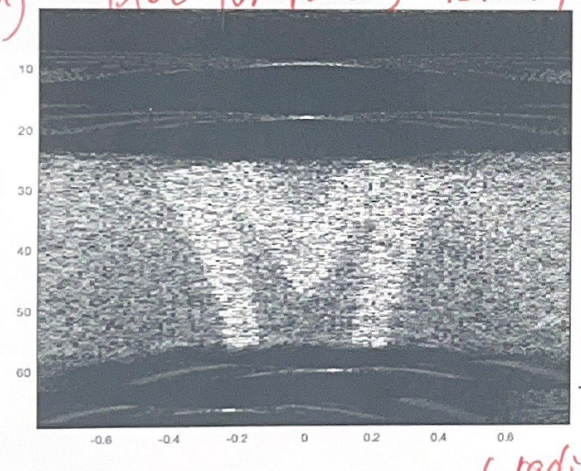
(radius)

Part I By implement equation

$$b(t) = \frac{1}{N}\sum_{n=1}^{N} b_n(t+[\tau_n])\, e^{\{\mp i\omega_0[\tau_n]\}},$$

rsdata_i can be ploted. From the plot, it is not better than figure in part E, because of the introduction of rectangle filter.

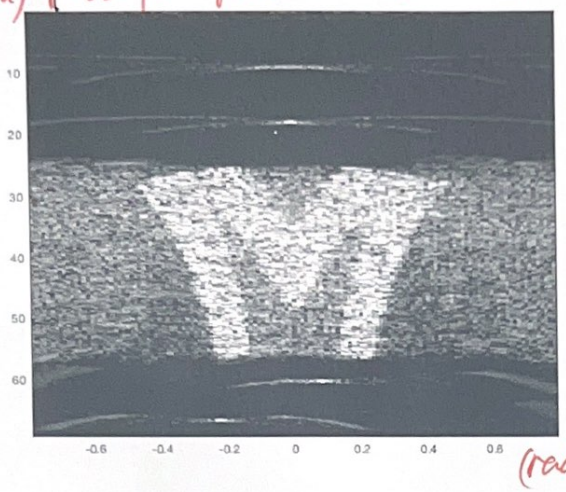
(mm) Plot for Part J : rsdata_j



(radius).

Part J By using only odd number elements, it is obviously the plot looks even worse. It is due to the insufficient of the information collection from ~~foser~~ lower number elements.

(mm) Plot for Part k: rsdata_k.



Published with MATLAB® R2022a

Part k. From the plot, I can have the ~~sum~~ following summary.

① the artifact ~~in B~~ located at r=9 and r=18 ~~and~~ are mirror image because of ~~to~~ the constructively symmetry.

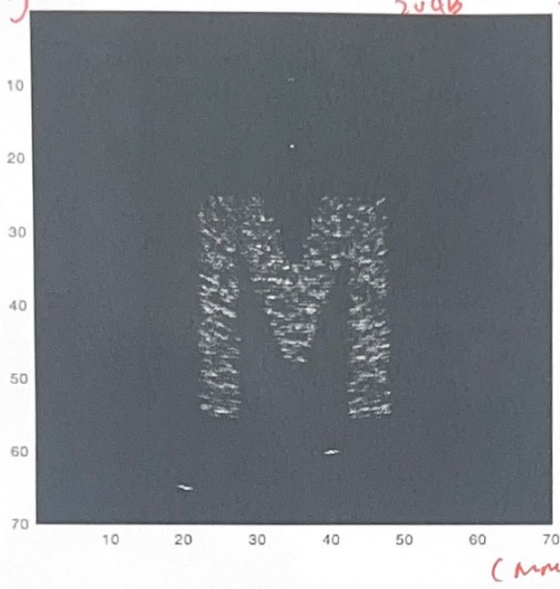
②. Refration: ~~from the~~ causing the distortion of the ~~image~~ image.

③. Acoustic noise: causing the speckle pattern in the image surrounding letter "M".

④ ~~Clutter~~: the low amplitude echoes obscure the structure in the image surrounding letter "M".

⑤ Reverberation: the noise below letter "M"

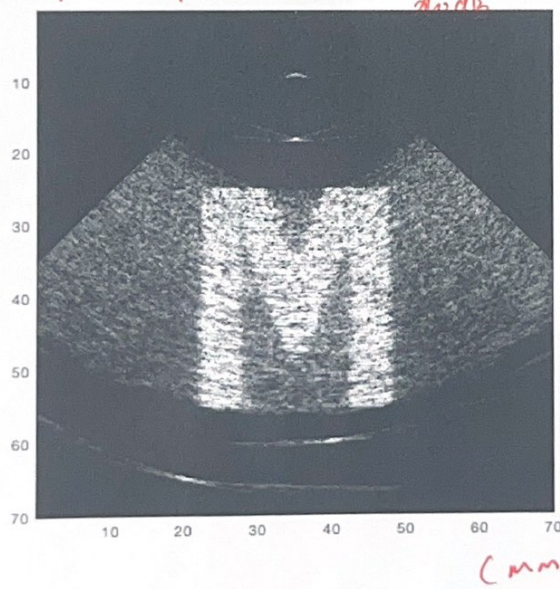
Part L : rsdata · 20dB



Part L

By using function "interp2" and "meshgrid", the letter "M" in the fitful shape and size was generated. The plot in 20dB looks better than in 40dB, without a lot of noise.

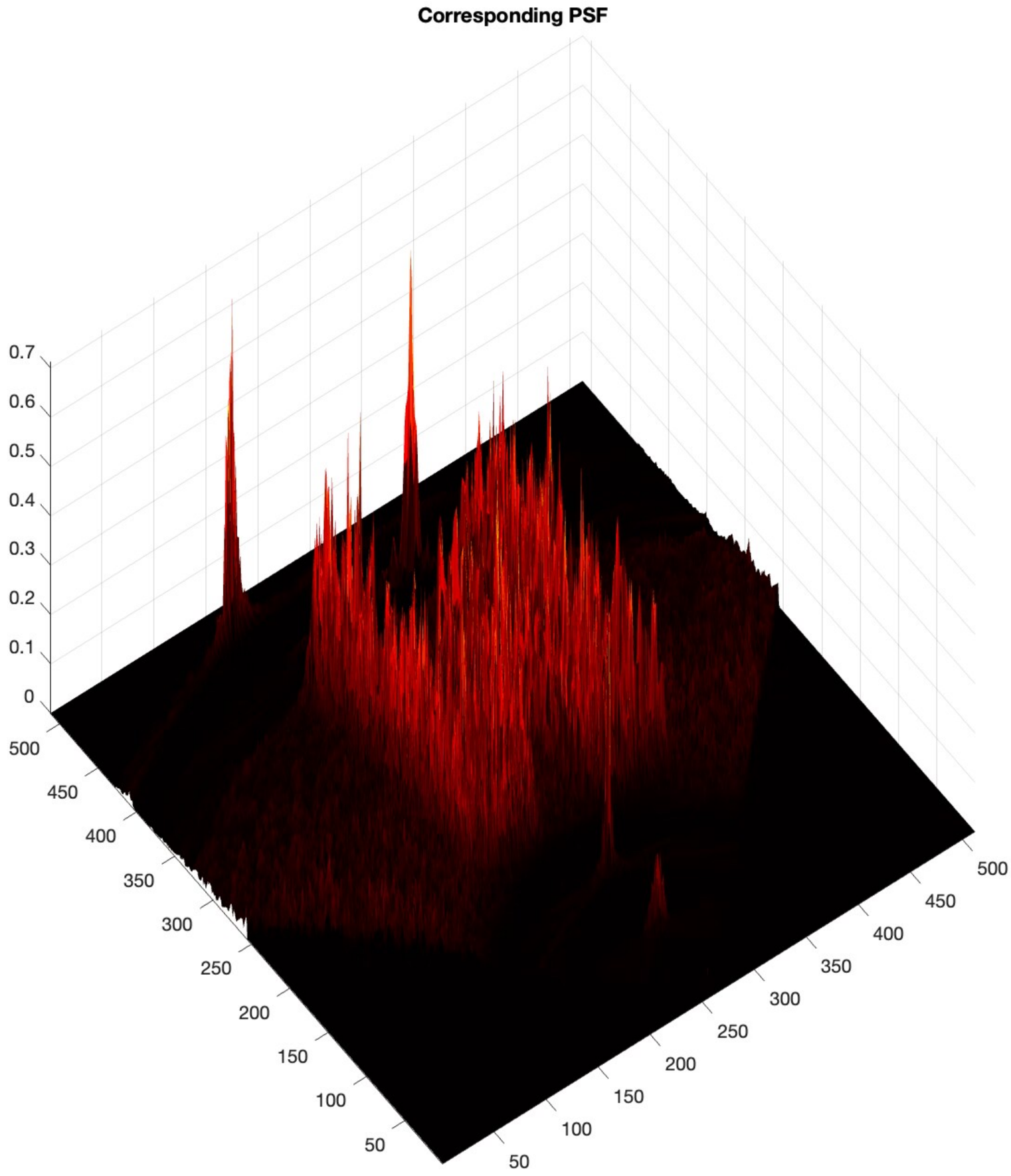
Part L: rsdata 40dB



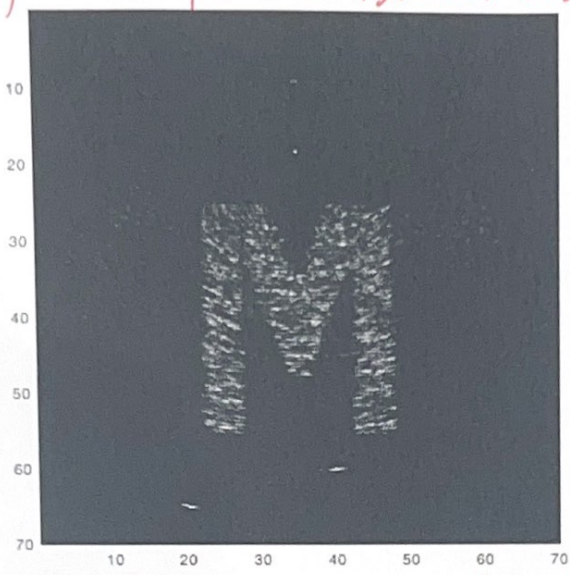
Part M

The object is the letter "M" and 2 mirror reflector ~~of~~. And about ~~de~~ quality of image e, i, j, k, as we disscussed from previous question, the plot of rsdata-j is the worst because of ~~the ineff~~ low number of elements, and the plot of rsdata ~~is~~ looks best, without too much noise but with clear shape and size.

Point spread funtion looks like a letter "M" with echoes and two mirror ~~ref~~ reflector (the shape) from the ~~value pos~~ view of point which is parrellel to the ~~plot surface~~ x, y axis ~~but~~ (see picture).

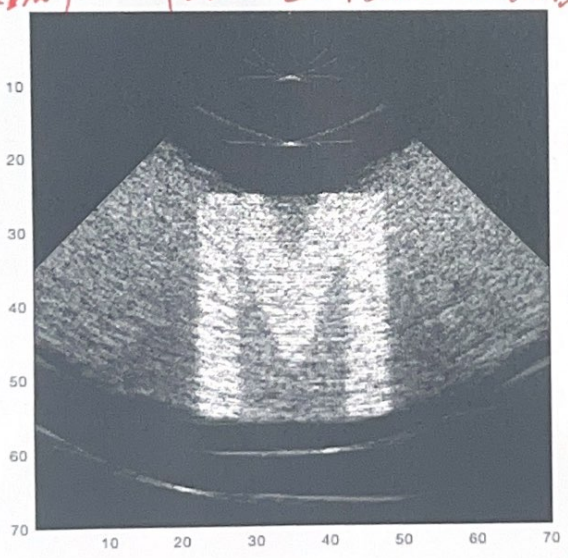

Corresponding PSF
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
450
400
350
300
250
200
150
100
50
50
100
150
200
250
300
350
400
450
500

(mm) Part L rsdata_i 2dB



(mm)

(mm) Part L: rsdata_i 60dB



(mm)

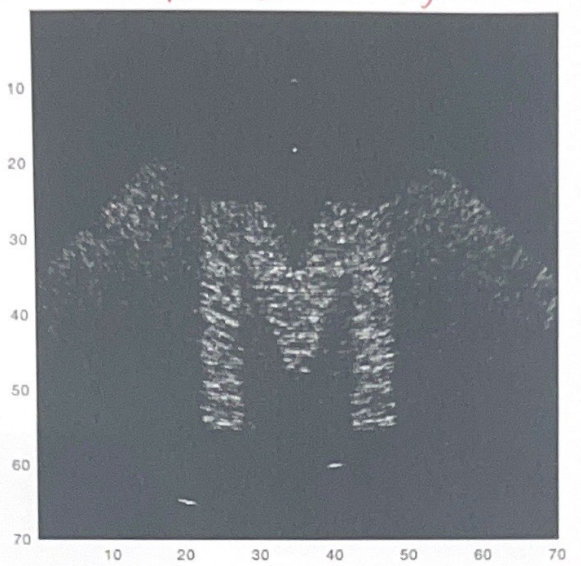
Part A and Part A

The shape can be seen as a rectangle in Part A, but which is a 90° sector in Part L, which is ~~the~~ normal shape and size. Part L canceled ~~the~~ error of angel. ~~So of the~~ and curvature.

In part A, the curvature of ~~the~~ mirror (r=9 and r=18) and echo (r=62 and r=68) are ~~two~~ 4 polyline ~~curve~~ ~~curvature~~ which are a quarter of circle line. ~~they~~ These symbol can be seen obviously ~~in the~~ between the plot A and plot L.

(mm)

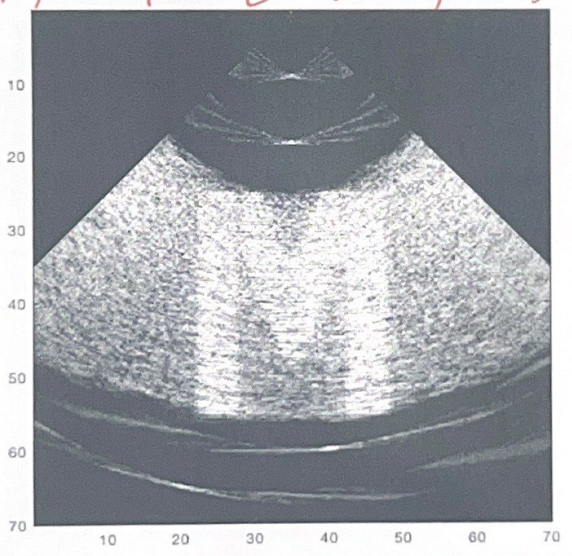
Part L: RSdata_j 20dB



(mm)

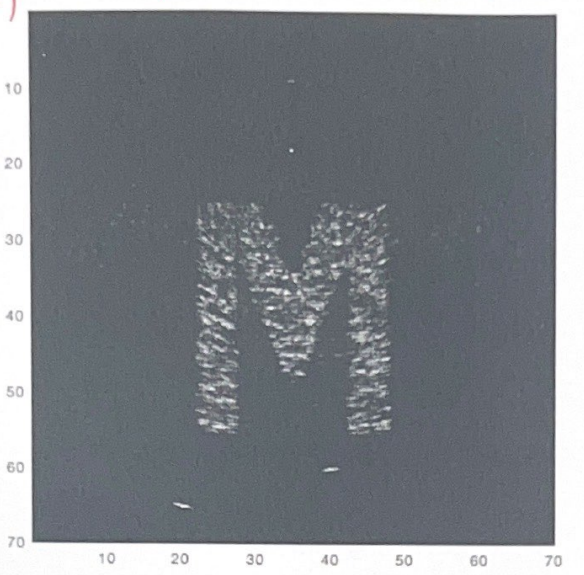
(mm)

Part L: RSdata_j 40dB

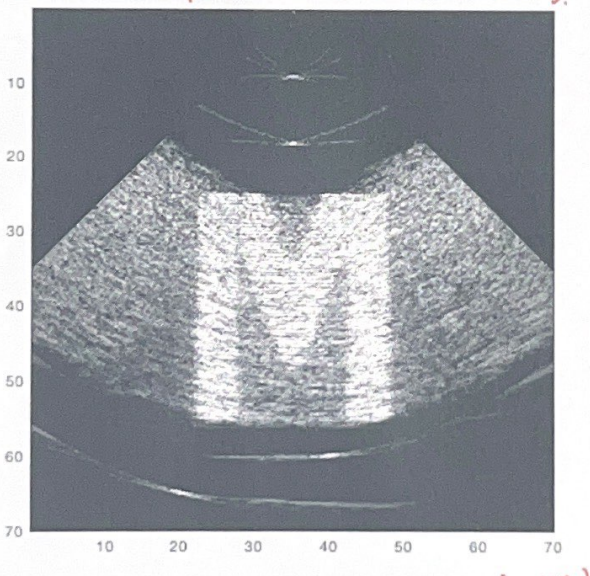


(mm)

Part L: rsdata_k. 20dB



Part L: rsdata_k 40dB



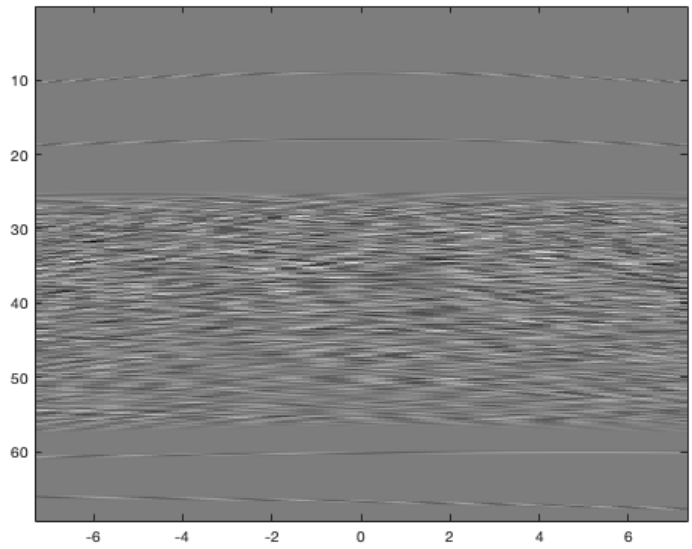
Published with MATLAB® R2022a

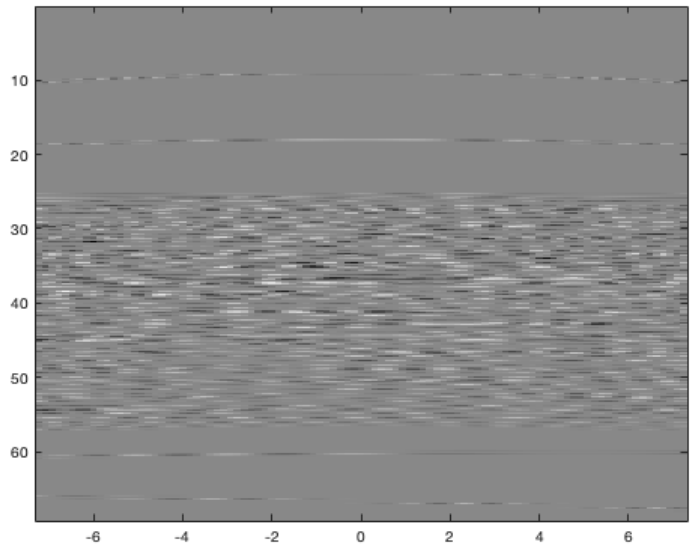
```
%       template_beamform.m
%       Ultrasound beamforming project template
%
%       This script requires the variables
%       Data [ntime, nelem]     RF signals from each array element
%       f0                      Transducer center frequency (MHz)
%       fs                      sampling frequency (MHz)
%       c                       speed of sound (mm/usec)
%       dx                      transducer element spacing (mm)
%
% Get needed variables
%
clear all clc
load data23;
f0 = 5;         % MHz
fs = 20;        % MHz
c = 1.54;       % mm/us
%
% Output Array Parameters
%
% Determine the array spacing dx in mm
dx = c/f0/2;
deltat=1/fs;
[ntime, nelem] = size(Data);            % # time samples, # array elements
disp(sprintf('f0=%g MHz, deltat=%g usec, dx=%g mm', f0, deltat, dx))
disp(sprintf('# of Time Samples=%g,  # of Array Elements=%g',ntime,nelem))

%

% --> QUESTION a. <--
% Make a "wavefield" plot of the raw Data
% Comment this out while you debug other parts of your program
%
x_n=linspace(-(nelem-1)*dx/2,(nelem-1)*dx/2,nelem);
t = linspace(1,ntime,ntime )*deltat;
r = c*t/2;
figure(1)
showimage3(Data, 1,-1,x_n,r)  % plotting the transpose here...time along x, tranducer x along y.
disp 'hit key', pause

%
% --> QUESTION b. <--
% Compute the number of total beams and beam spacing
% I used variables called
%       nbeam = number of beams
%       sin_theta = vector of beam positions
%
nbeam = round(2*sqrt(2)*(nelem-1)*dx*f0/c);
disp(sprintf('nbeam = %g', nbeam));
sin_theta = sin (linspace(-pi/4,pi/4,nbeam));

% --> QUESTION f. <--
databb = zeros(ntime,nelem);    % Phase-Shifted Baseband data

% code here to demodulate and write into databb
for ie = 1:nelem
    for it=1:ntime
        databb(it,ie) = fft(Data(it,ie)*exp(2i*pi*f0*t(it)));
        if databb(it,ie) < 5
            databb(it,ie) = ifft(databb(it,ie));
        end

    end
end
% Plot and Show data

figure(2)
showimage3(real(databb), 1,-1,x_n,r)
disp(sprintf('real part'))
disp 'hit key', pause

figure(3)
showimage3(imag(databb), 1,-1,x_n,r)
disp(sprintf('imag part'))
disp 'hit key', pause
%plot...
figure(4)
showimage3(abs(databb), 1,-1,x_n,r)
```

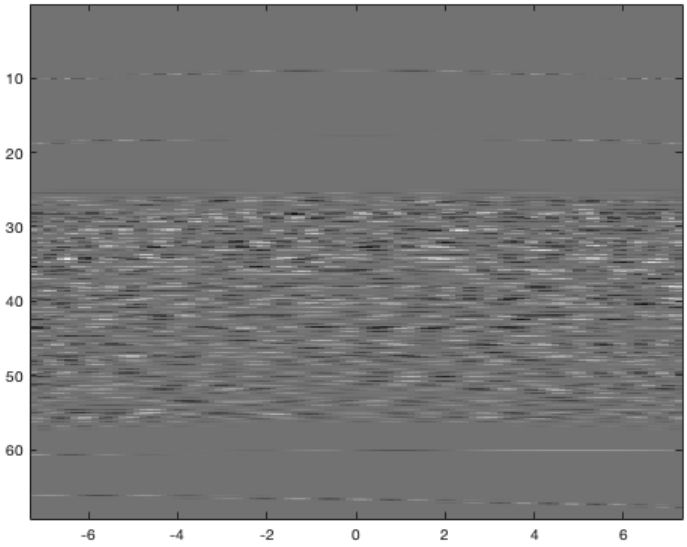
```
disp(sprintf('databb'))
disp 'hit key', pause

%
% Create room for answers and precompute information
%       This section merely makes matlab allocate memory before doing
%       any other processing. It is not a necessary section, but will
%       make Matlab run a little faster since it doesn't have to reallocate
%       space every time you add another beam
%
rsdata = zeros(ntime,nbeam);% r-sin(theta) data buffer
rsdata_i = zeros(ntime,nbeam);
rsdata_j = zeros(ntime,nbeam);
rsdata_k = zeros(round(ntime/3),nbeam);
bbshift = zeros(ntime,nelem);   % Phase-Shifted Baseband data
theta_1=linspace(-pi/4,pi/4,nbeam);
x_n=linspace(-(nelem-1)*dx/2,(nelem-1)*dx/2,nelem);
tau=zeros(ntime,nelem,nbeam);
t=linspace(0,ntime/fs,ntime);
mt=zeros(ntime,nelem,nbeam);
phase=zeros(ntime,nelem,nbeam);
%
% Repeat for every beam
%
for ib=1:nbeam
        %disp(sprintf('Beam %d of %d', ib, nbeam))

% --> QUESTION c, d, g, h <--
% For the current beam, compute time delays (or delayed samples)
% and phase rotations (if needed) for each channel (i.e. transducer
% element) as a function of range, as well as gain corrections.
    for ie=1:nelem
        for it=1:ntime
%--> QUESTION c <--
            tau(it,ie,ib) = -x_n(ie)*sin(theta_1(ib))/c + (x_n(ie)*cos(theta_1(ib)))^2/2/c/r(it);
            mt(it,ie,ib)=floor(tau(it,ie,ib)*fs);
%--> QUESTION g <--
            phase(it,ie,ib)= -2*pi*f0*tau(it,ie,ib);
            if (td <= 1800) && (td > 0)
%--> QUESTION d <--
                bbshift(it,ie) = Data(it + mt(it,ie,ib),ie);
%--> QUESTION e <--
                rsdata(it,ib) = rsdata(it,ib) + Data(it + mt(it,ie,ib),ie)/nelem;
%--> QUESTION i <--
                rsdata_i(it,ib) = rsdata_i(it,ib) + databb(it + mt(it,ie,ib),ie)*exp(1i*phase(it,ie,ib))/nelem;
%--> QUESTION j <--
                if mod(ie,2) == 1
                    rsdata_j(it,ib) = rsdata_j(it,ib) + databb(it + mt(it,ie,ib),ie)*exp(1i*phase(it,ie,ib))/round(nelem/2);
                end
%--> QUESTION k <--
                if mod(it,3) == 0
                    rsdata_k(round(it/3),ib) = rsdata_k(round(it/3),ib) + databb(it + mt(it,ie,ib),ie)*exp(1i*phase(it,ie,ib))/nelem;
                end
            end
        end
    end
end

figure(5)
subplot(3,1,1)
plot(t/ntime,Data(:,12)*dx-(nelem-1)*dx/2)
xlabel("s")
ylabel("mm")
hold on
plot(t/ntime,bbshift(:,12)*dx-(nelem-1)*dx/2)
legend("before shifted","after shifted")
title("element=12")
xlabel("s")
ylabel("mm")

subplot(3,1,2)
plot(t/ntime,Data(:,30)*dx-(nelem-1)*dx/2)
title("element=30")
xlabel("s")
ylabel("mm")
hold on
plot(t/ntime,bbshift(:,30)*dx-(nelem-1)*dx/2)
legend("before shifted","after shifted")
title("element=30")
xlabel("s")
```

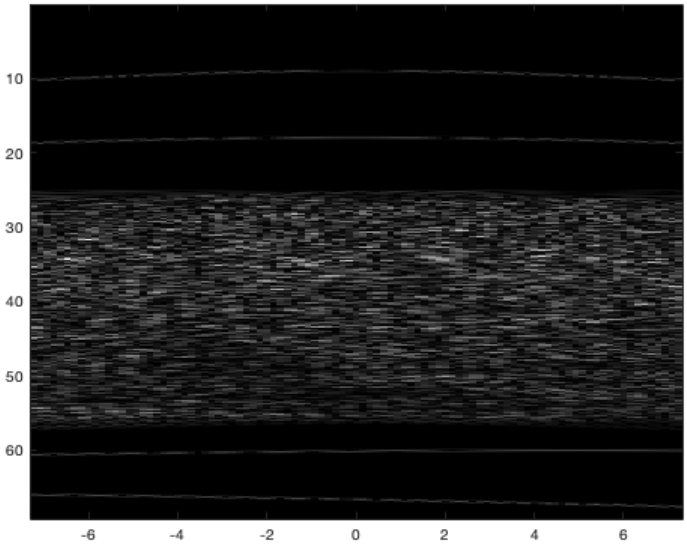
```
ylabel("mm")

subplot(3,1,3)
plot(t/ntime,Data(:,47)*dx-(nelem-1)*dx/2)
title("element=47")
xlabel("s")
ylabel("mm")
hold on
plot(t/ntime,bbshift(:,47)*dx-(nelem-1)*dx/2)
legend("before shifted","after shifted")
title("element=47")
xlabel("s")
ylabel("mm")


figure(6)
showimage3(abs(rsdata), 1, 40,theta_1,r)
disp(sprintf('rsdata'))
disp 'hit key', pause

figure(7)
showimage3(abs(rsdata_i), 1, 40,theta_1,r)
disp(sprintf('rsdata_i'))
disp 'hit key', pause

figure(8)
showimage3(abs(rsdata_j), 1, 40,theta_1,r)
disp(sprintf('rsdata_j'))
disp 'hit key', pause

figure(9)
showimage3(abs(rsdata_k), 1, 40,theta_1,3*r)
disp(sprintf('rsdata_k'))
disp 'hit key', pause
```

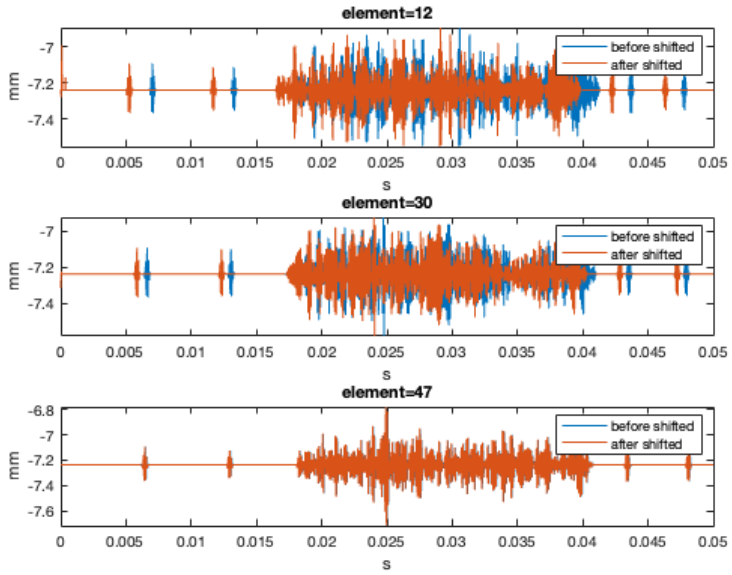
```
f0=5 MHz, deltat=0.05 usec, dx=0.154 mm
# of Time Samples=1800,  # of Array Elements=95
hit key
nbeam = 133
real part
hit key
imag part
hit key
databb
hit key
rsdata
hit key
rsdata_i
hit key
rsdata_j
hit key
rsdata_k
```

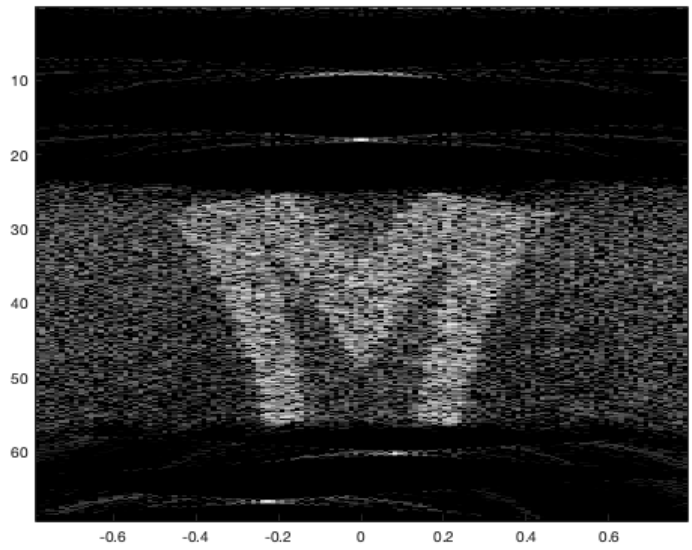

10
20
30
40
50
60
-6
-4
-2
0
2
4
6

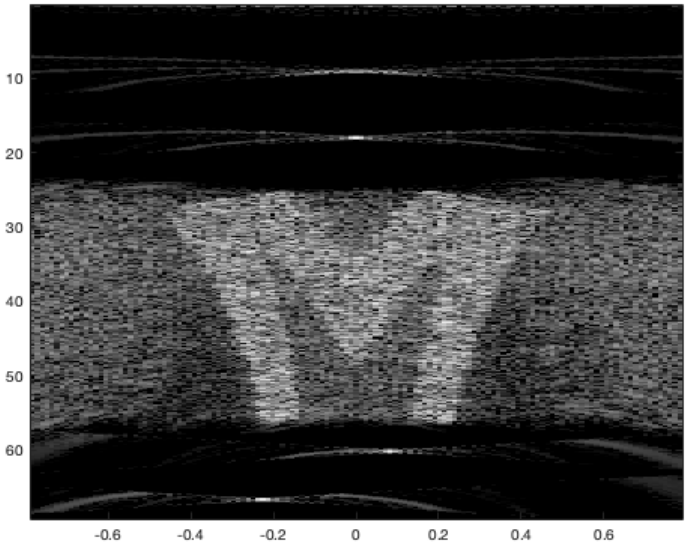

10
20
30
40
50
60
-6
-4
-2
0
2
4
6

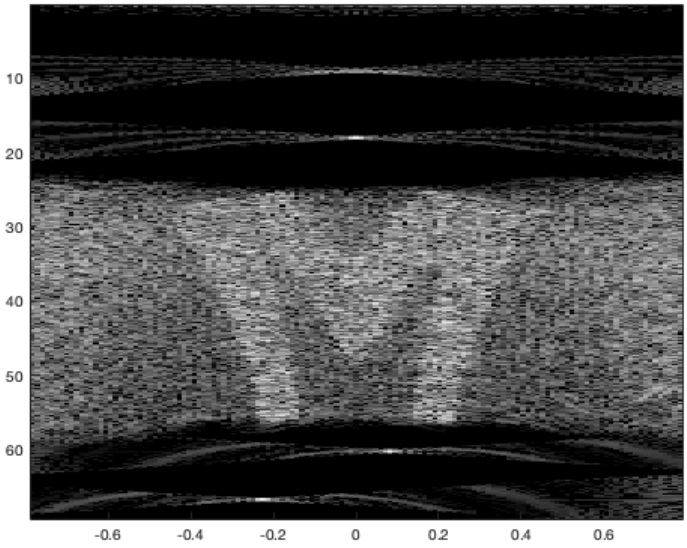

10
20
30
40
50
60
-6
-4
-2
0
2
4
6

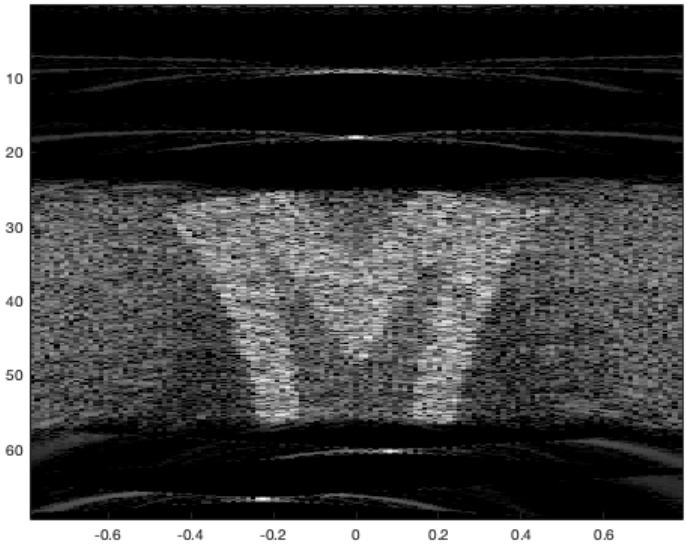

10
20
30
40
50
60
-6
-4
-2
0
2
4
6


element=12
before shifted
after shifted
mm
s
element=30
before shifted
after shifted
mm
s
element=47
before shifted
after shifted
mm
s



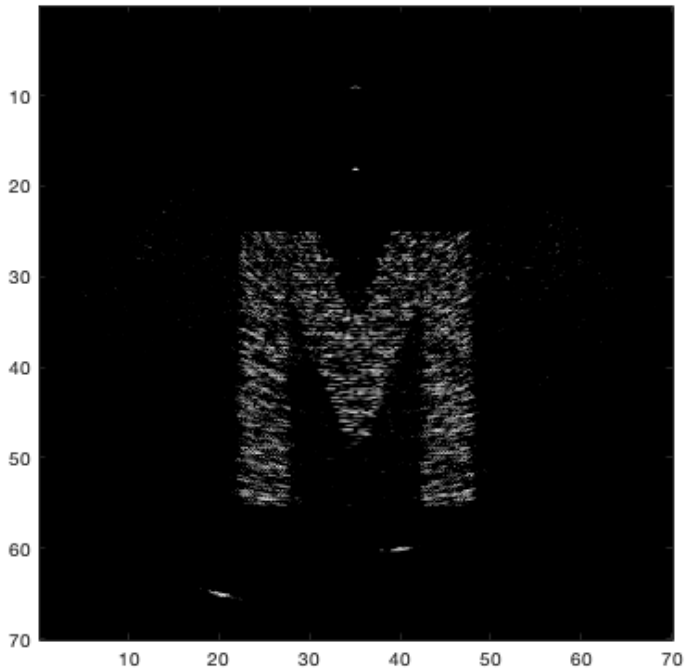

10
20
30
40
50
60
-0.6
-0.4
-0.2
0
0.2
0.4
0.6

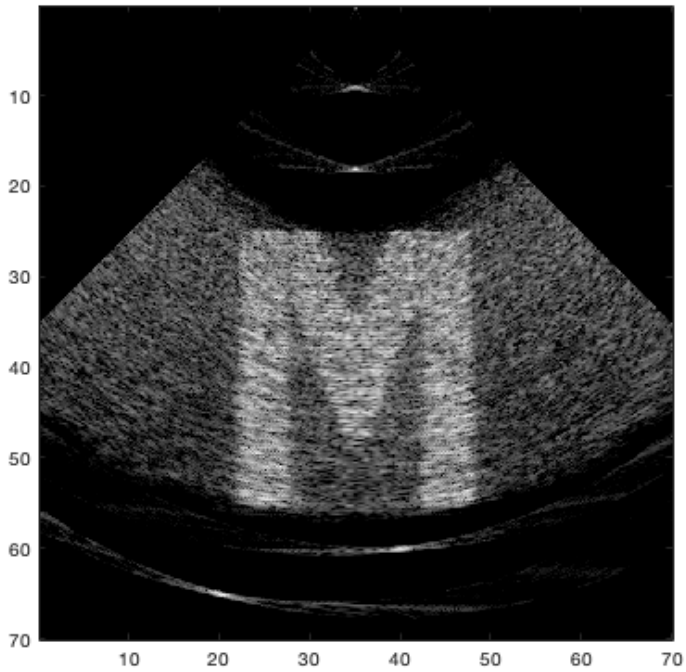

10
20
30
40
50
60
-0.6
-0.4
-0.2
0
0.2
0.4
0.6

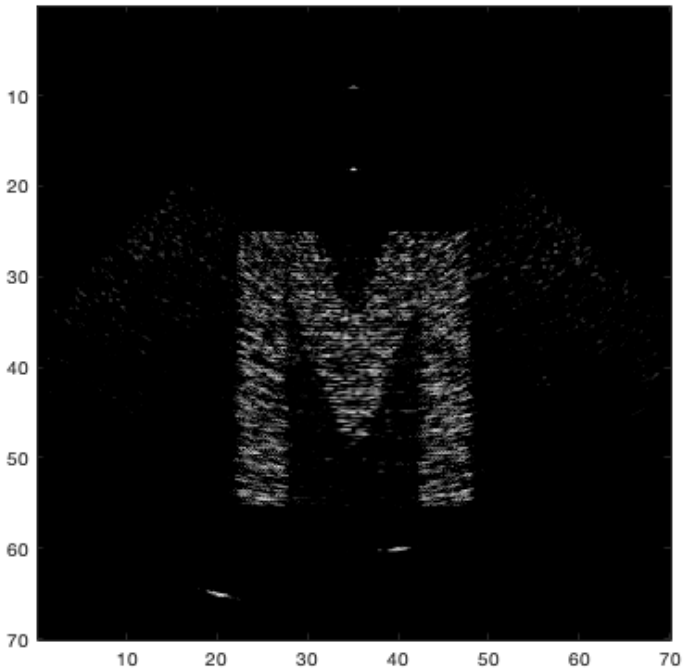


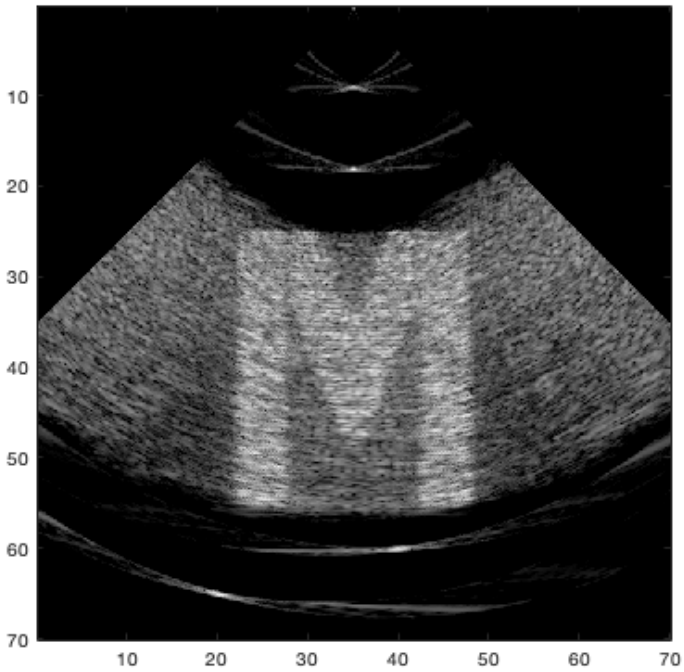
Published with MATLAB® R2022a

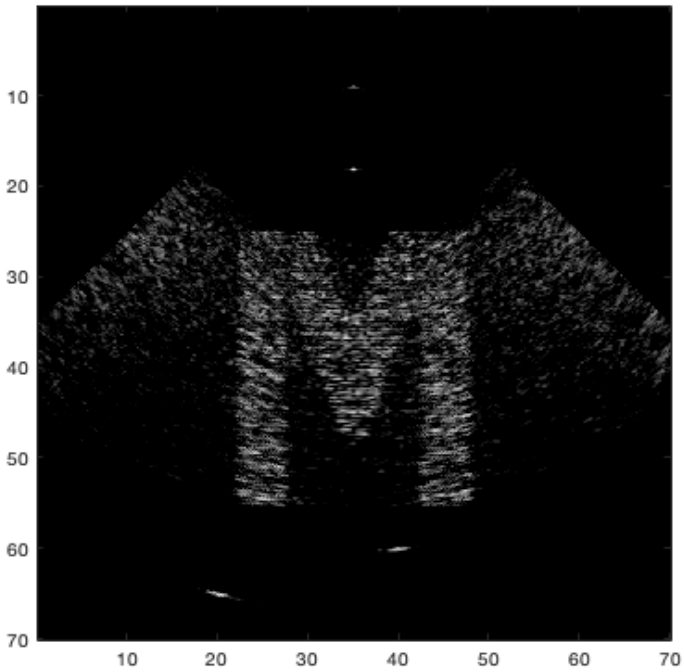
```
%       template_scancon.m
%       template script for converting from r-sin(theta) data to x-y image
%       must load r-sin(theta) data: rsdata

% --> QUESTION 1. <--
% Scan convert the r-sin(theta) buffer to produce a sector scan image.
% Use bilinear interpolation to compute the image values on the
% sector scan image grid.  Matlab's "interp2" function will help you
% do bilinear interpolation.

% compute values needed for interp2
x = linspace(-35,35,512);
z = linspace(0,70,512);
[XI,ZI] = meshgrid(x,z);
[theta_i,r_i]=meshgrid(sin_theta,r);
r_ii = sqrt(XI.^2+ZI.^2);
theta_ii = XI./r_ii;

% Create image w/ bilinear interpolation

im = interp2(theta_i, r_i, abs(rsdata), theta_ii, r_ii, 'bilinear');
tt = find(isnan(im));
im(tt) = zeros(size(tt));

im_i = interp2(theta_i, r_i, abs(rsdata_i), theta_ii, r_ii, 'bilinear');
tt_i = find(isnan(im_i));
im(tt_i) = zeros(size(tt_i));

im_j = interp2(theta_i, r_i, abs(rsdata_j), theta_ii, r_ii, 'bilinear');
tt_j = find(isnan(im_j));
im(tt_j) = zeros(size(tt_j));

im_k = interp2(theta_i, r_i, abs(rsdata_i), theta_ii, r_ii, 'bilinear');
tt_k = find(isnan(im_k));
im(tt_k) = zeros(size(tt_k));

% do similarly all r-sin(theta) buffers

% --> QUESTION 1. <--
% Use two images on a logarithmic scale to answer this question:
% one on a 40dB scale, the other on a 20dB scale
figure(10); showimage3(im, 1, 20,70/512,70/512); axis('image')          % Display 20 dB scale image
figure(11); showimage3(im, 1, 40,70/512,70/512); axis('image')          % Display 40 dB scale image

figure(12); showimage3(im_i, 1, 20,70/512,70/512); axis('image')                % Display 20 dB scale image
figure(13); showimage3(im_i, 1, 40,70/512,70/512); axis('image')                % Display 40 dB scale image

figure(14); showimage3(im_j, 1, 20,70/512,70/512); axis('image')                % Display 20 dB scale image
figure(15); showimage3(im_j, 1, 40,70/512,70/512); axis('image')                % Display 40 dB scale image

figure(16); showimage3(im_k, 1, 20,70/512,70/512); axis('image')                % Display 20 dB scale image
figure(17); showimage3(im_k, 1, 40,70/512,70/512); axis('image')                % Display 40 dB scale image
```

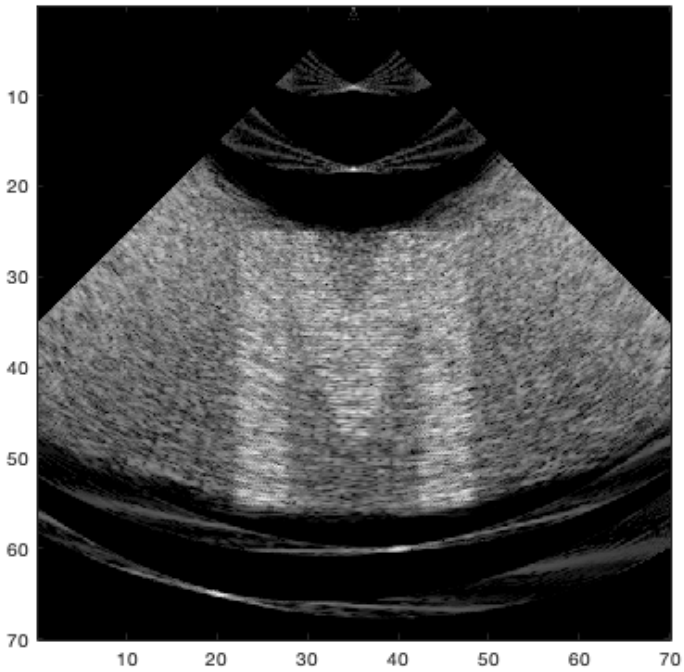

10
20
30
40
50
60
70
10
20
30
40
50
60
70

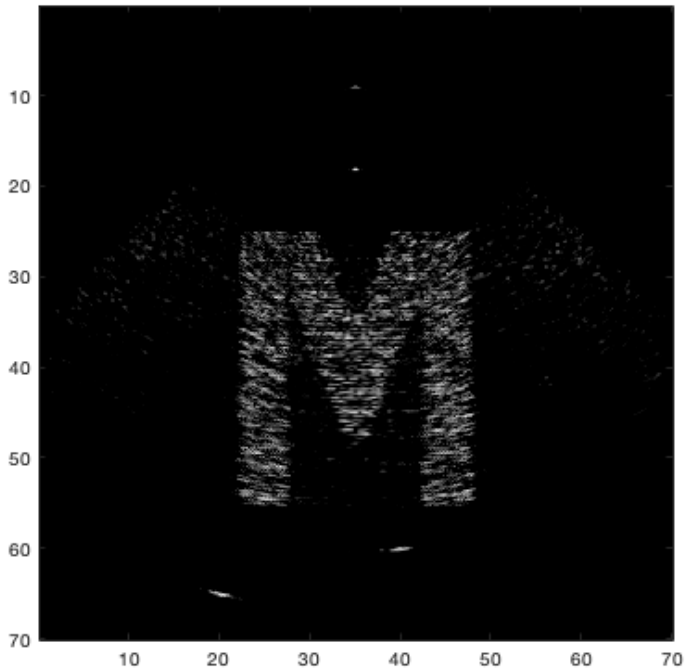

10
20
30
40
50
60
70
10
20
30
40
50
60
70

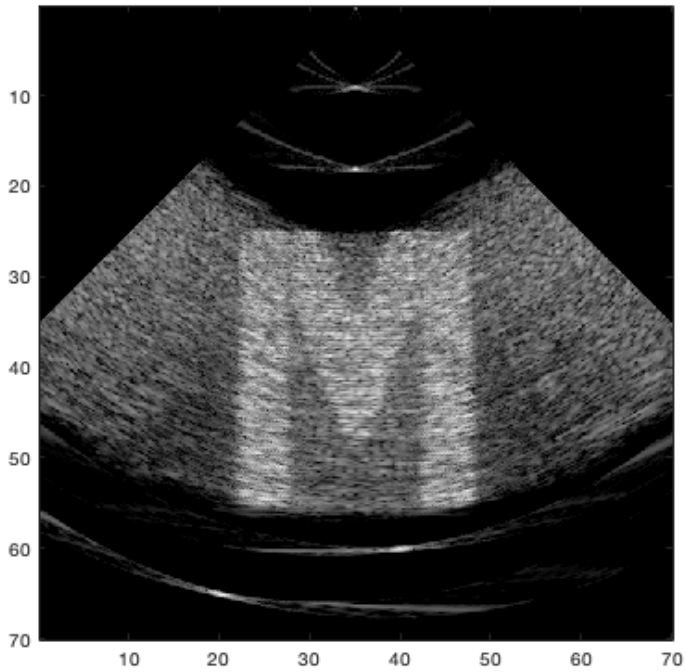

10
20
30
40
50
60
70
10
20
30
40
50
60
70


10
20
30
40
50
60
70
10
20
30
40
50
60
70


10
20
30
40
50
60
70
10
20
30
40
50
60
70


10
20
30
40
50
60
70
10
20
30
40
50
60
70





Published with MATLAB® R2022a