Table of Contents

Question 1

```
phantom_image = phantom(128);
```

Part (a)

myIntegration(image, t, theta, deltaS); As we can see that the value of delta s set to 0.5 is good enough as the integration value doesnt change much. But the time will increase linearly. For the most accurate and still reasonable performance a low value such as 0.01 or lower is favorable.

```
sampleT = 5;
sampleTheta = 75;
i1 = myIntegration(phantom_image, sampleT, sampleTheta, 2);
i2 = myIntegration(phantom_image, sampleT, sampleTheta, 1);
i3 = myIntegration(phantom_image, sampleT, sampleTheta, 0.5);
i4 = myIntegration(phantom_image, sampleT, sampleTheta, 0.1);
i5 = myIntegration(phantom_image, sampleT, sampleTheta, 0.05);
i6 = myIntegration(phantom_image, sampleT, sampleTheta, 0.01);
i7 = myIntegration(phantom_image, sampleT, sampleTheta, 0.001);
display([i1, i2, i3, i4, i5, i6, i7]);
```

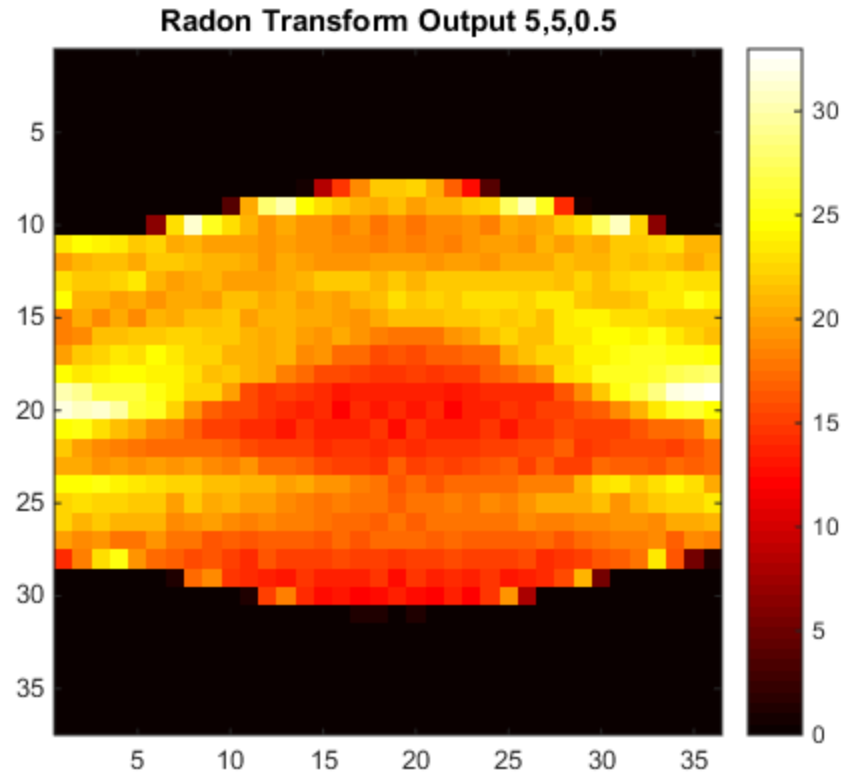
```
ans =

   13.0060   12.9055   12.9230   12.9274   12.9262   12.9266   12.9266
```

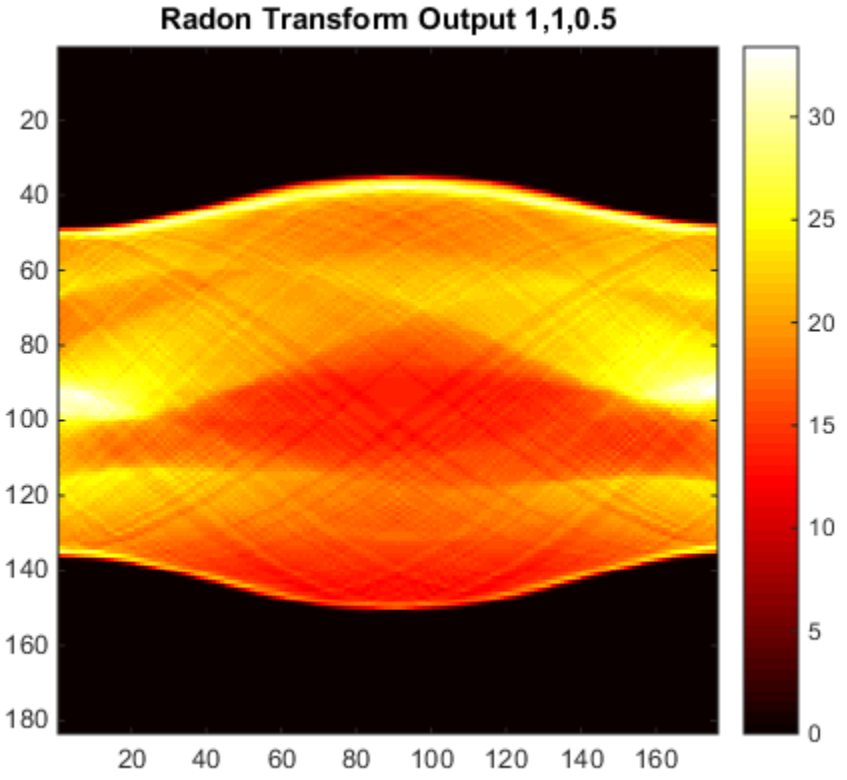
Part (b)

myRadonTransform(image, deltaT, deltaTheta, deltaS);

```
% deltaT = 5, deltaTheta = 5, deltaS = 0.5
R = myRadonTransform(phantom_image, 5, 5, 0.5);
showImage(R, 'Radon Transform Output 5,5,0.5');

% deltaT = 1, deltaTheta = 1, deltaS = 0.5
R = myRadonTransform(phantom_image, 1, 1, 0.5);
showImage(R, 'Radon Transform Output 1,1,0.5');
```

```
Starting parallel pool (parpool) using the 'local' profile ... connected to 4 work
```

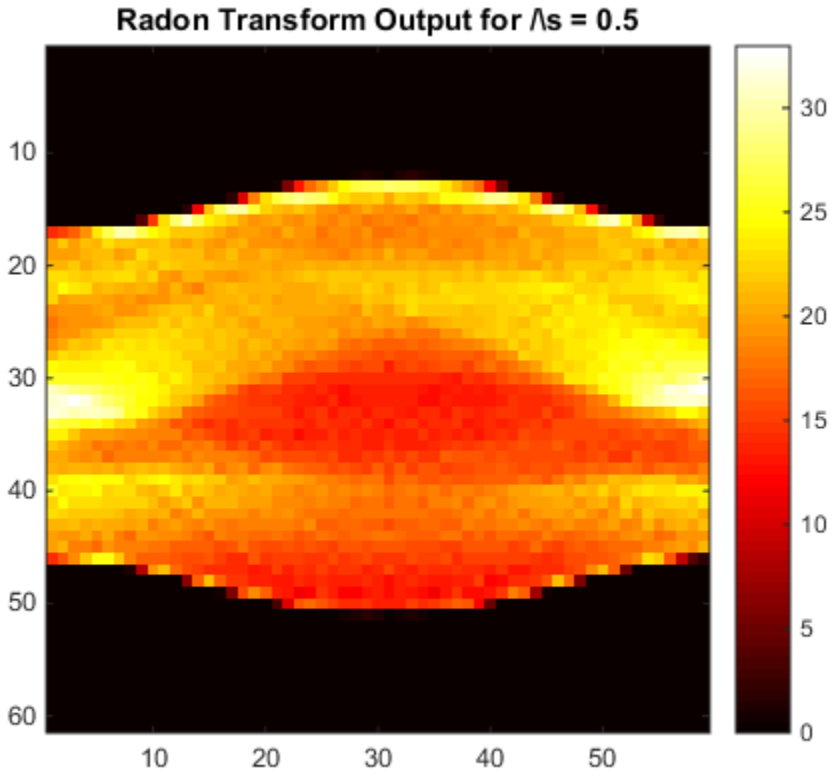

Radon Transform Output 5,5,0.5
5
10
15
20
25
30
35
5
10
15
20
25
30
35
30
25
20
15
10
5
0

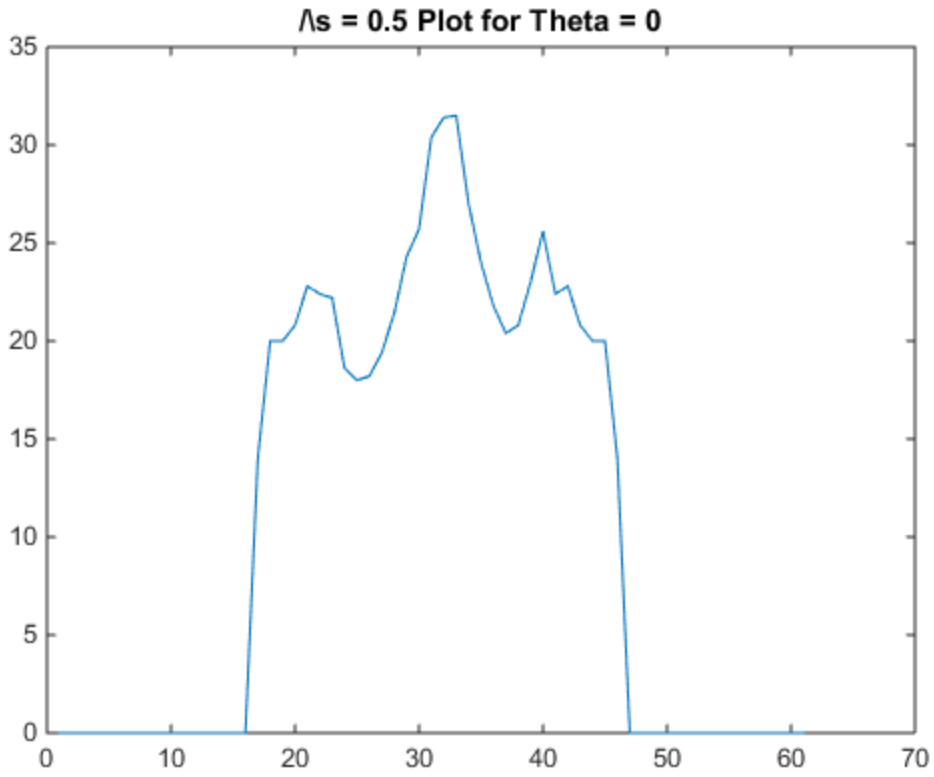

Radon Transform Output 1,1,0.5
20
40
60
80
100
120
140
160
180
20
40
60
80
100
120
140
160
30
25
20
15
10
5
0

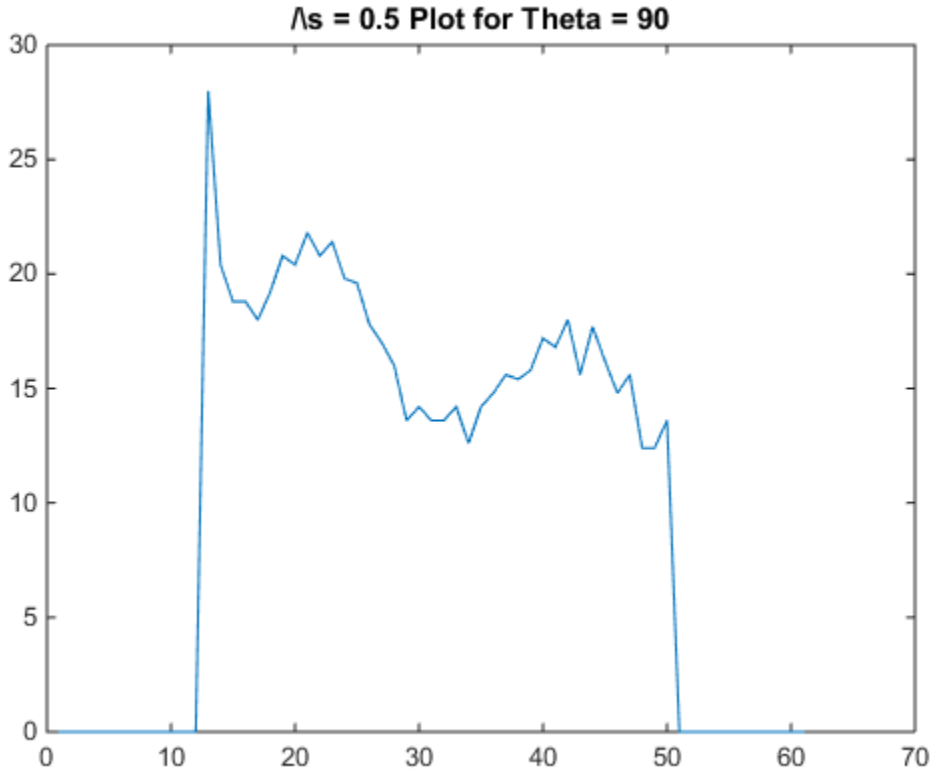
Part (c)

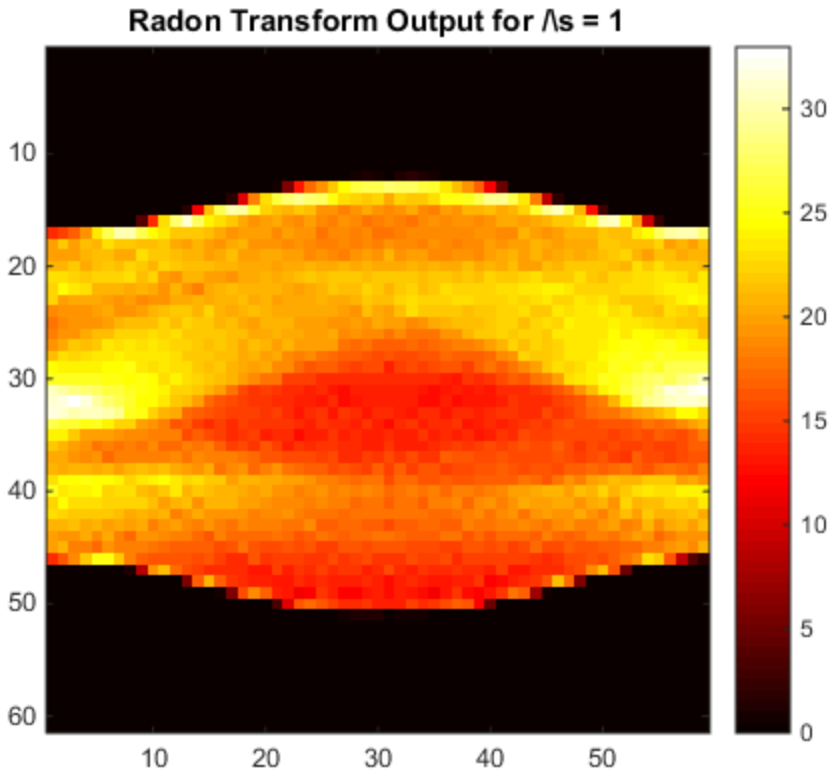
The 1D and Image appear the smoothest for /\s = 0.5. This is because the calculation of the integral is finer and more accurate. And the result is smoother as it is closer to the original value.

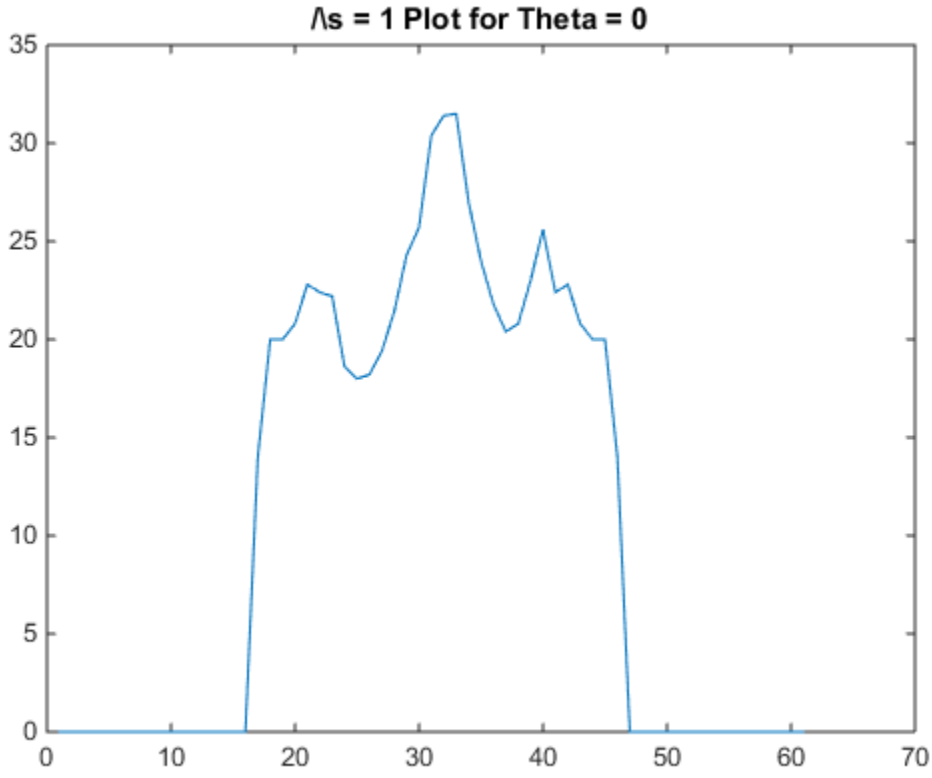
```
delta_t = 3;
delta_theta = 3;

R = myRadonTransform(phantom_image, delta_t, delta_theta, 0.5);
showImage(R, 'Radon Transform Output for /\\s = 0.5');
plotForPartC(R, 0, '/\\s = 0.5 Plot for Theta = 0', delta_theta);
plotForPartC(R, 90, '/\\s = 0.5 Plot for Theta = 90', delta_theta);

R = myRadonTransform(phantom_image, delta_t, delta_theta, 1);
showImage(R, 'Radon Transform Output for /\\s = 1');
plotForPartC(R, 0, '/\\s = 1 Plot for Theta = 0', delta_theta);
plotForPartC(R, 90, '/\\s = 1 Plot for Theta = 90', delta_theta);

R = myRadonTransform(phantom_image, delta_t, delta_theta, 3);
showImage(R, 'Radon Transform Output for /\\s = 3');
plotForPartC(R, 0, '/\\s = 3 Plot for Theta = 0', delta_theta);
plotForPartC(R, 90, '/\\s = 3 Plot for Theta = 90', delta_theta);
```

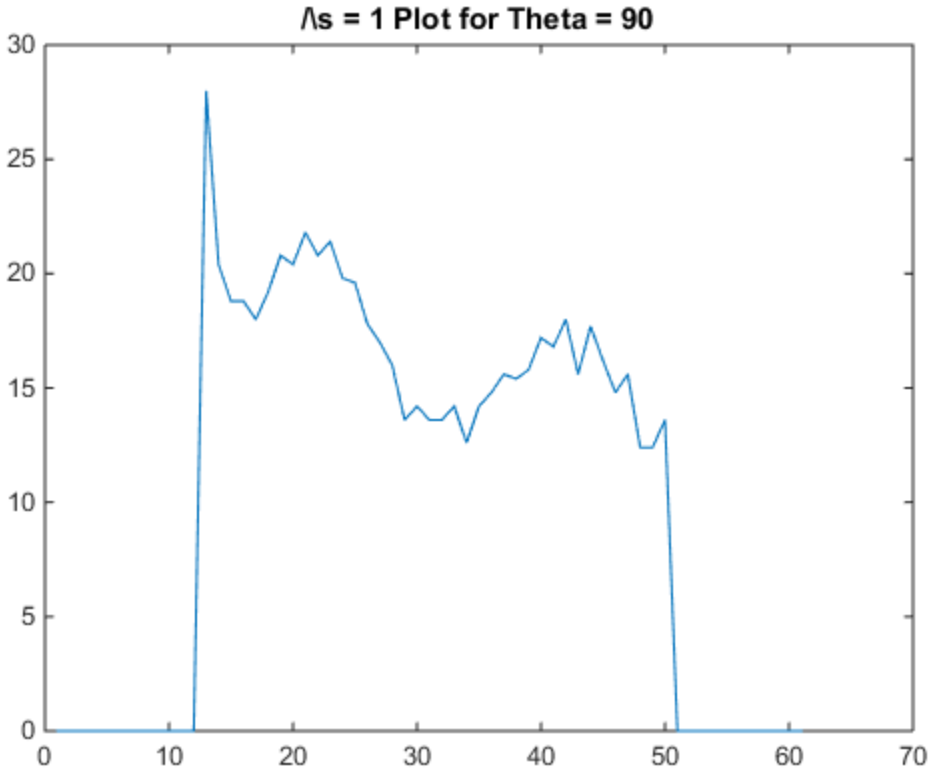


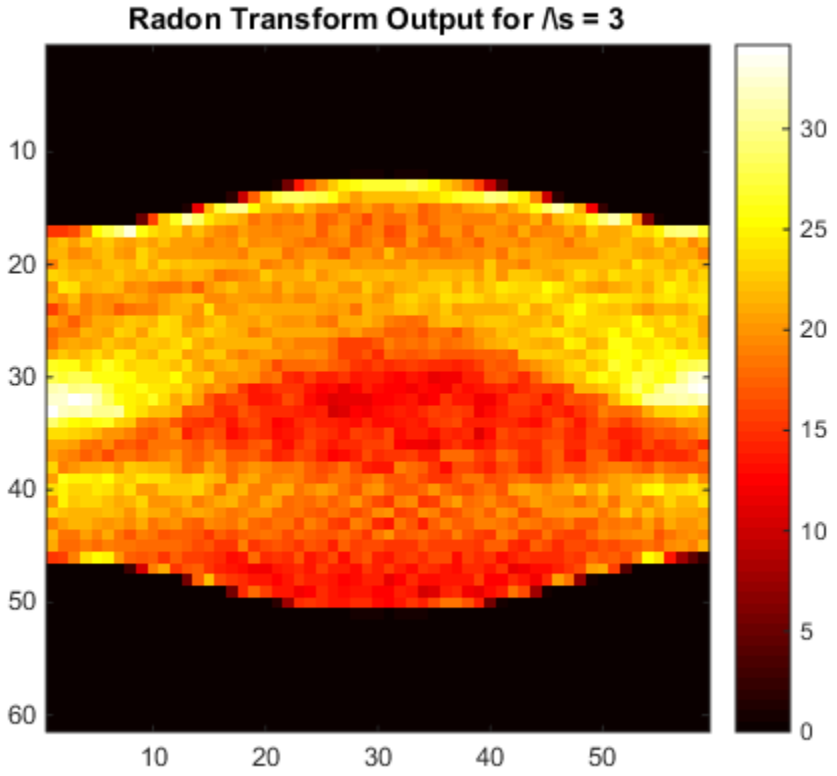

Λs = 0.5 Plot for Theta = 0
35
30
25
20
15
10
5
0
0
10
20
30
40
50
60
70

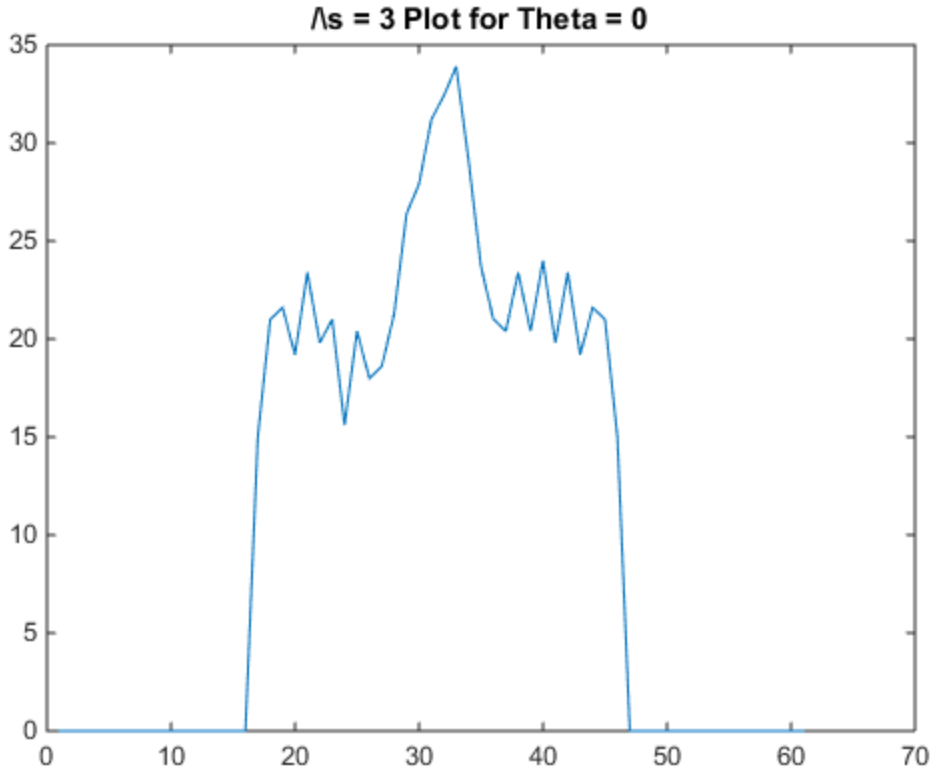

Λs = 0.5 Plot for Theta = 90
30
25
20
15
10
5
0
0
10
20
30
40
50
60
70

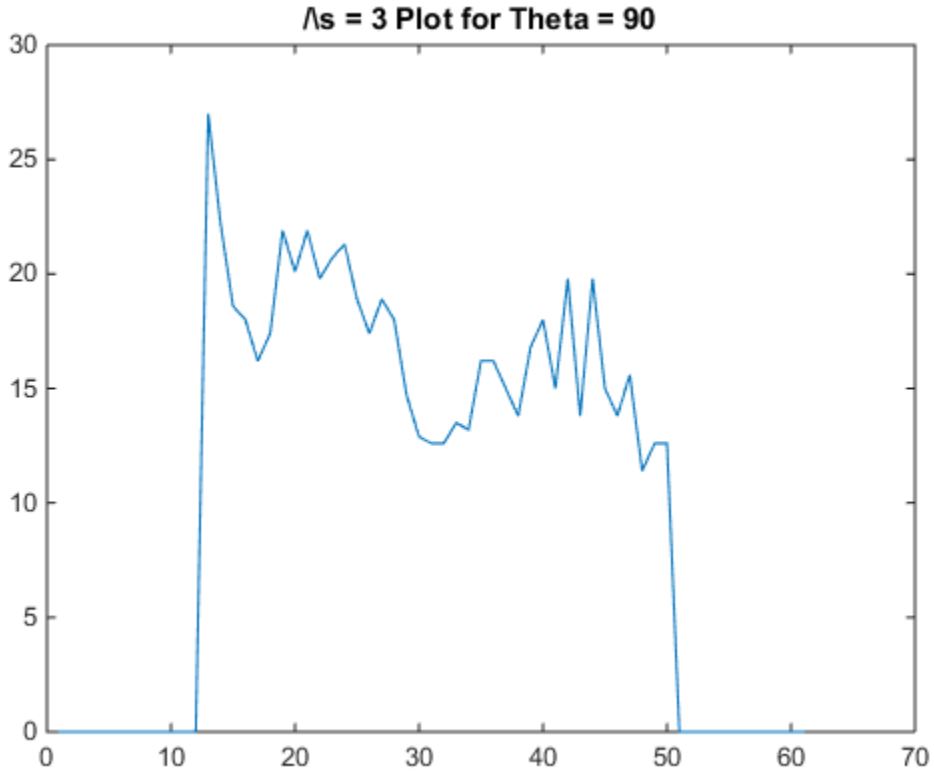

Radon Transform Output for /\s = 1
10
20
30
40
50
60
10
20
30
40
50
30
25
20
15
10
5
0


/\s = 1 Plot for Theta = 0
35
30
25
20
15
10
5
0
0
10
20
30
40
50
60
70


Δs = 1 Plot for Theta = 90
30
25
20
15
10
5
0
0
10
20
30
40
50
60
70


Radon Transform Output for Δs = 3
10
20
30
40
50
60
10
20
30
40
50
30
25
20
15
10
5
0


Λs = 3 Plot for Theta = 0
35
30
25
20
15
10
5
0
0
10
20
30
40
50
60
70


Λs = 3 Plot for Theta = 90
30
25
20
15
10
5
0
0
10
20
30
40
50
60
70

Part (d)

Value of /\t and /\theta should be small. As large values make the image very boxy and unclear. However too small values will make the code very slow. As we saw in the first part a value of 1 is enough to resolve the values as close to one pixel apart. Also, we saw that it resembles very close the true radon transform. Hence a /\t = 1 is good enough and /\theta = 1 is also needed to resolve all pixel far away.

Part (e)

For this case a pixel grid which has enough resolution to detect objects of importance and their boundaries should be chosen. For instance, most internal arteries need to be well resolved and so do some alveolar ducts. Hence a pixel size corresponding to 0.1m or so will work. Lower can be done to improve resolution.

A delta s of 0.5pixel width will be able to resolve most of the corner cases well as we have previously seen. Ofcourse doing 0.01 or so will improve reconstruction upto a limit.

For /\s 1, the benefit of reducing will fade after a point. (Seen in part(a)). Hence after a point, no change will be visible, the computation time will increase with no benefit. Some floating point artifact might even corrupt the result For /\s > 1, the image will be very blurry and not accurate. It will be unable to clearly resolve the various parts of the body.

Published with MATLAB® R2014b

Question 2

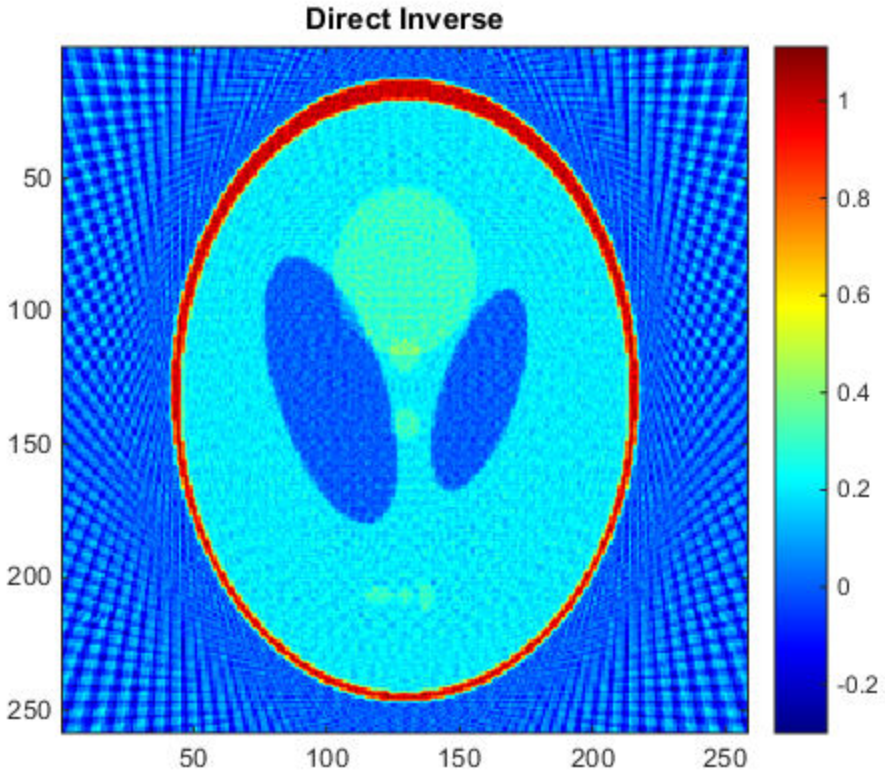
Table of Contents

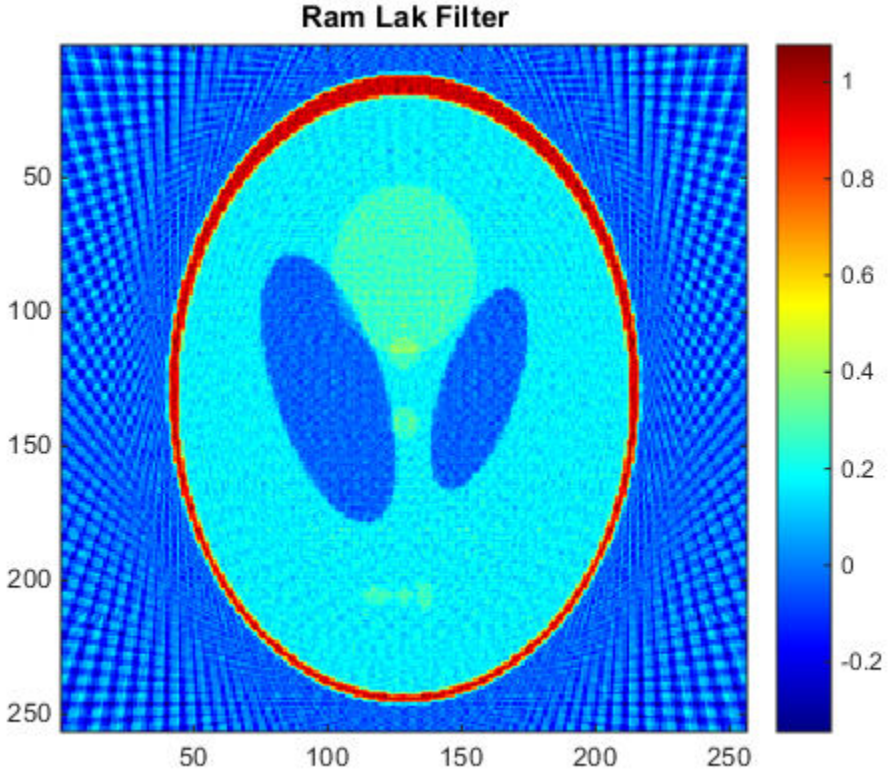
Part (a)

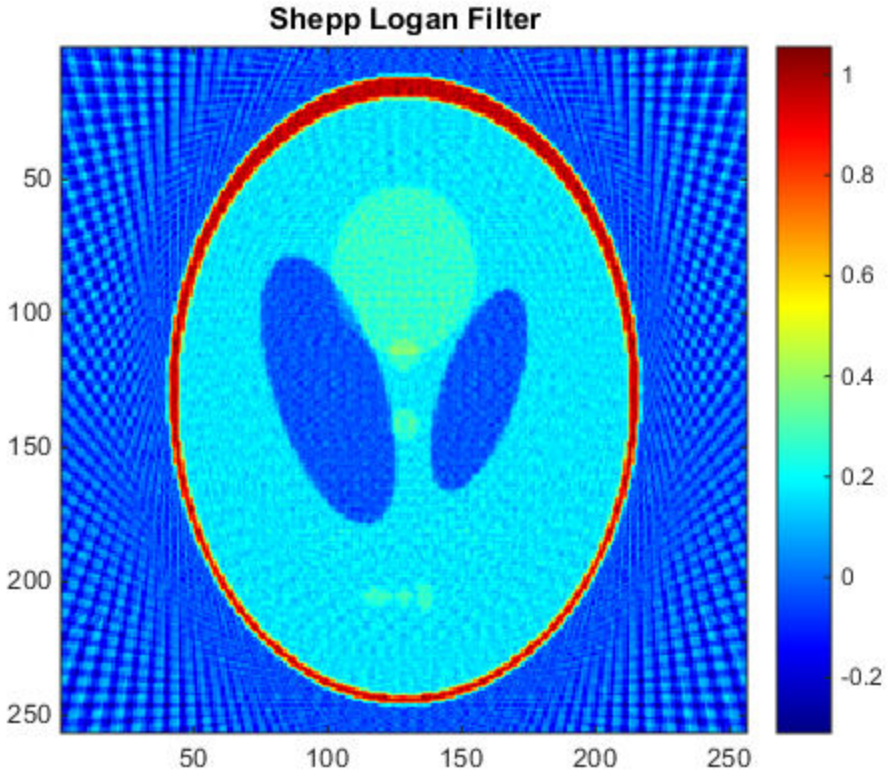
```
image_size = 256;
phantom_image = phantom(image_size);
thetas = 0:3:177;

phantom_radon = radon(phantom_image, thetas);
phantom_fft = fft(phantom_radon, [], 1);
phantom_fft = fftshift(phantom_fft, 1);

w_max = floor(size(phantom_fft, 1) / 2);
L1 = w_max;
L2 = w_max / 2;
```

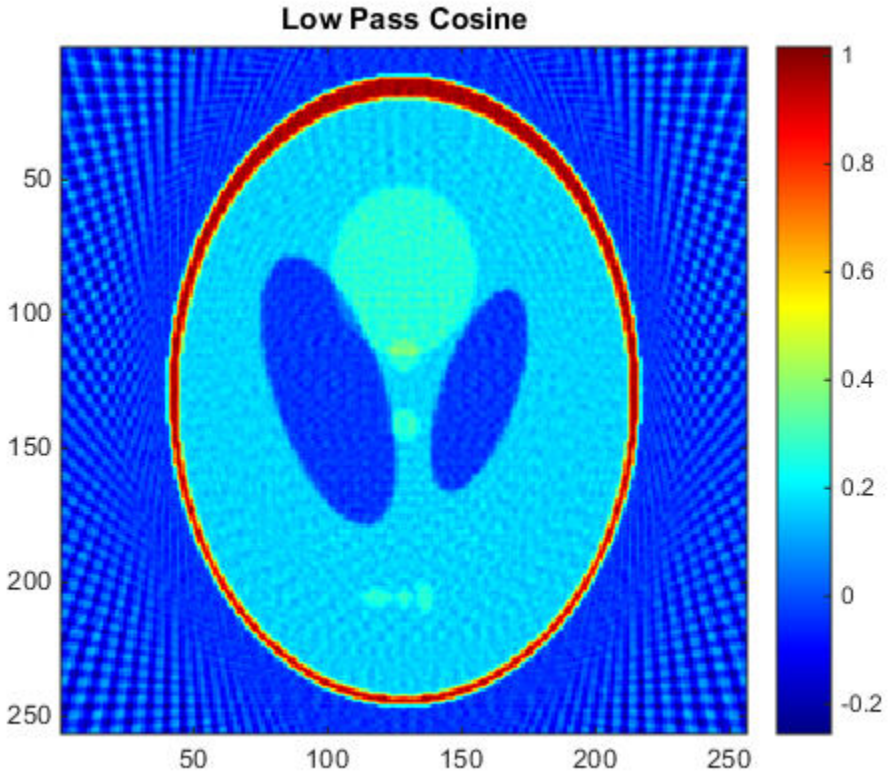
L1 = w_max

```
% Direct Inverse
direct_inverse = iradon(phantom_radon, thetas);
showImage(direct_inverse, 'Direct Inverse');

% Ram Lak Filter
phantom_ram_lak_fft = applyRamLak(phantom_fft, L1);
ram_lak_inverse = getImageFromFiltered(phantom_ram_lak_fft, ...
    thetas, image_size);
showImage(ram_lak_inverse, 'Ram Lak Filter');

% Shepp Logan Filter
phantom_shepp_logan_fft = applySheppLogan(phantom_fft, L1);
shepp_logan_inverse = getImageFromFiltered(phantom_shepp_logan_fft, ...
    thetas, image_size);
showImage(shepp_logan_inverse, 'Shepp Logan Filter');

% Low Pass Cosine Filter
phantom_low_pass_cosine_fft = applyLowPassCosine(phantom_fft, L1);
low_pass_cosine_inverse = getImageFromFiltered( ...
    phantom_low_pass_cosine_fft, thetas, image_size);
showImage(low_pass_cosine_inverse, 'Low Pass Cosine');
```


Direct Inverse
50
100
150
200
250
50
100
150
200
250
1
0.8
0.6
0.4
0.2
0
-0.2

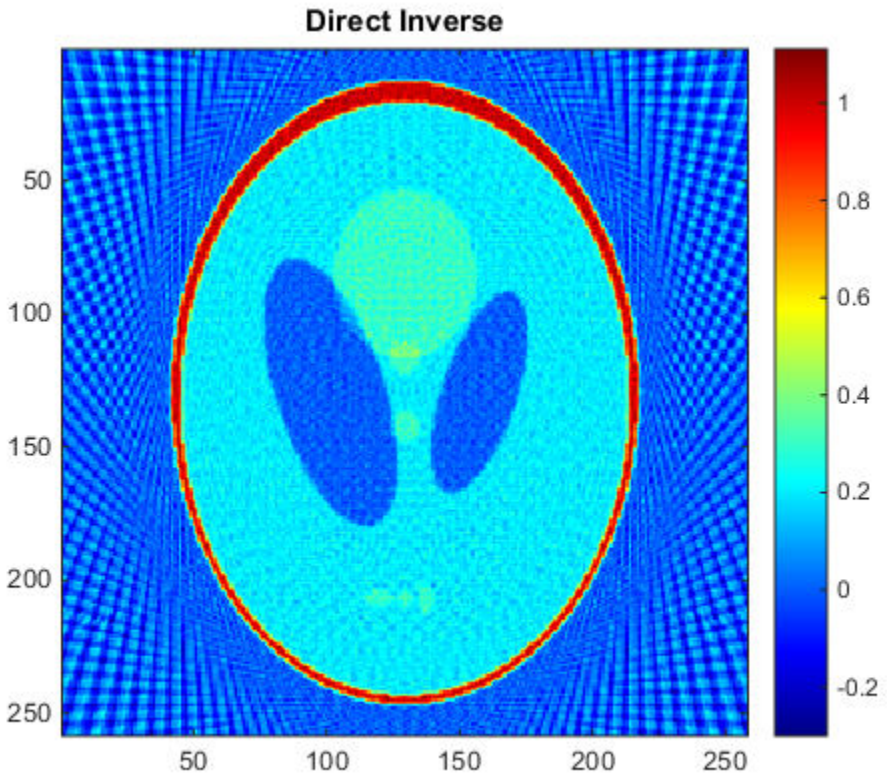

Ram Lak Filter
50
100
150
200
250
50
100
150
200
250
1
0.8
0.6
0.4
0.2
0
-0.2

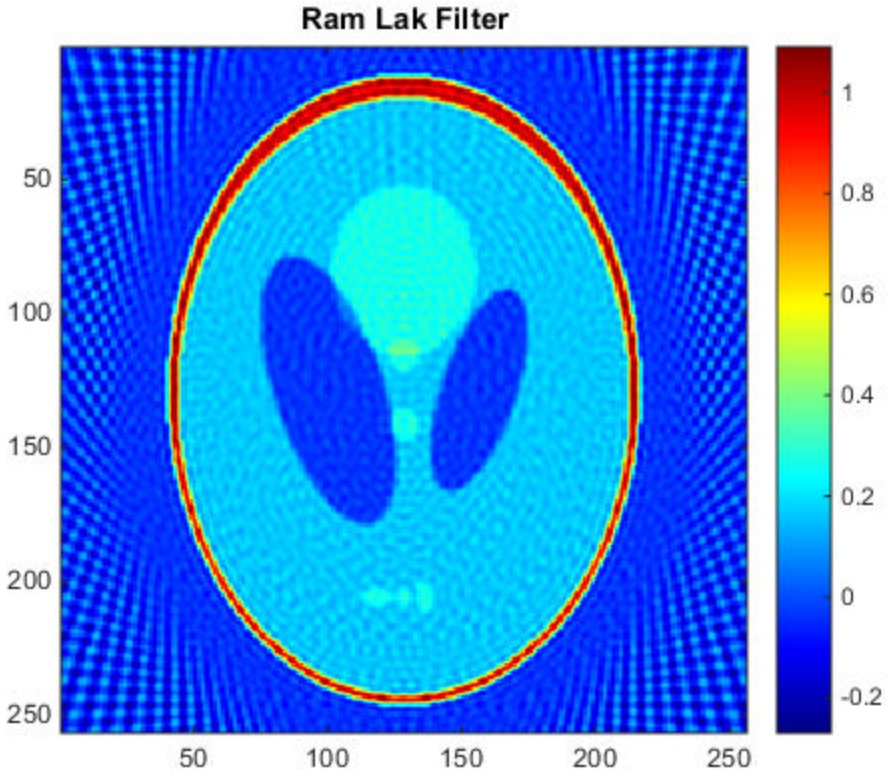

Shepp Logan Filter
50
100
150
200
250
50
100
150
200
250
1
0.8
0.6
0.4
0.2
0
-0.2

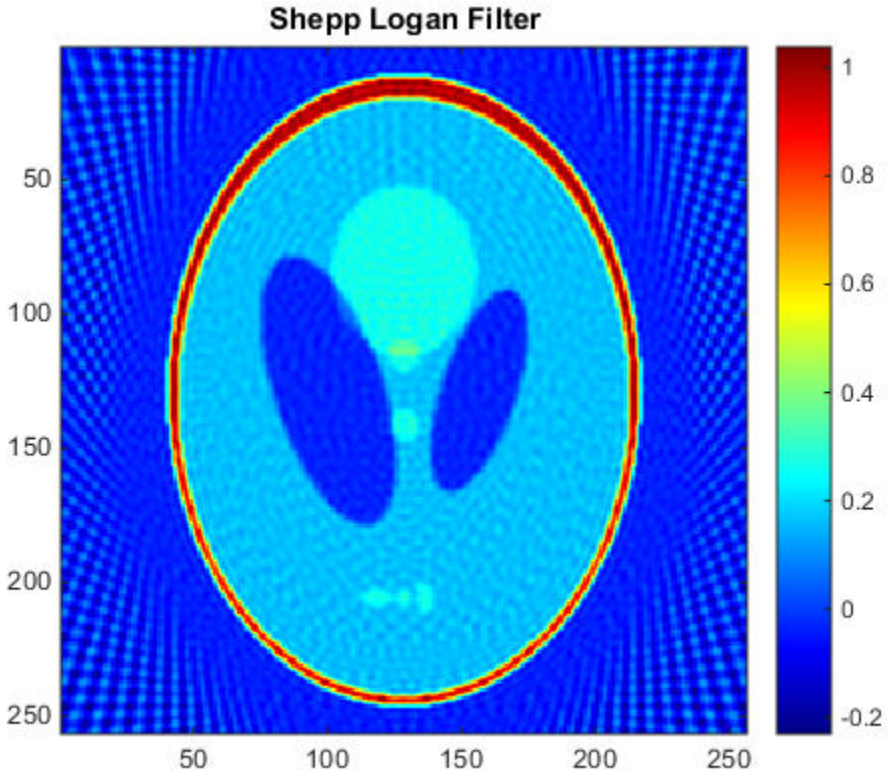

Low Pass Cosine
50
100
150
200
250
50
100
150
200
250
1
0.8
0.6
0.4
0.2
0
-0.2

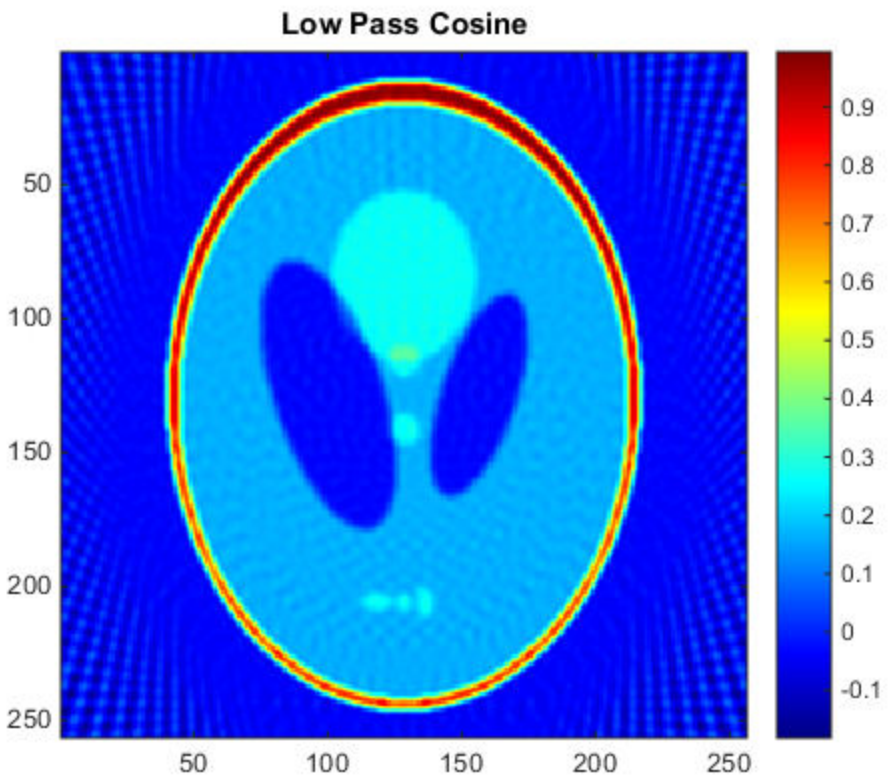
L2 = w_max / 2

```
% Direct Inverse
direct_inverse = iradon(phantom_radon, thetas);
showImage(direct_inverse, 'Direct Inverse');

% Ram Lak Filter
phantom_ram_lak_fft = applyRamLak(phantom_fft, L2);
ram_lak_inverse = getImageFromFiltered(phantom_ram_lak_fft, ...
    thetas, image_size);
showImage(ram_lak_inverse, 'Ram Lak Filter');

% Shepp Logan Filter
phantom_shepp_logan_fft = applySheppLogan(phantom_fft, L2);
shepp_logan_inverse = getImageFromFiltered(phantom_shepp_logan_fft, ...
    thetas, image_size);
showImage(shepp_logan_inverse, 'Shepp Logan Filter');

% Low Pass Cosine Filter
phantom_low_pass_cosine_fft = applyLowPassCosine(phantom_fft, L2);
low_pass_cosine_inverse = getImageFromFiltered( ...
    phantom_low_pass_cosine_fft, thetas, image_size);
showImage(low_pass_cosine_inverse, 'Low Pass Cosine');
```




Ram Lak Filter
50
100
150
200
250
50
100
150
200
250
1
0.8
0.6
0.4
0.2
0
-0.2


Shepp Logan Filter
50
100
150
200
250
50
100
150
200
250
1
0.8
0.6
0.4
0.2
0
-0.2



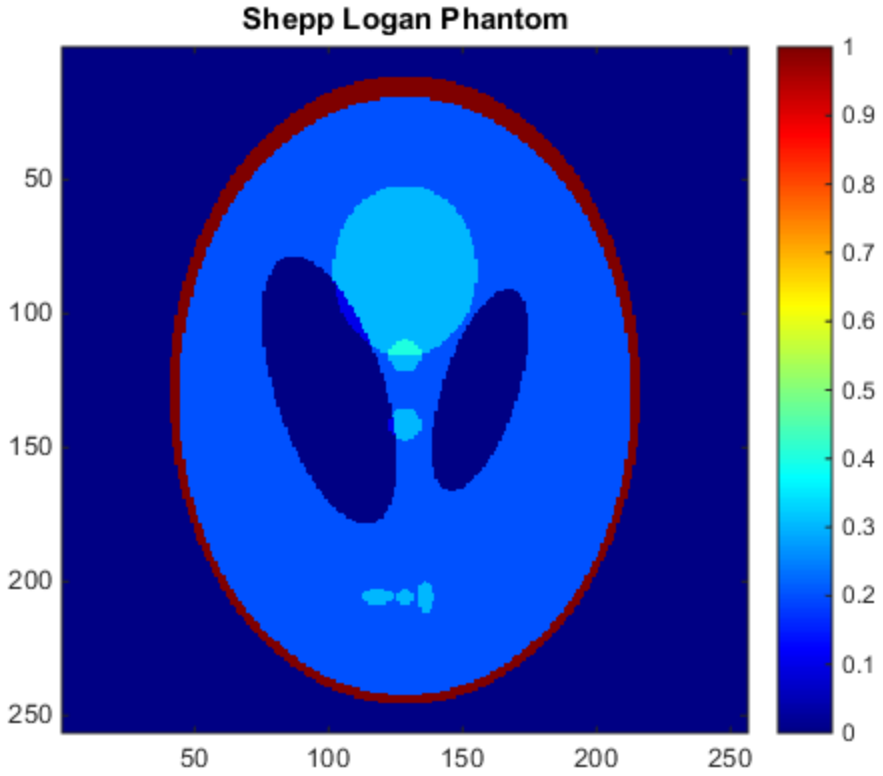
Part (b)

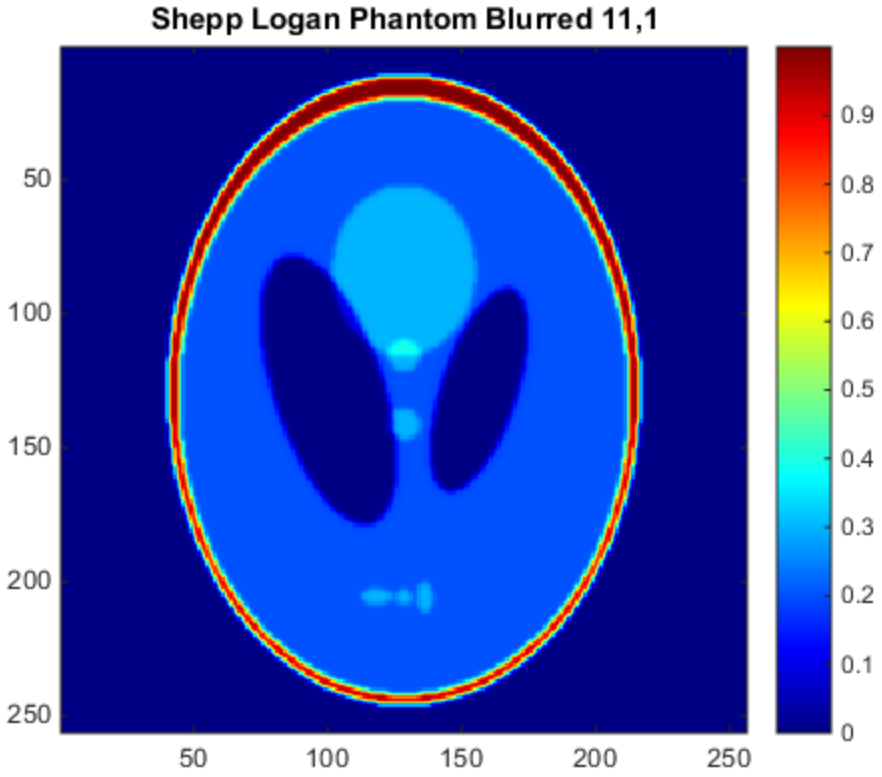
The RRMSE for the most blurred is the least because the image is blurred the noise is minimised. However the reconstruction is not substantially better as the RamLak Filter create sharpening which the blurry image is not wanting.

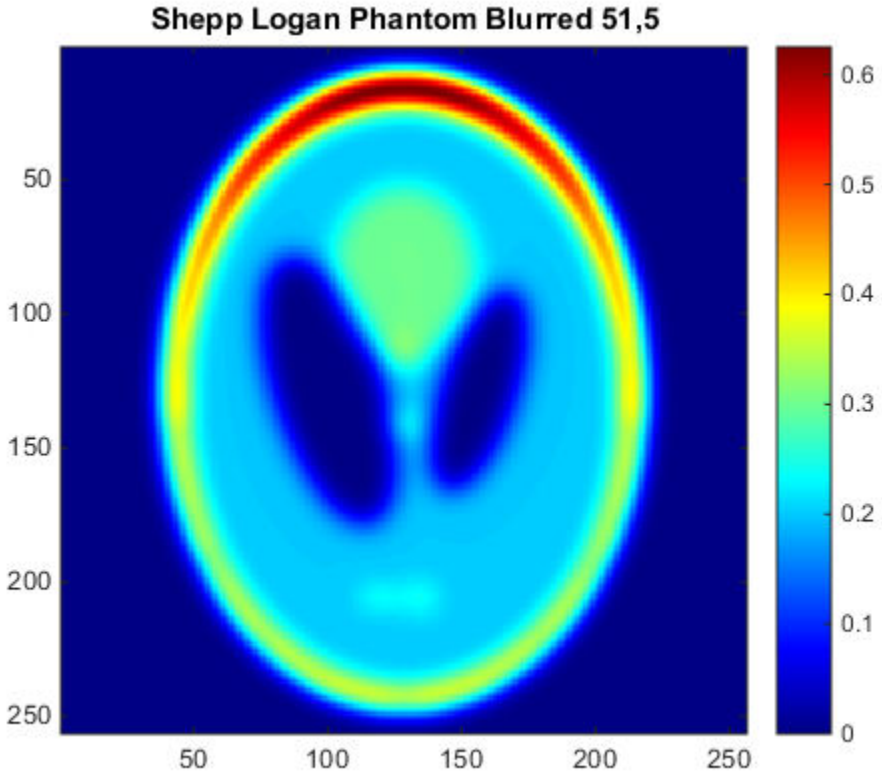
```
phantom_image = phantom(image_size);
mask = fspecial ('gaussian', 11, 1);
blurred_image_a = conv2 (phantom_image, mask, 'same');

mask = fspecial ('gaussian', 51, 5);
blurred_image_b = conv2 (phantom_image, mask, 'same');

showImage(phantom_image, 'Shepp Logan Phantom');
showImage(blurred_image_a, 'Shepp Logan Phantom Blurred 11,1');
showImage(blurred_image_b, 'Shepp Logan Phantom Blurred 51,5');

thetas = 0:3:177;

rrmse_phantom = rrmseForPartB(phantom_image, thetas, ...
    'Shepp Logan Phantom Reconstructed', image_size);
rrmse_phantom_blurred_a = rrmseForPartB(blurred_image_a, ...
    thetas, 'Shepp Logan Phantom Blurred 11,1 Reconstructed', image_size);
rrmse_phantom_blurred_b = rrmseForPartB(blurred_image_b, ...
    thetas, 'Shepp Logan Phantom Blurred 51,5 Reconstructed', image_size);
```

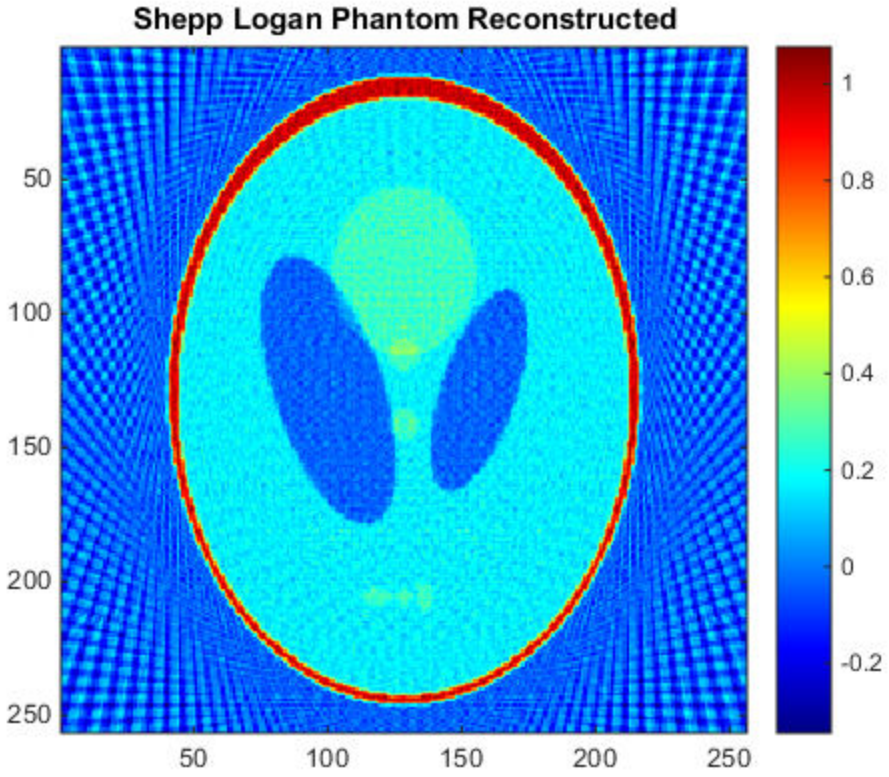
```
display(sprintf('RRMSE for default phantom %f', ...
    rrmse_phantom));
display(sprintf('RRMSE for blurred (11,1) phantom %f', ...
    rrmse_phantom_blurred_a));
display(sprintf('RRMSE for even blurred (51,5) phantom %f', ...
    rrmse_phantom_blurred_b));
```

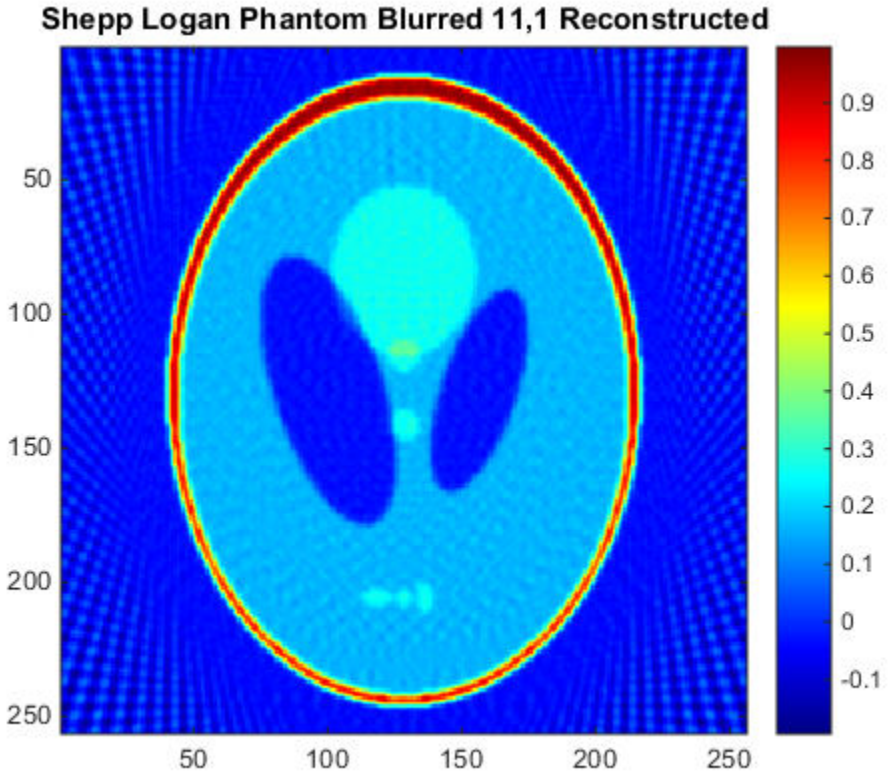
```
RRMSE for default phantom 0.326902
RRMSE for blurred (11,1) phantom 0.206541
RRMSE for even blurred (51,5) phantom 0.203948
```

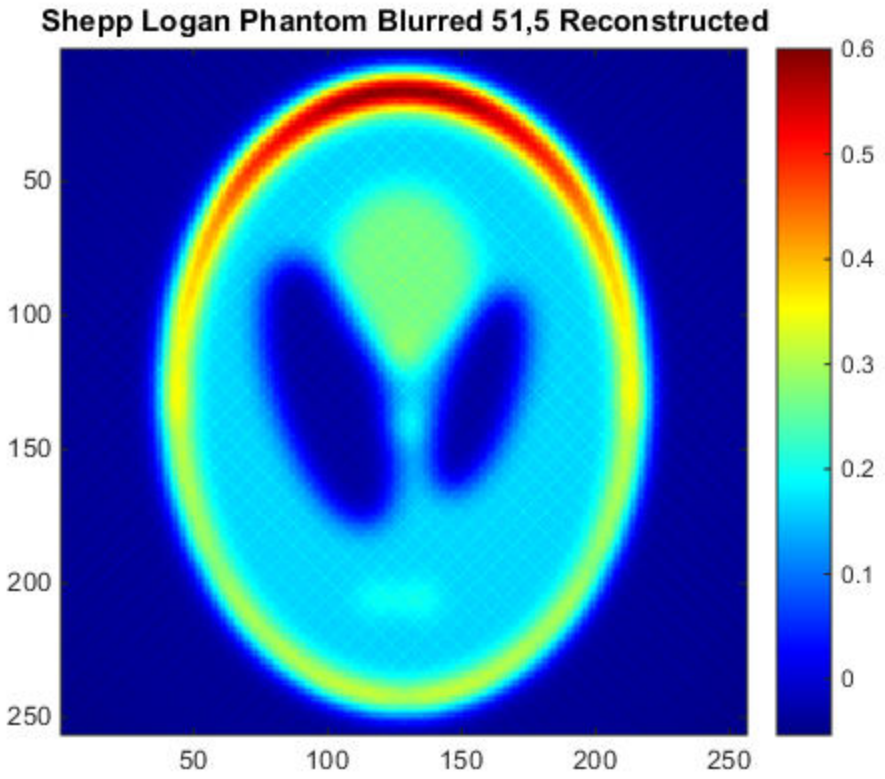



Shepp Logan Phantom Blurred 11,1
50
100
150
200
250
50
100
150
200
250
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Shepp Logan Phantom Blurred 51,5
50
100
150
200
250
50
100
150
200
250
0.6
0.5
0.4
0.3
0.2
0.1
0


Shepp Logan Phantom Reconstructed
50
100
150
200
250
50
100
150
200
250
1
0.8
0.6
0.4
0.2
0
-0.2

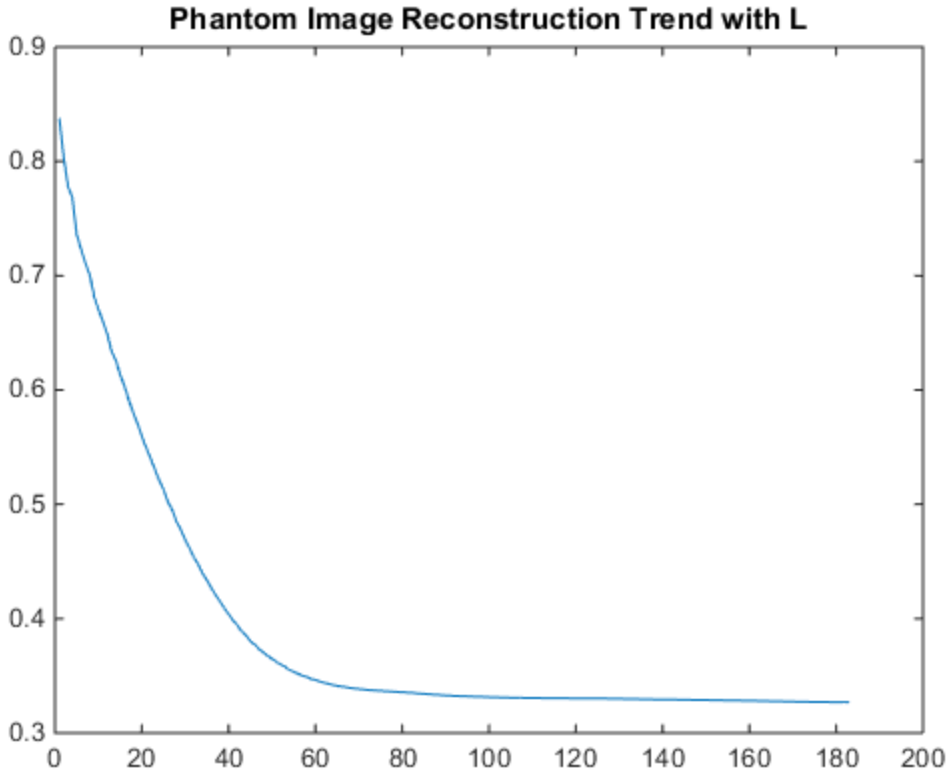

Shepp Logan Phantom Blurred 11,1 Reconstructed
50
100
150
200
250
50
100
150
200
250
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1



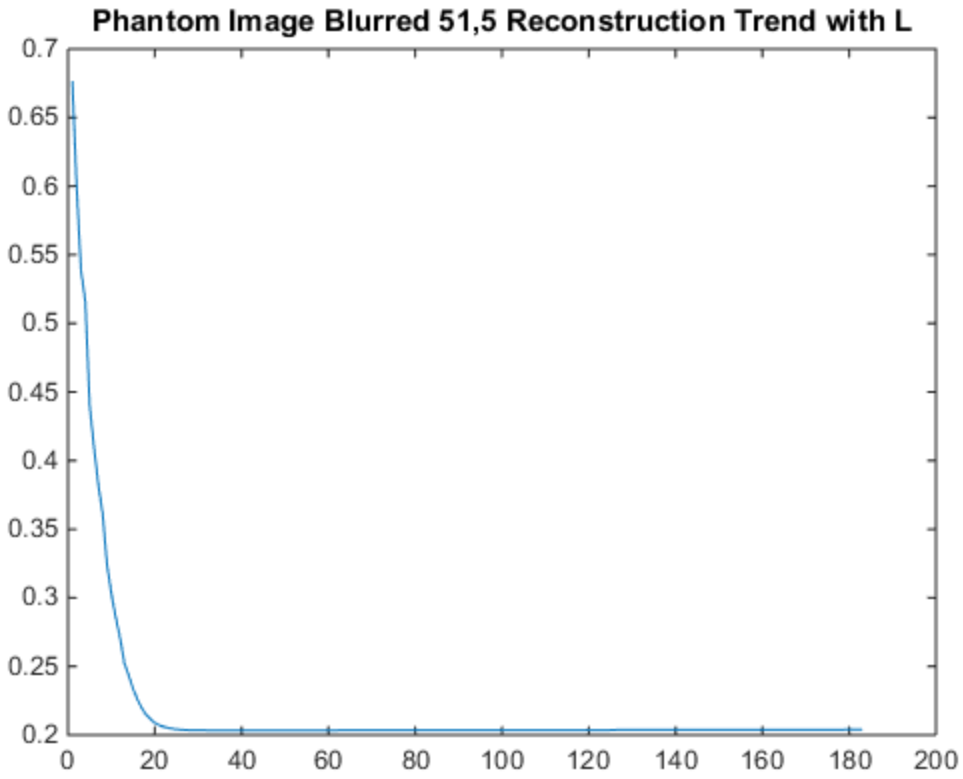
Part (c)

The trend for all the three is such that the RMSE reduces as L increases till it becomes steady. This is because initially we are removing important frequency components from the source. As the image is mostly noise free, the increase after L increases is not visible.

```
phantom_image = phantom(image_size);

mask = fspecial ('gaussian', 11, 1);
blurred_image_a = conv2 (phantom_image, mask, 'same');

mask = fspecial ('gaussian', 51, 5);
blurred_image_b = conv2 (phantom_image, mask, 'same');

thetas = 0:3:177;

plotForPartC(phantom_image, thetas, ...
    'Phantom Image Reconstruction Trend with L', image_size);
plotForPartC(blurred_image_a, thetas, ...
    'Phantom Image Blurred 11,1 Reconstruction Trend with L', image_size);
plotForPartC(blurred_image_b, thetas, ...
    'Phantom Image Blurred 51,5 Reconstruction Trend with L', image_size);
```


Phantom Image Reconstruction Trend with L
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
120
140
160
180
200

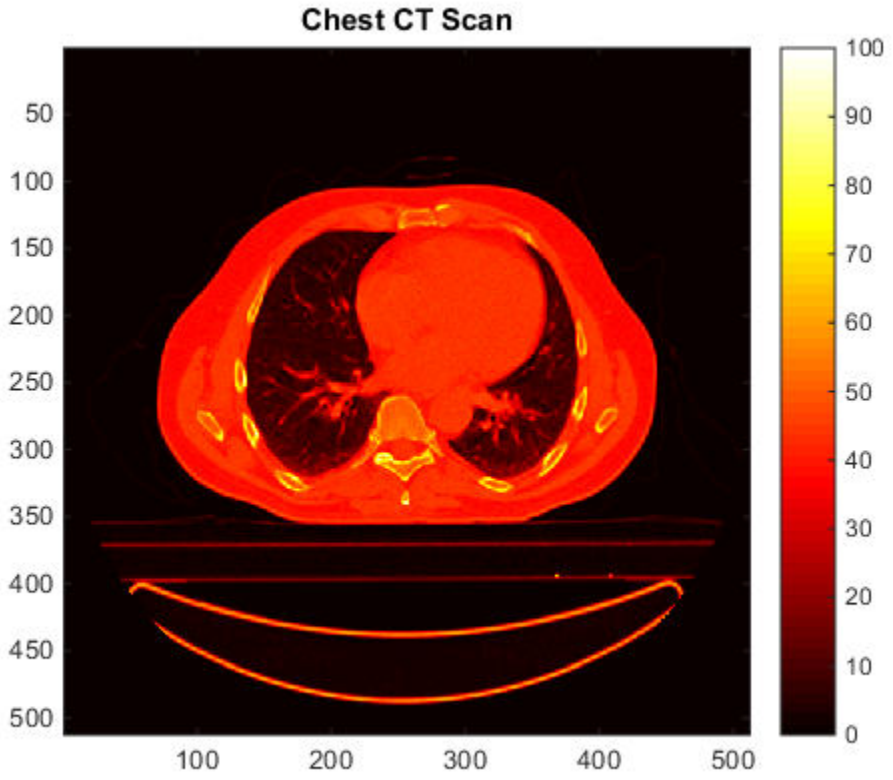

Phantom Image Blurred 11,1 Reconstruction Trend with L
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
120
140
160
180
200

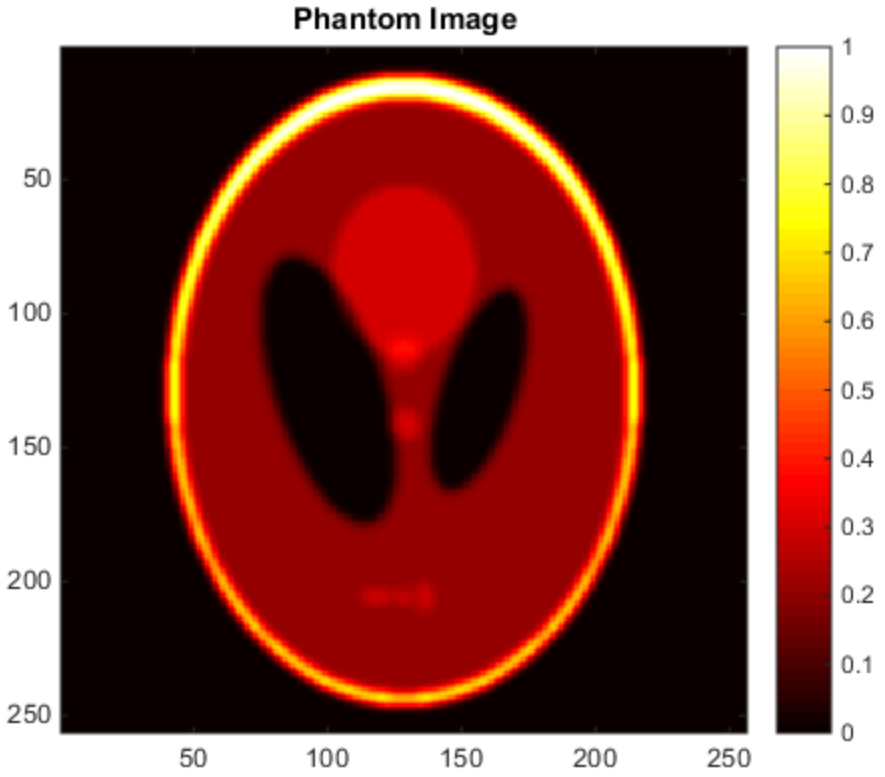


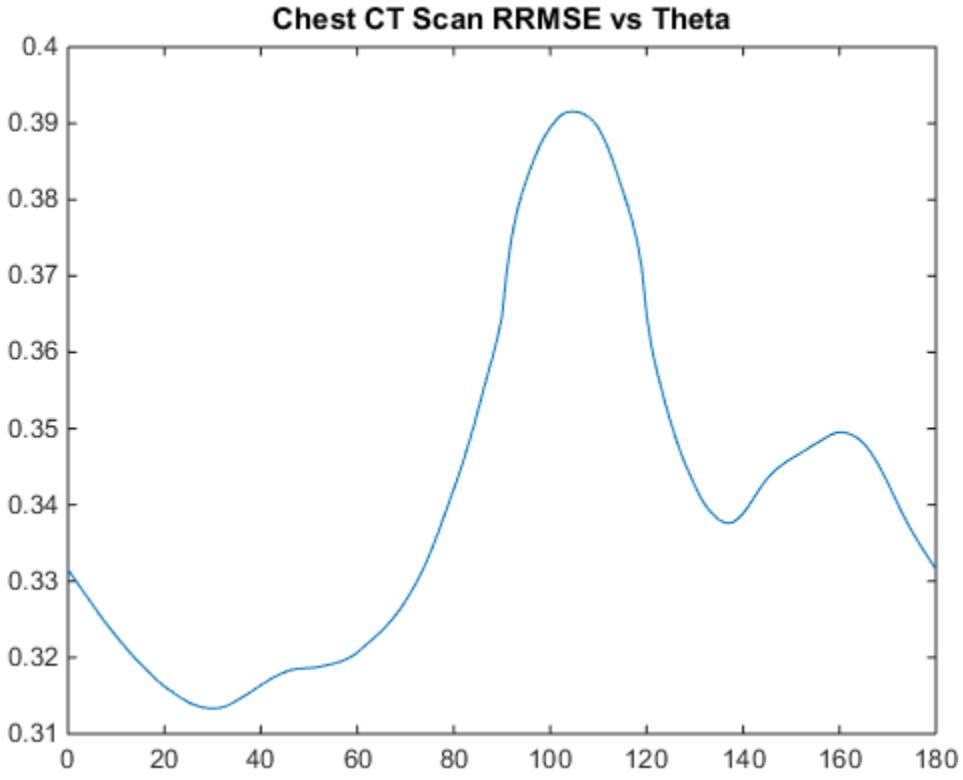
Published with MATLAB® R2014b

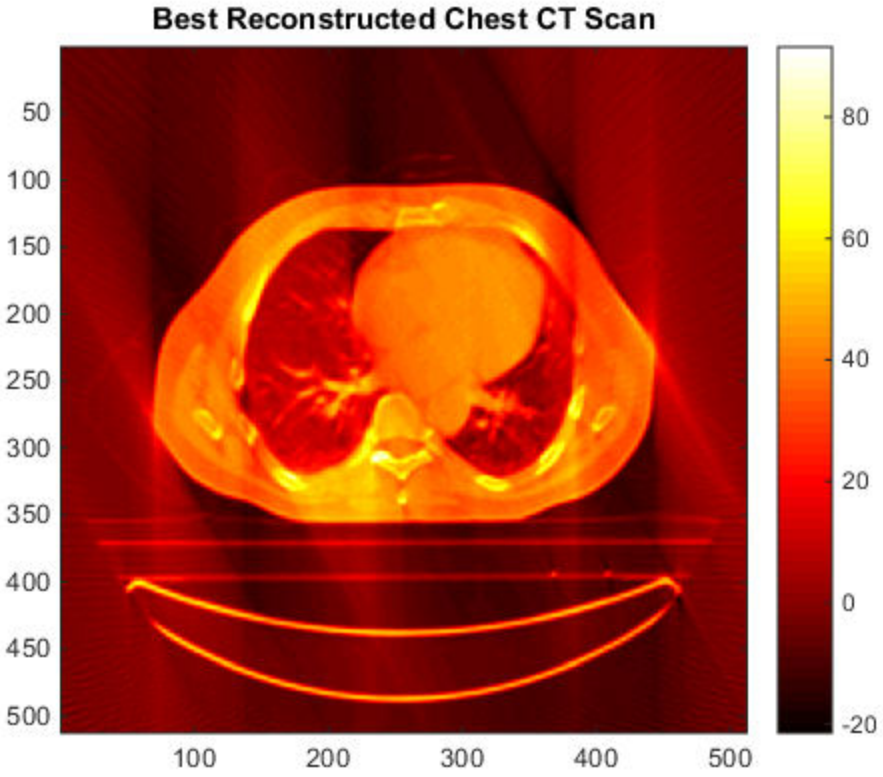
Question 3

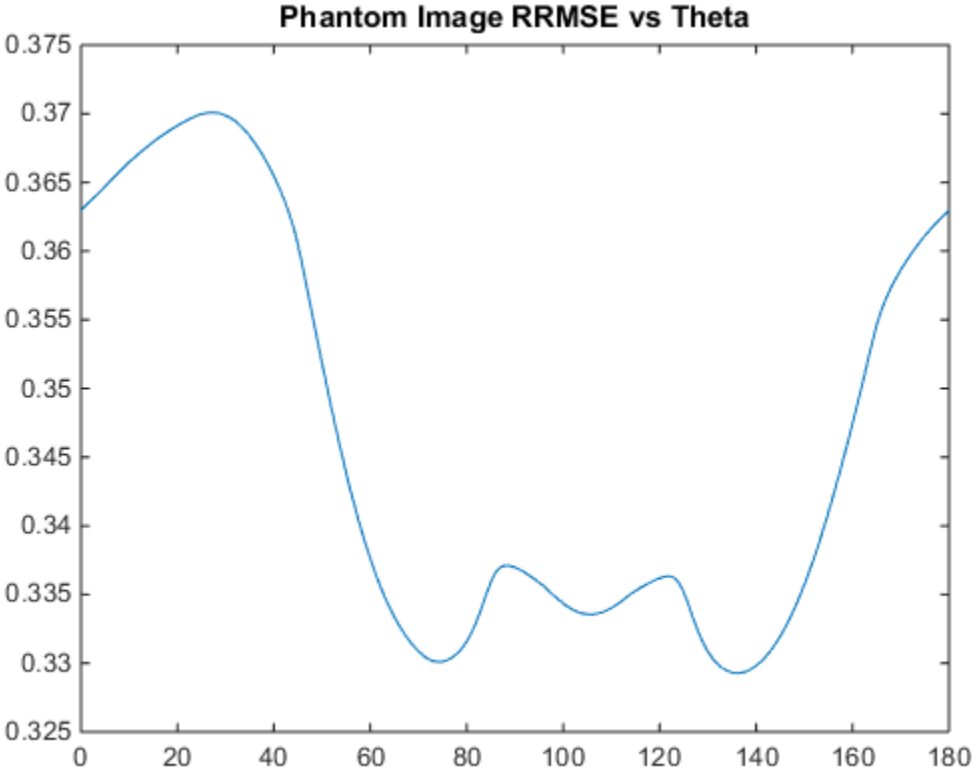
```
addpath '../../2/code';

% Loading the variables
load('../data/CT_Chest.mat');
chest_phantom = imageAC;

load('../data/myPhantom.mat');
my_phantom = imageAC;

showImage(chest_phantom, 'Chest CT Scan');
showImage(my_phantom, 'Phantom Image');

generateRRMSEPlot(chest_phantom, 'Chest CT Scan RRMSE vs Theta', ...
    'Best Reconstructed Chest CT Scan', size(chest_phantom, 1));
generateRRMSEPlot(my_phantom, 'Phantom Image RRMSE vs Theta', ...
    'Best Reconstructed Phantom Image', size(my_phantom, 1));
```

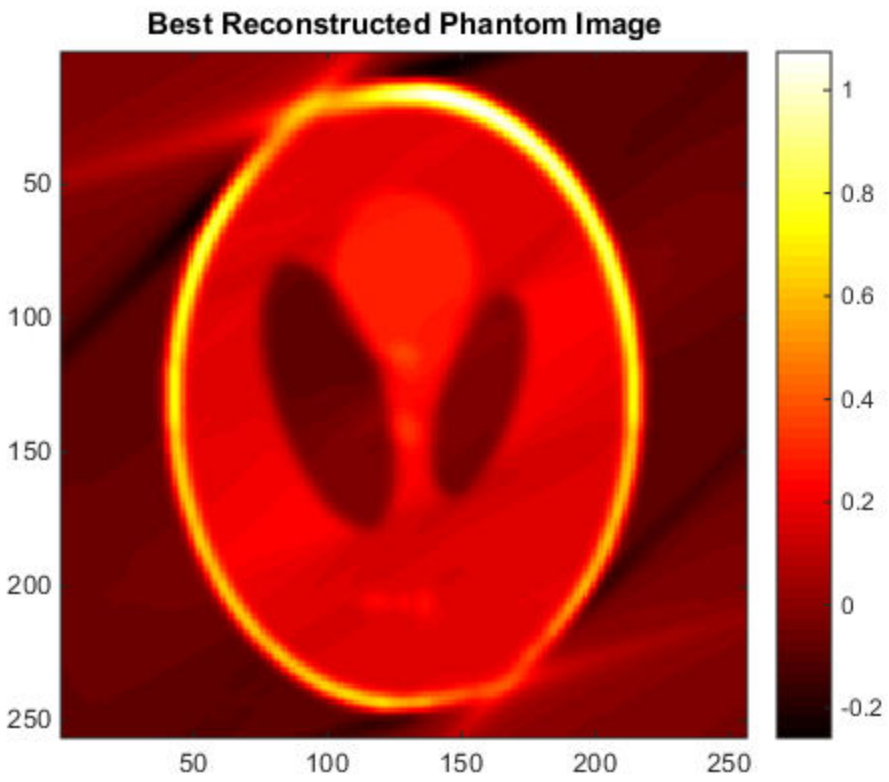



Phantom Image
50
100
150
200
250
50
100
150
200
250
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Chest CT Scan RRMSE vs Theta
0.4
0.39
0.38
0.37
0.36
0.35
0.34
0.33
0.32
0.31
0
20
40
60
80
100
120
140
160
180


Best Reconstructed Chest CT Scan
50
100
150
200
250
300
350
400
450
500
100
200
300
400
500
80
60
40
20
0
-20


Phantom Image RRMSE vs Theta
0.375
0.37
0.365
0.36
0.355
0.35
0.345
0.34
0.335
0.33
0.325
0
20
40
60
80
100
120
140
160
180



Published with MATLAB® R2014b