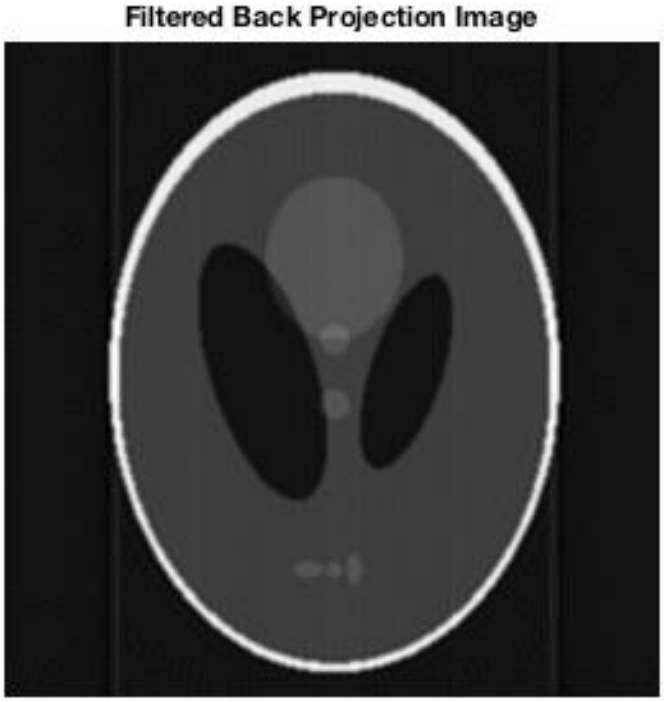
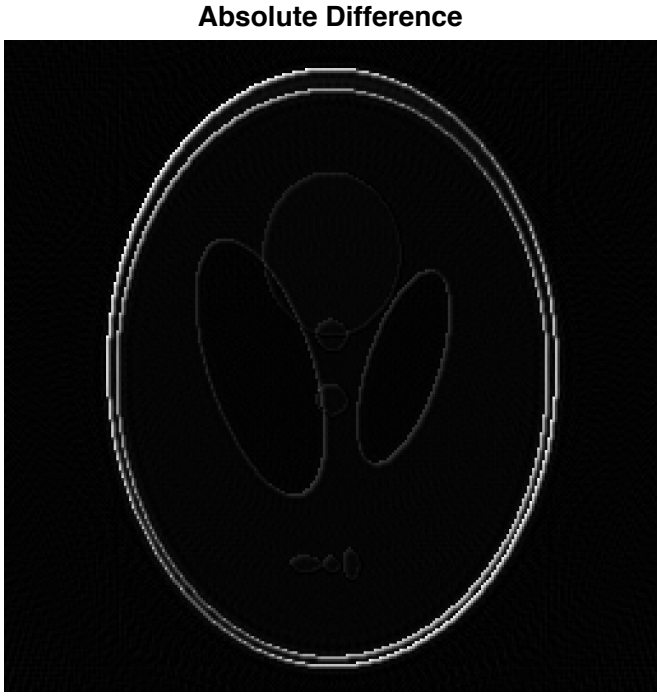
HW5

Hongyi Zhou

1.





MATLAB code:

```
clear;
close all;
%% Load image
img = phantom('Modified Shepp-Logan',256);
img = img/max(max(img));
theta = 0:1:180;
rad = radon(img, theta);
npad = 200;
ramlak_thres = 180;

%% filter before back projection
R_fbp = zeros(size(rad));
for idx = 1:length(theta)
    % Get a slice's Fourier transform
    R_slice = rad(:, idx);
    R_slice = [R_slice; zeros(npad, 1) ]';
    F_slice = fftshift(fft(R_slice));

    % Create ramp in Fourier domain
    ramp = -(length(F_slice)-1)/2:(length(F_slice)-1)/2;

    % Filter using ramlak filter
    ramp(1:ramlak_thres) = ramp(ramlak_thres);
    ramp(end-ramlak_thres+1:end) = ramp(end-ramlak_thres+1);
    F_slice = F_slice.*abs(ramp);

    % Take inverse FFT, but remember to use ifftshift instead of fftshift
    R_fbp_slice = real(ifft(ifftshift(F_slice)));

    % Now save the Radon slice
    R_fbp(:, idx) = R_fbp_slice(1:size(rad, 1))';
end
%% back projection
fbp_img = zeros(size(rad,1));
temp = zeros(size(rad,1));
for i = 0:1:180
    %smearing along y axis
    temp = ((R_fbp(:,i+1))*ones(1, size(rad, 1)))';
    temp = imrotate(temp, i, 'bilinear', 'crop');
    fbp_img = fbp_img + temp;
end

fbp_img = fbp_img(56:311, 56:311);
fbp_img = fbp_img/max(max(fbp_img));
%
figure
imshow(img);
ylabel('x'),xlabel('y'),title('Original Image');

figure
imshow(fbp_img, []);
ylabel('x'),xlabel('y'),title('Filtered Back Projection Image');

figure
l = imabsdiff(img, fbp_img);
imshow(l, []);
ylabel('x'),xlabel('y'),title('Absolute Difference');
```

2.

2. radon transform of two images: $i_1(x,y) \rightarrow r_1(\alpha,\theta)$

$i_2(x,y) \rightarrow r_2(\alpha,\theta)$

$i_3(x,y) = (i_1 * i_2)(x,y)$ 2D convolution of i_1 and i_2

From convolution theorem: FT of convolution of 2 signals is the pointwise product

let I_3, I_2, I_1 be 2D FT of i_3, i_2, i_1 of their FT.

$$I_3 = I_1 \cdot I_2$$

If we take a slice at an angle θ from I_3,

From fourier slice theorem, the slice of 2D FT of a signal (at angle θ) is equal to the 1D FT of line integral of signal at angle θ

we can write, for given θ. $I_3 = I_1 \cdot I_2 = G_1 \cdot G_2$ ← G_1, G_2 are 1D FT of line integrals of i_1, i_2 at angle θ i.e. FT of radon

$G_{3\theta}(\alpha) = G_{1\theta}(\alpha)\, G_{2\theta}(\alpha)$ (G are FT of radon)

therefore $r_{3\theta}(\alpha) = r_{1\theta}(\alpha) * r_{2\theta}(\alpha)$

$r_3(\alpha,\theta) = r_1(\alpha,\theta) * r_2(\alpha,\theta)$

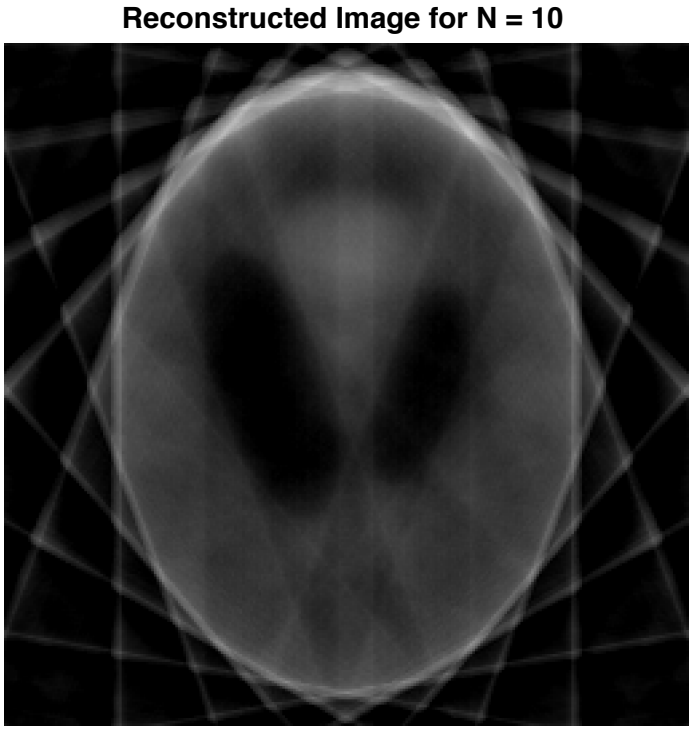
$= (r_1 * r_2)(\alpha,\theta)$

3.

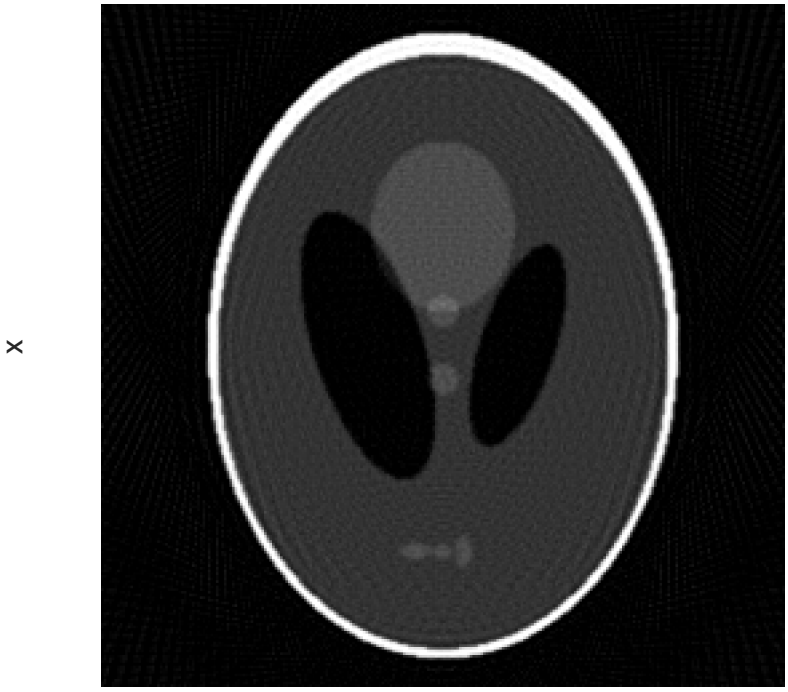
Deliverable A:

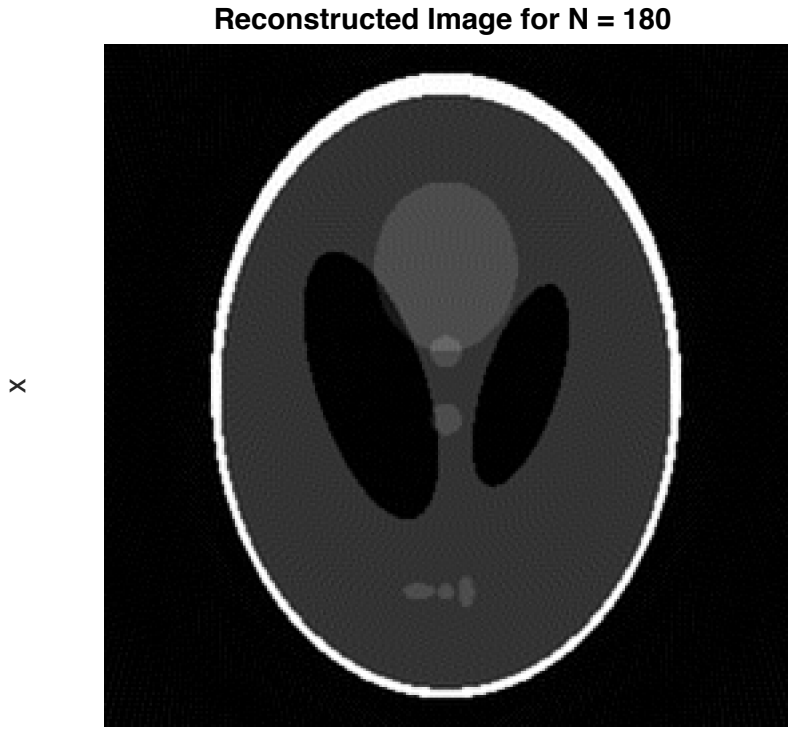
```
function x = least_square(rad, theta, s)
    ra_siz = size(rad);
    ira = iradon(rad, theta, 'linear', 'none', s);

    %%constants
    cgs_iters = 100;              % Number of conjugate gradient descent
iterations
    cgs_tol = 1e-4;              % CGS tolerance.
    %lambda_reg = 1;           % Regularization parameter for regularized
least squares.

    %%function handle
    Aradon = @(img) radon(reshape(img, s, s), theta);
    Aradon_adjoint = @(rad_img) iradon(reshape(rad_img, ra_siz), theta,
'linear', 'none', 1, s);


    %Dxadj = @(z) vec(Dxadjoint(z));
    %Dxfor = @(z) vec(Dx(z));
    %Dyadj = @(z) vec(Dyadjoint(z));
    %Dyfor = @(z) vec(Dy(z));

    A = @(x) vec(Aradon_adjoint(Aradon(x))) ;
    %A = @(x) vec(Aradon_adjoint(Aradon(x))) + lambda_reg *
((Dxadj(Dxfor(x)))+(Dyadj(Dyfor(x))));
    b = vec(ira);

    x = cgs(A, b, cgs_tol, cgs_iters);
    x = reshape(x, s, s);

end
```

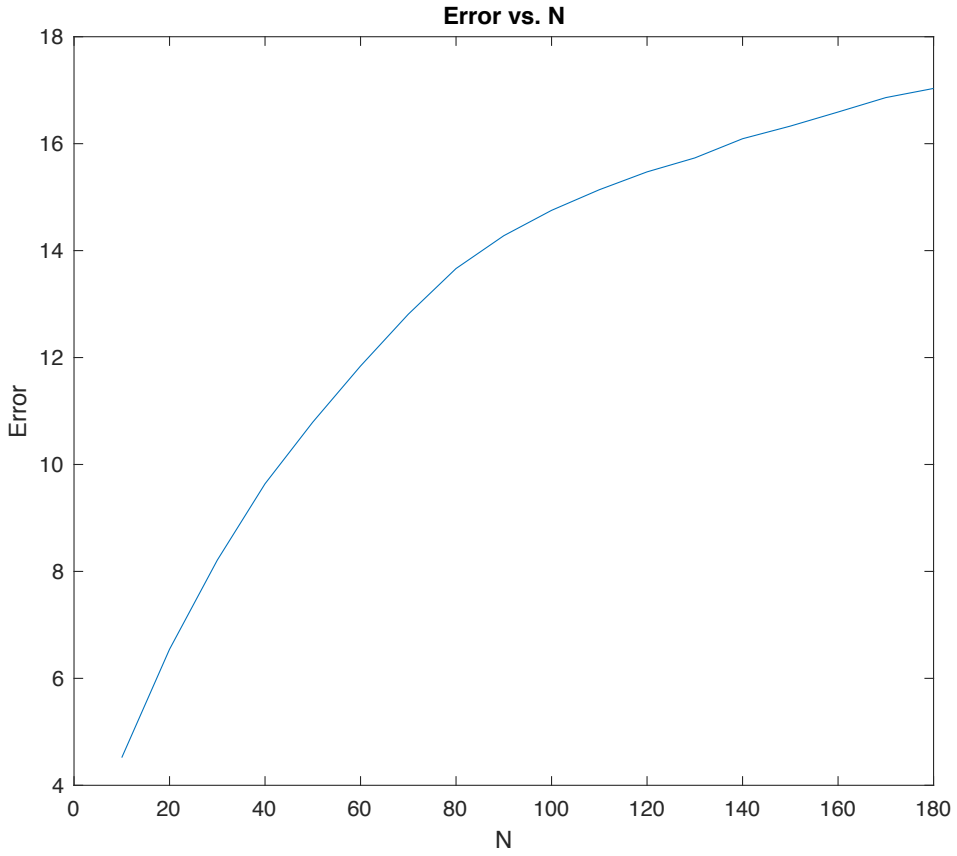
Deliverable B:



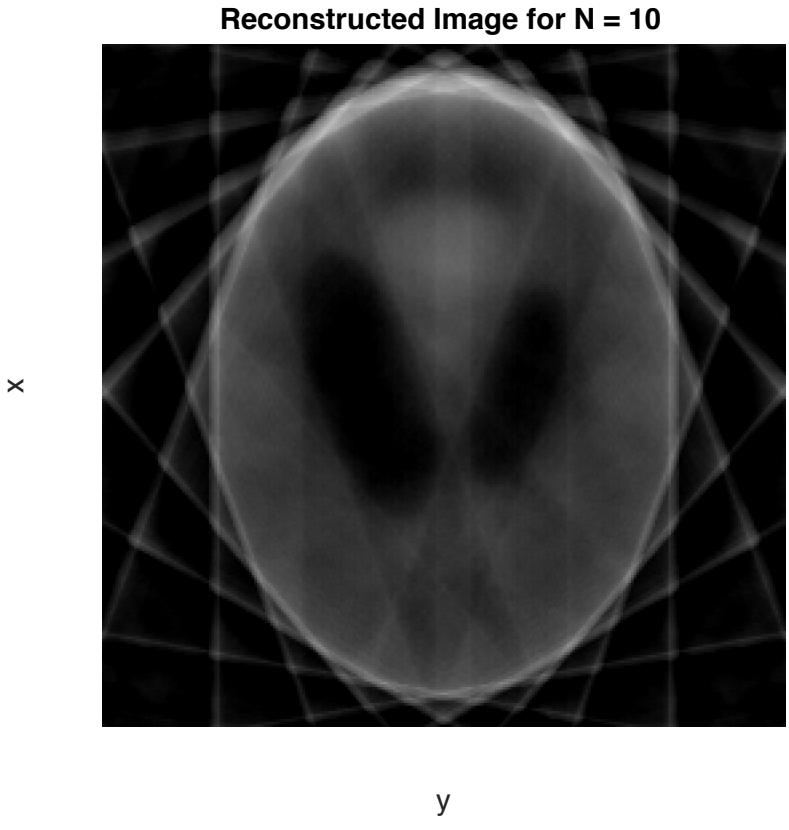
D

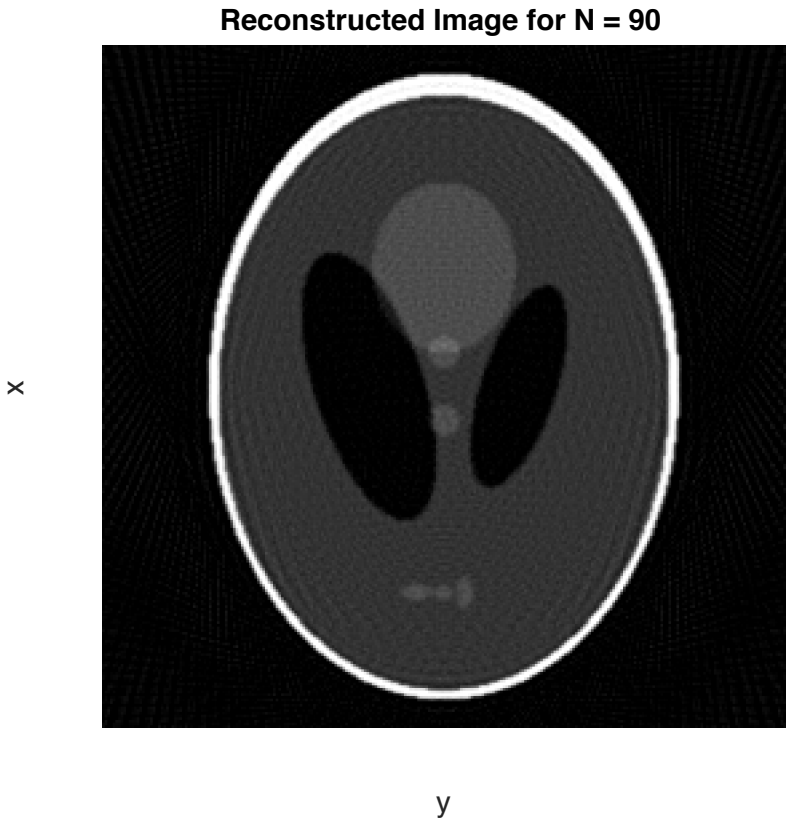


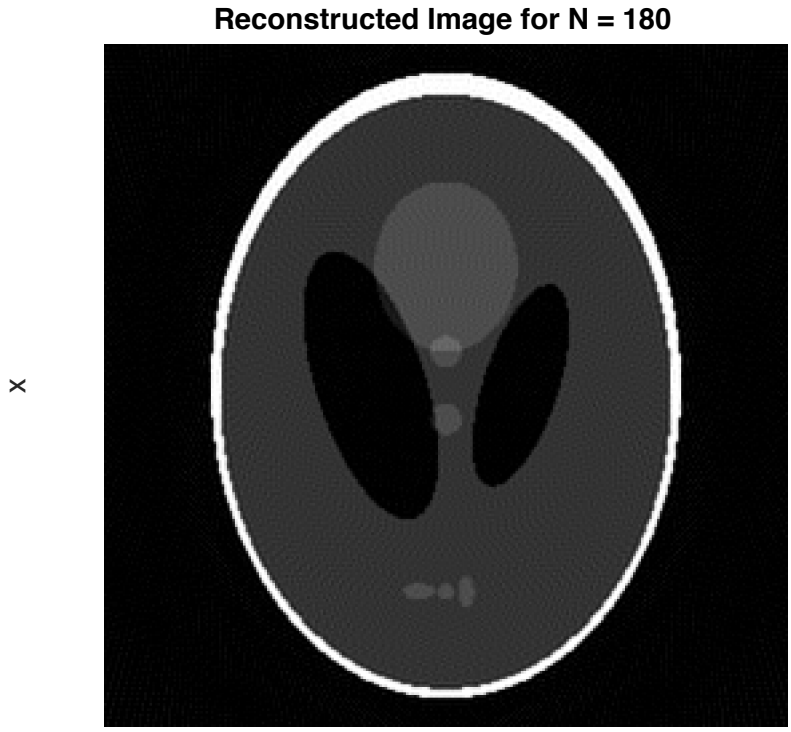

Reconstructed Image for N = 180
x
y

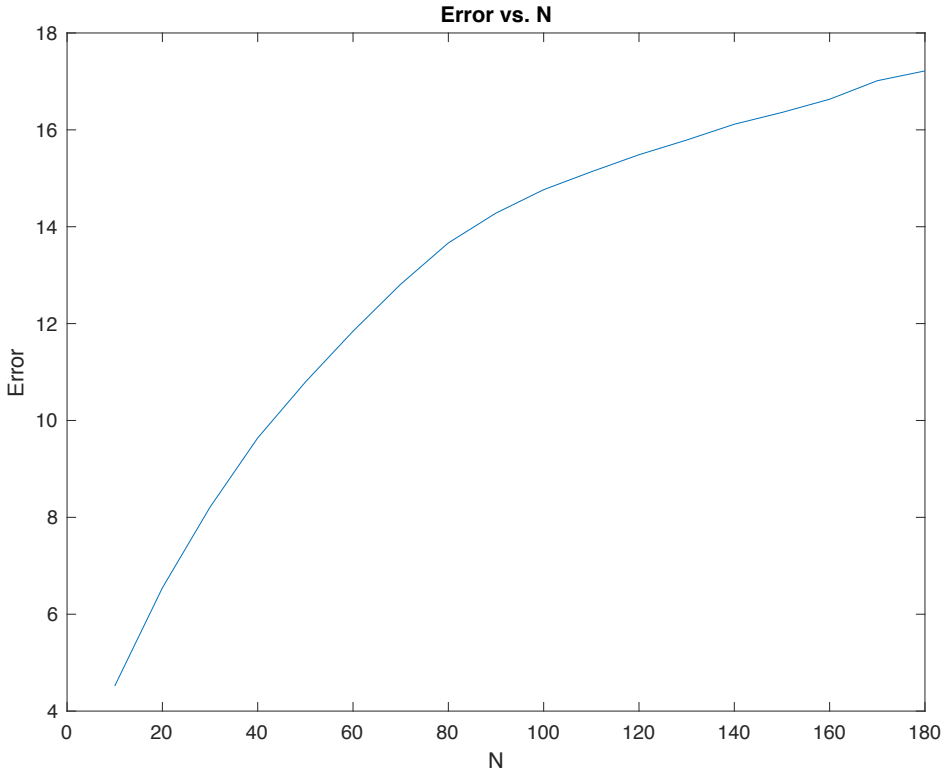

Error vs. N
Error
N
18
16
14
12
10
8
6
4
0
20
40
60
80
100
120
140
160
180

Deliverable C:






Reconstructed Image for N = 180
x
y


Error vs. N
Error
N
18
16
14
12
10
8
6
4
0
20
40
60
80
100
120
140
160
180

MATLAB code:

```
clear
close all
%% Load image
img = phantom('Modified Shepp-Logan',256);
M = 256;
N = linspace(10,180,18); % number of angles
siz = size(img);

%% Simulate different angles
err = zeros(18,1);

for i = 1:numel(N)
    %disp(i)
    Theta = 0:180/(N(1,i)-1):180;
    rad = radon(img, Theta);
    imgrec = least_square(rad, Theta, siz(1));
    if i == 1 || i == 9 || i == 18
        figure
        imshow(imgrec);
        str = sprintf('Reconstructed Image for N = %d',i*10);
        ylabel('x'),xlabel('y'),title(str);
    end
    err(i,1) = error(img, imgrec);
end

figure
plot(N,err);
title('Error vs. N')
ylabel('Error')
xlabel('N')
%%
function x = least_square(rad, theta, s)
    ra_siz = size(rad);
    ira = iradon(rad, theta, 'linear', 'none', s);

    %%constants
    cgs_iters = 100;              % Number of conjugate gradient descent
iterations
    cgs_tol = 1e-4;               % CGS tolerance.
    lambda_reg = 1e-02;           % Regularization parameter for
regularized least squares.

    %%function handle
    Aradon = @(img) radon(reshape(img, s, s), theta);
    Aradon_adjoint = @(rad_img) iradon(reshape(rad_img, ra_siz), theta,
'linear', 'none', 1, s);


    Dxadj = @(z) vec(Dxadjoint(z));
    Dxfor = @(z) vec(Dx(z));
    Dyadj = @(z) vec(Dyadjoint(z));
    Dyfor = @(z) vec(Dy(z));

%     A = @(x) vec(Aradon_adjoint(Aradon(x))) ;
    A = @(x) vec(Aradon_adjoint(Aradon(x))) + lambda_reg *
((Dxadj(Dxfor(x)))+(Dyadj(Dyfor(x))));
    b = vec(ira);
```

```
    x = cgs(A, b, cgs_tol, cgs_iters);
    x = reshape(x, s, s);

end

function e = error(img, imgrec)
    e = -20*log10( norm(img(:) - imgrec(:))/norm(img(:)));
end

%% Dx
function y = Dx(I)
    i = reshape(I, [256,256]);
    y = i(:, 2:end)-i(:, 1:end-1);
    y = vec(y);
end

%% Dx adjoint
function I = Dxadjoint(y)
    Y = reshape(y, [256,255]);
    I = zeros(256,256);
    I(:,1) = -Y(:,1);
    I(:,2:end-1) = Y(:,1:end-1) - Y(:,2:end);
    I(:,end) = Y(:,end);
    I = vec(I);
end

%% Dy
function y = Dy(I)
    i = reshape(I, [256,256]);
    y = i(2:end, :)-i(1:end-1, :);
    y = vec(y);
end

%% Dy adjoint
function I = Dyadjoint(y)
    Y = reshape(y, [255,256]);
    I = zeros(256,256);
    I(1,:) = -Y(1,:);
    I(2:end-1,:) = Y(1:end-1,:) - Y(2:end,:);
    I(end,:) = Y(end,:);
    I = vec(I);
end
```

4. 4 hours + 2 office hours