Berfin Kavşut
21602450
EEE475
April 27, 2020

REPORT FOR HOMEWORK#3

PART 1 – System Matrix and SVD

Q1.1 - Preparing System Matrix

Plots

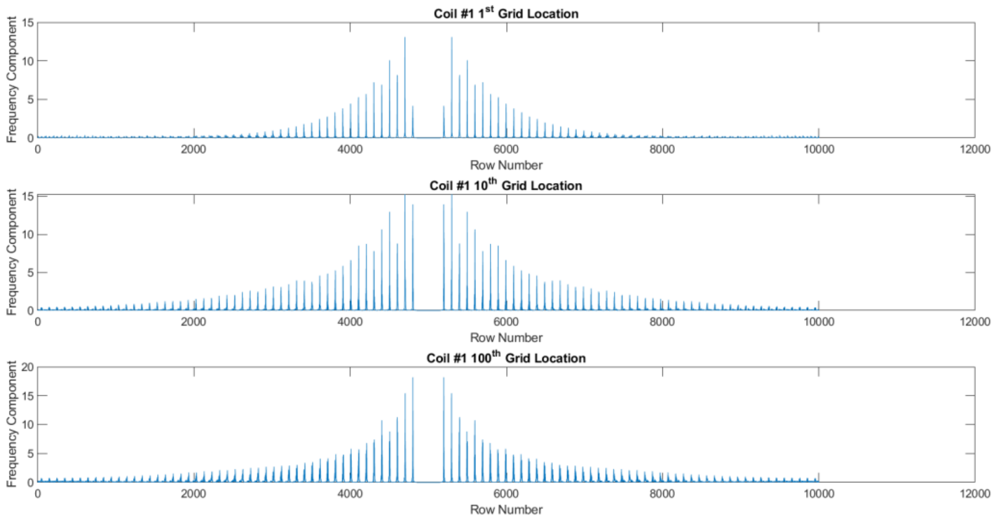


Figure 1. Frequency Components of Coil #1

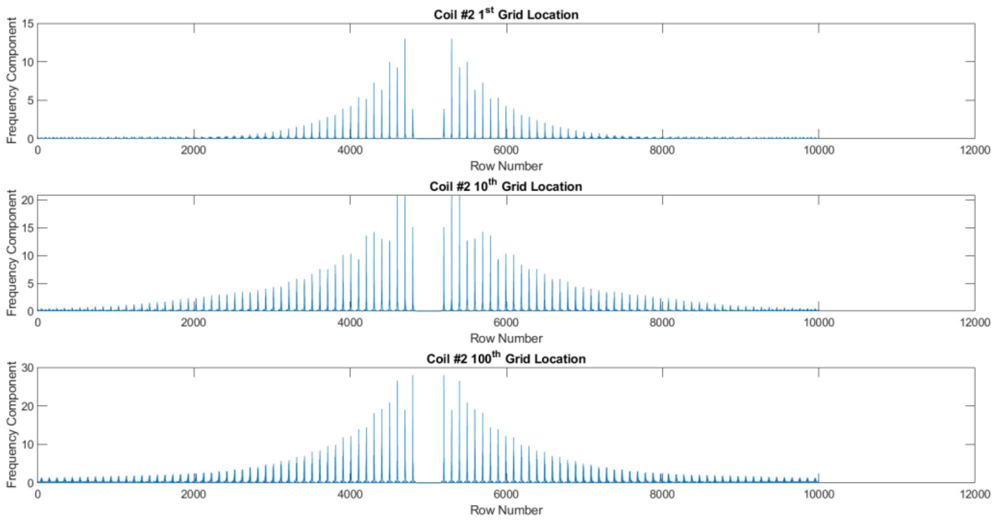


Figure 2. Frequency Components of Coil #2

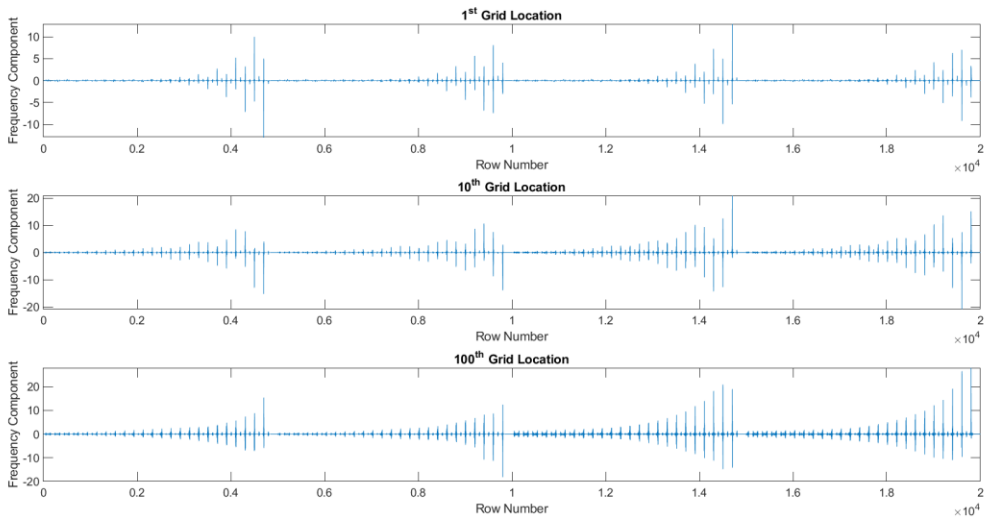


Figure 3. Frequency Components of System Matrix (S)

Code

```
disp('1.1');
disp('Preparing System Matrix');

load("SM_40x40.mat");

%For both coils, plot the magnitude spectrum for
%calibration measurements at three different grid locations
figure;set(gcf, 'WindowState', 'maximized');
subplot(3,1,1); plot(abs(fftc(SM_coil1(:,1))));
title('Coil #1 1^{st} Grid Location');
xlabel('Row Number'); ylabel('Frequency Component');
subplot(3,1,2); plot(abs(fftc(SM_coil1(:,10))));
title('Coil #1 10^{th} Grid Location');
xlabel('Row Number'); ylabel('Frequency Component');
subplot(3,1,3); plot(abs(fftc(SM_coil1(:,100))));
title('Coil #1 100^{th} Grid Location');
xlabel('Row Number'); ylabel('Frequency Component');
saveas(gcf,'1.1_Coil#1.png');

figure;set(gcf, 'WindowState', 'maximized');
subplot(3,1,1); plot(abs(fftc(SM_coil2(:,1))));
title('Coil #2 1^{st} Grid Location');
xlabel('Row Number'); ylabel('Frequency Component');
subplot(3,1,2); plot(abs(fftc(SM_coil2(:,10))));
title('Coil #2 10^{th} Grid Location');
xlabel('Row Number'); ylabel('Frequency Component');
subplot(3,1,3); plot(abs(fftc(SM_coil2(:,100))));
title('Coil #2 100^{th} Grid Location');
xlabel('Row Number'); ylabel('Frequency Component');
saveas(gcf,'1.1_Coil#2.png');

%prepare the system matrix

%FFT from calibration measurement from Coil #1
S1 = zeros(size(SM_coil1));
for n = 1:1600
    %size: 10001x1 for each column vector
    S1(:,n) = fftc(SM_coil1(:,n));
end
%size: 5001x1600
S1(5001:10001,:) = [];
%size: 10002x1600
S1 = [real(S1);imag(S1)];

%FFT from calibration measurement from Coil #1
S2 = zeros(size(SM_coil2));
for n = 1:1600
    %size: 10001x1 for each column vector
    S2(:,n) = fftc(SM_coil2(:,n));
end
%size: 5001x1600
S2(5001:10001,:) = [];
%size: 10002x1600
S2 = [real(S2);imag(S2)];

%size of system matrix: 20004x1600
S = [S1;S2];

%Plot three different columns of the system matrix
figure;set(gcf, 'WindowState', 'maximized');
subplot(3,1,1); plot( S(:,1) );
title('1^{st} Grid Location');
xlabel('Row Number'); ylabel('Frequency Component');
subplot(3,1,2); plot( S(:,10) );
title('10^{th} Grid Location');
 xlabel('Row Number'); ylabel('Frequency Component');
subplot(3,1,3); plot( S(:,100) );
title('100^{th} Grid Location');
 xlabel('Row Number'); ylabel('Frequency Component');
saveas(gcf,'1.1_System_Matrix.png');

%save S, system matrix, as a mat-file
save("S.mat",'S');
```

Q1.2 - Preparing Measurement Vector

Plots

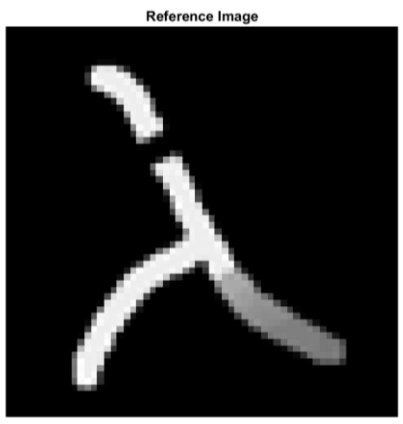


Figure 4. Reference Image (Phantom)

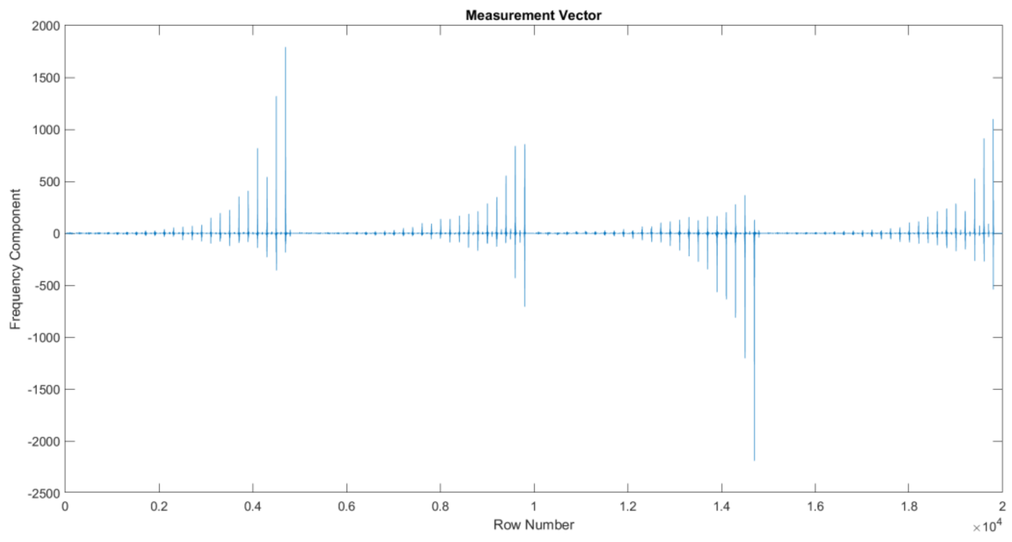


Figure 5. Frequency Components of Measurement Vector (u)

Code

```
disp('1.2');
disp("Preparing Measurement Vector");

load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');

%display phantom, which will be reference image
figure;set(gcf, 'WindowState', 'maximized');
imshow(phantom,[]); title('Reference Image');
saveas(gcf,'1.2_phantom.png');

%prepare measurement vector

%measurement from coil #1
%size: 10001x1
u1 = fftc(meas_coil1);
%size: 5001x1
u1(5001:10001) = [];
%size: 10002x1
u1 = [real(u1);imag(u1)];

%measurement from coil #2
%size: 10001x1
u2 = fftc(meas_coil2);
%size: 5001x1
u2(5001:10001) = [];
%size: 10002x1
u2 = [real(u2);imag(u2)];
%20004x1
u = [u1;u2];

figure;set(gcf, 'WindowState', 'maximized');
plot(u); title('Measurement Vector');
xlabel('Row Number'); ylabel('Frequency Component');
saveas(gcf,'1.2_measurement_vector.png');

%save u, measurement vector, as a mat-file
save("u.mat",'u');
```

Q1.3 – SVD

Condition number: is 725.6196 and this value is checked with MATLAB’s built-in *cond* function.

Plots

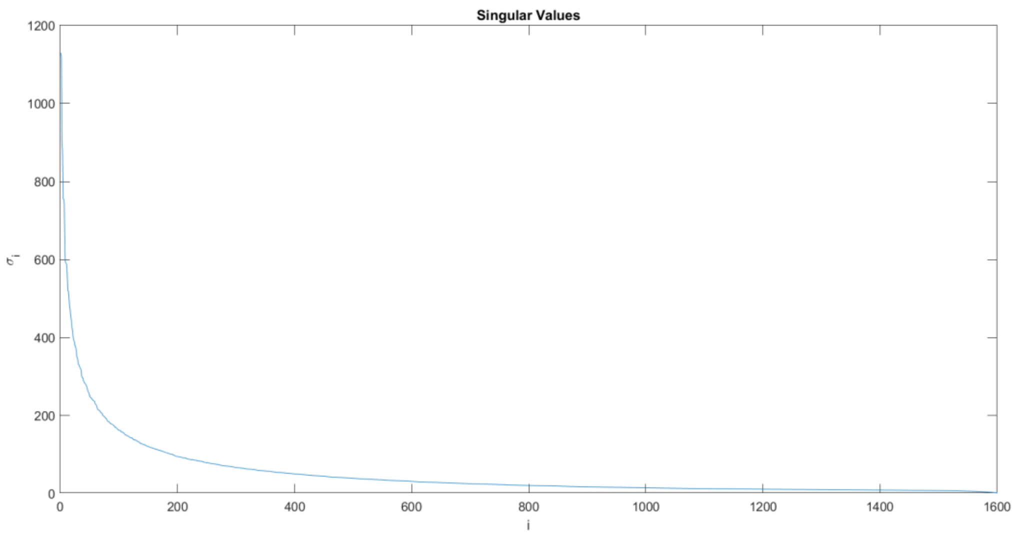


Figure 6. Singular Values

Code

```
disp('1.3');
disp("SVD");

load("S.mat");

[U,Sigma,V] = svd(S,'econ');
%U: size of 20004x1600
%Sigma: diagonal matrix, size of 1600x1600
%V: size of 1600x1600
size_U = size(U)
size_Sigma = size(Sigma)
size_V = size(V)

%plot the singular values (i.e. diagonal entries of Sigma)
singular_values = diag(Sigma);
figure;set(gcf, 'WindowState', 'maximized');
plot(singular_values); title('Singular Values');
xlabel('i'); ylabel('{\sigma}_i');
saveas(gcf,'1.3_singular_values.png');

condition_no = singular_values(1)/singular_values(end);
%condition_no_matlab = cond(Sigma), gives the same result of 725.6196
disp(strcat('Condition Number: ', num2str(condition_no)));
```

Q1.4 – Reconstruction via SVD

PSNR and SSIM measures are really bad right now. High frequency components can be seen in our result image. Still, phantom can be seen roughly in our result. With Truncated SVD method, we will improve our results by eliminating small singular values. So the contribution of very high frequencies, which are actually noise, will be eliminated.

Results

PSNR:8.6617

SSIM:0.19413

Plots

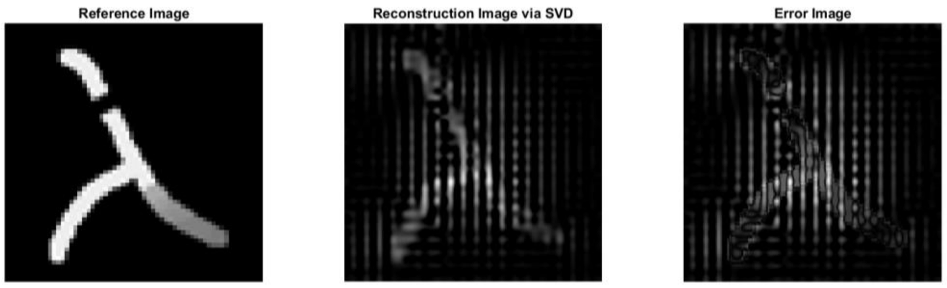


Figure 7. Images of Reconstruction via SVD

Code

```
disp('1.4');
disp("Reconstruction via SVD");

load("S.mat"); load("u.mat");
load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');

%SVD of System Matrix
[U,Sigma,V] = svd(S,'econ');
%compute the image c using Moore-Pensore
%pseudo inverse from SVD
c = V*inv(Sigma)*U'*u; %conjugate transpose of U = U'
%alternative solution: c = V*((U'*u)./diag(Sigma));

%construct the image
ima = reshape(c,40,40);
%resize the image for 280x280
ima = imresize(ima, [280 280]);
%set negative values to zero
ima(ima<0) = 0;

figure;set(gcf, 'WindowState', 'maximized');
subplot(1,3,1); imshow(phantom,[]);
title('Reference Image');
subplot(1,3,2); imshow(ima,[]);
title('Reconstruction Image via SVD');
subplot(1,3,3); imshow(abs(phantom-ima),[]);
title('Error Image');
saveas(gcf,'1.4.png');
```

```
%IQA measurements
PSNR = psnr(ima,phantom);
SSIM = ssim(ima,phantom);
disp(strcat('PSNR: ', num2str(PSNR)));
disp(strcat('SSIM: ', num2str(SSIM)));
```

Q1.5 – Truncated SVD

Our visual results improved for Truncated SVD. Also, IQA measures improved. PSNR increased from 8.6617 to 17.0779. SSIM increased from 0.19413 to 0.39036. We got rid of most of very high frequency components, which are noise.

Results

Condition Number:100.0179

N:1199

PSNR:17.0779

SSIM:0.39036

Plots

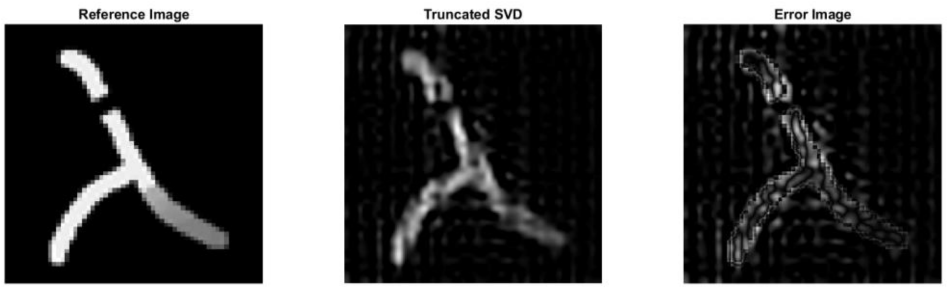


Figure 8. Images of Truncated SVD

Code

```
disp('1.5');
disp("Truncated SVD");

load("S.mat");load("u.mat");
load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');

[U,Sigma,V] = svd(S,'econ');
singular_values = diag(Sigma);

condition_no = 0;
for i = 1: length(singular_values)
    condition_no = singular_values(1)/singular_values(i);
    if(condition_no >=100)
        N = i;
        break;
    end
end

disp(strcat('Condition Number: ', num2str(condition_no)));
disp(strcat('N: ', num2str(N)));
```

```
U(:,(N+1):end) = []; %U: 20004xN
V(:,(N+1):end) = []; %V: 1600xN
Sigma = Sigma(1:N,1:N); %Sigma: NxN

c = V*inv(Sigma)*U'*u;

ima = reshape(c,40,40);
ima = imresize(ima, [280 280]);
ima(ima<0) = 0;

figure;set(gcf, 'WindowState', 'maximized');
subplot(1,3,1); imshow(phantom,[]);
title('Reference Image');
subplot(1,3,2); imshow(ima,[]);
title('Truncated SVD');
subplot(1,3,3); imshow(abs(phantom-ima),[]);
title('Error Image');
saveas(gcf,'1.5.png');

%IQA measurements
PSNR = psnr(ima,phantom);
SSIM = ssim(ima,phantom);
disp(strcat('PSNR: ', num2str(PSNR)));
disp(strcat('SSIM: ', num2str(SSIM)));
```

Q1.6 – Truncated SVD with Different Condition Numbers

We tried Truncated SVD with condition numbers of 5, 10, 20, 30, 50.

The best condition number was 20. The highest PSNR was for 20 and the highest SSIM was for 10. The best visual result was for 20.

When condition number is too small, the image is effectively low-pass filtered. High frequency components are discarded as condition number decreases. Basically, we do not use as much as frequency component we desire, which can also be seen from the number of N: When the condition number is too large, we cannot have the effect of low-pass filter we desire. Therefore, condition number should be somewhere between for given values.

Results

Condition Number:5.1238

N:63

PSNR for Condition Number=5.1238:15.2384

SSIM for Condition Number=5.1238:0.55669

Condition Number:10.0328

N:166

PSNR for Condition Number=10.0328:18.5374

SSIM for Condition Number=10.0328:0.57585

Condition Number:20.0587

N:360

PSNR for Condition Number=20.0587:20.6856

SSIM for Condition Number=20.0587:0.56743

Condition Number:30.0033

N:518

PSNR for Condition Number=30.0033:19.8441

SSIM for Condition Number=30.0033:0.54387

Condition Number:50.0287

N:761

PSNR for Condition Number=50.0287:18.4993

SSIM for Condition Number=50.0287:0.49436

Plots

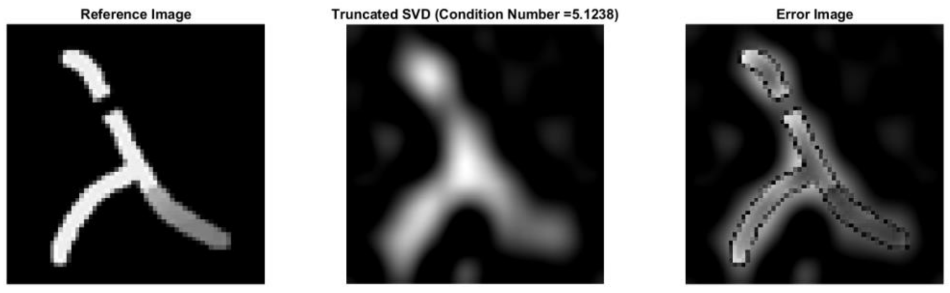


Figure 9. Images of Truncated SVD with Condition Number = 5.1238

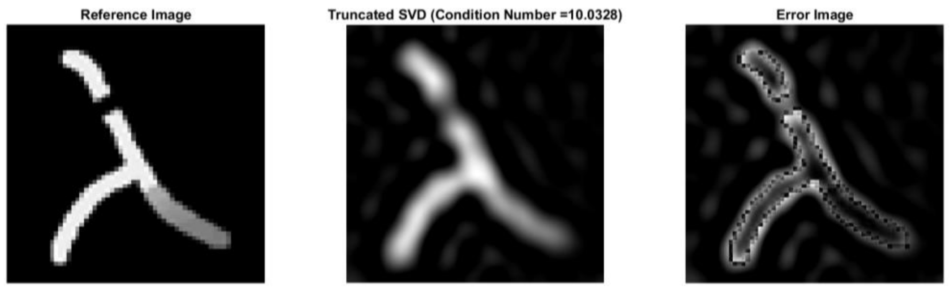


Figure 10. Images of Truncated SVD with Condition Number = 10.0328

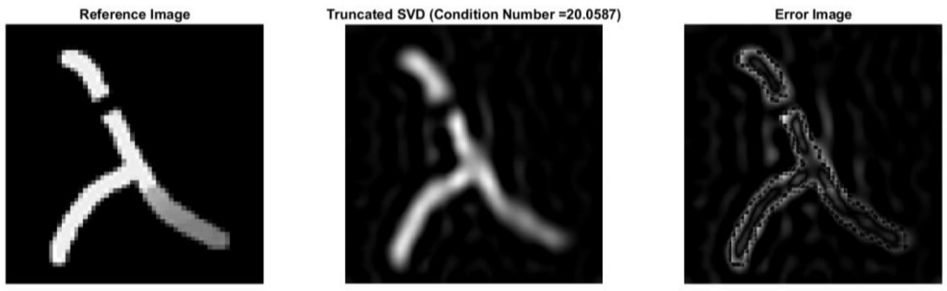


Figure 11. Images of Truncated SVD with Condition Number = 20.0587

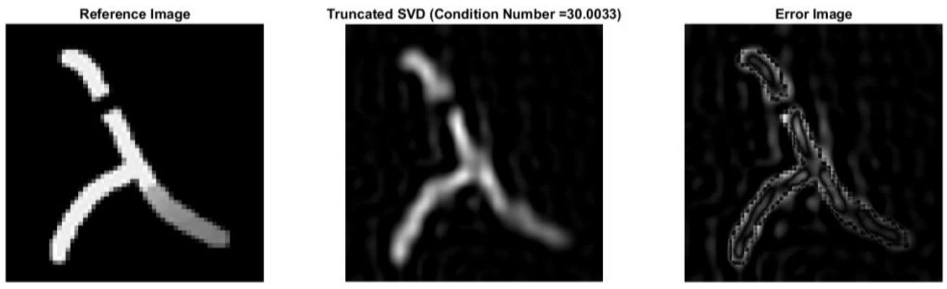


Figure 12. Images of Truncated SVD with Condition Number = 30.0033

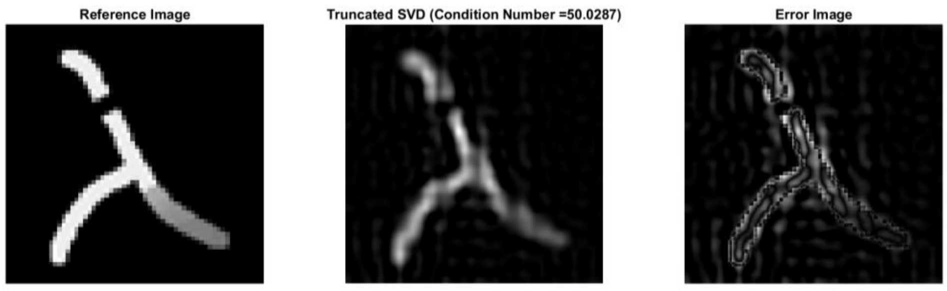


Figure 13. Images of Truncated SVD with Condition Number = 50.0287

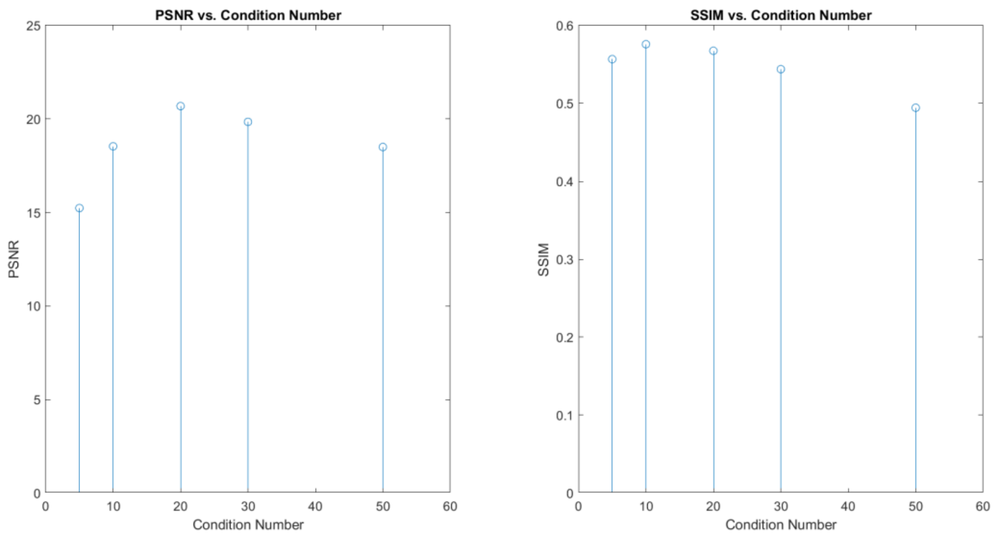


Figure 14. PSNR and SSIM vs. Condition Number

Code

```
disp('1.6');
disp("Truncated SVD with Different Condition Numbers");

load("S.mat"); load("u.mat");
load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');

[U_ref,Sigma_ref,V_ref] = svd(S,'econ');
singular_values = diag(Sigma_ref);

cond = [5, 10, 20, 30, 50];
for k = 1:length(cond)
    condition_no = 0;
    for i = 1: length(singular_values)
        condition_no = singular_values(1)/singular_values(i);
        if(condition_no >= cond(k))
            N = i;
            break;
        end
    end

    disp(strcat('Condition Number: ', num2str(condition_no)));
    disp(strcat('N: ', num2str(N)));

    U = U_ref(:,1:N); %U: 20004xN
    V = V_ref(:,1:N); %V: 1600xN
    Sigma = Sigma_ref(1:N,1:N); %Sigma: NxN

    c = V*inv(Sigma)*U'*u;

    ima = reshape(c,40,40);
    ima = imresize(ima, [280 280]);
    ima(ima<0) = 0;

    figure;set(gcf, 'WindowState', 'maximized');
    subplot(1,3,1); imshow(phantom,[]);
    title('Reference Image');
    subplot(1,3,2); imshow(ima,[]);
    title(strcat('Truncated SVD (Condition Number = ',
num2str(condition_no), ')'));
    subplot(1,3,3); imshow(abs(phantom-ima),[]);
    title('Error Image');
    saveas(gcf,strcat('1.6_cond', num2str( cond(k) ), '.png' ));

    %IQA measurements
    PSNR(k) = psnr(ima,phantom);
    SSIM(k) = ssim(ima,phantom);
    %disp(strcat('Condition Number: ', num2str(cond(k))));
    disp(strcat('PSNR for Condition Number=', num2str(condition_no),
':',num2str(PSNR(k))));
    disp(strcat('SSIM for Condition Number=', num2str(condition_no),
':',num2str(SSIM(k))));

end

figure;set(gcf, 'WindowState', 'maximized');
subplot(1,2,1); stem(cond,PSNR); xlim([0 60]);
title('PSNR vs. Condition Number');
xlabel('Condition Number'); ylabel('PSNR');
subplot(1,2,2); stem(cond,SSIM); xlim([0 60]);
title('SSIM vs. Condition Number');
xlabel('Condition Number'); ylabel('SSIM');
saveas(gcf,'1.6_PSNR_SSIM.png');

% best condition number: 20
```

Q1.7 – Filtered SVD

Similar to Truncated SVD, our aim is to get rid of very high frequencies to eliminate noise in our result image. This time, we do not discard the high freqeuncy components but weight them down. Singular value comes as a denominator for frequency component when computing image signal. In Figure 15, the filtered singular values are higher for very high frequency components. This means that we are low-pass filtering our result image.

Result

Regularization Parameter:1758.7353

PSNR:20.4461

SSIM:0.65643

Plots

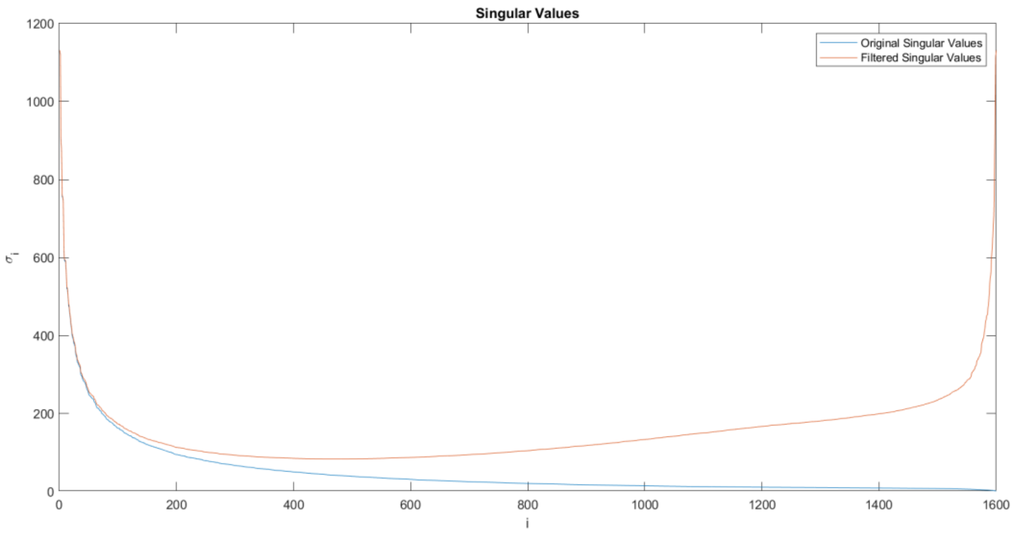


Figure 15. Original Singular Values vs. Filtered Singular Values

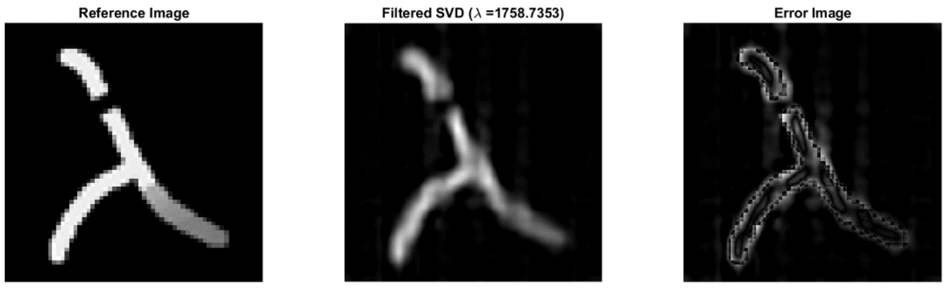


Figure 16. Filtered SVD with Lambda = 1758.7353

Code

```
disp('1.7');
disp("Filtered SVD");

load("S.mat"); load("u.mat");
load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');

[U,Sigma,V] = svd(S,'econ');
singular_values = diag(Sigma);

%lambda: regularization parameter
%lambda should be much larger than square of minimum singular value
%and much smaller than square of maximum singular value
%for starting, lambda = sing_max * sing_min;
singular_max = singular_values(1);
singular_min = singular_values(end);
lambda = singular_max * singular_min;
disp(strcat('Regularization Parameter: ', num2str(lambda)));

%find the new singular values (filtered singular values)
%replace them in new Sigma
new_Sigma =  Sigma + (lambda./Sigma);
```

```
%singular values
original_singular_values = diag(Sigma);
filtered_singular_values = diag(new_Sigma);

%comparison of original singular values and filtered singular values
%singular values increased for higher frequency components
%so that the image is effectively low-pass filtered, eliminated noise
figure; set(gcf, 'WindowState', 'maximized');
plot(original_singular_values); hold on;
plot(filtered_singular_values);
title('Filtered Singular Values');
xlabel('i'); ylabel('{\sigma}_i');
legend('Original Singular Values','Filtered Singular
Values','magenta','orange','Northwest');
saveas(gcf,'1.7_singular_values.png');

c = V*((U'*u)./diag(new_Sigma));

ima = reshape(c,40,40);
ima = imresize(ima, [280 280]);
ima(ima<0) = 0;

figure;set(gcf, 'WindowState', 'maximized');
subplot(1,3,1); imshow(phantom,[]);
title('Reference Image');
subplot(1,3,2); imshow(ima,[]);
title(strcat('Filtered SVD ({\lambda} = ', num2str(lambda), ')'));
subplot(1,3,3); imshow(abs(phantom-ima),[]);
title('Error Image');
saveas(gcf,strcat('1.7_', num2str(lambda), '.png' ));

%IQA measurements
PSNR = psnr(ima,phantom);
SSIM = ssim(ima,phantom);
disp(strcat('PSNR: ', num2str(PSNR)));
disp(strcat('SSIM: ', num2str(SSIM)));
```

Q1.8– Filtered SVD with Different Lambdas

Lambda should be much larger than square of minimum singular value and much smaller than square of maximum singular value. For this, we should look for closest singular values to the left-corner in our singular values plot (Figure 6). The best lambda value comes from the right side of the closest singular values. Practically, it does not come from the closest singular values to the left-corner.

Regularization parameters in the range of from 2500 to 10000 are shown as steps here. I chose regularization parameter to be 4900. This is the best value in terms of PSNR, SSIM and visual assessments.

Results for best lambda is shown later. When it is 1/10 times of lambda, the image is noisy as explained in previous questions. When it is 10 times of lambda, the image look too smooth, i.e. it is low pass filtered and we lost our resolution.

Results

Regularization Parameter:10000

PSNR for Regularization Parameter=10000:19.3031

SSIM for Regularization Parameter=10000:0.72772

Regularization Parameter:8100

PSNR for Regularization Parameter=8100:19.6324

SSIM for Regularization Parameter=8100:0.72442

Regularization Parameter:6400

PSNR for Regularization Parameter=6400:19.9515

SSIM for Regularization Parameter=6400:0.71827

Regularization Parameter:4900

PSNR for Regularization Parameter=4900:20.2363

SSIM for Regularization Parameter=4900:0.7088

Regularization Parameter:3600

PSNR for Regularization Parameter=3600:20.4476

SSIM for Regularization Parameter=3600:0.6951

Regularization Parameter:2500

PSNR for Regularization Parameter=2500:20.5279

SSIM for Regularization Parameter=2500:0.67666

Optimum Regularization Parameter:4900

Regularization Parameter:490

PSNR for Regularization Parameter=490:19.3527

SSIM for Regularization Parameter=490:0.55334

Regularization Parameter:4900

PSNR for Regularization Parameter=4900:20.2363

SSIM for Regularization Parameter=4900:0.7088

Regularization Parameter:49000

PSNR for Regularization Parameter=49000:16.3495

SSIM for Regularization Parameter=49000:0.68738

Plots

<u>Steps to Find Regularization Parameter:</u>

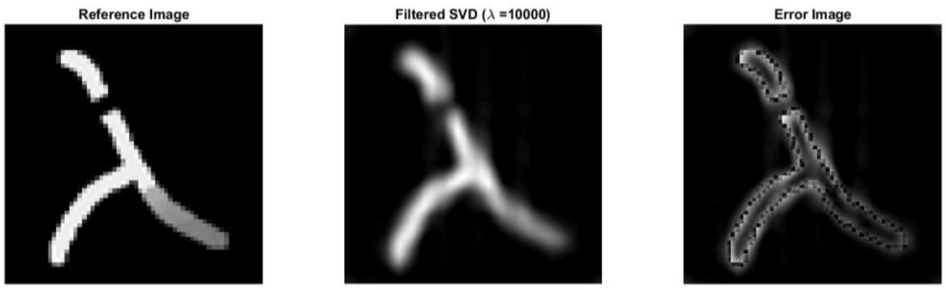


Figure 17. Filtered SVD with Lambda = 10000

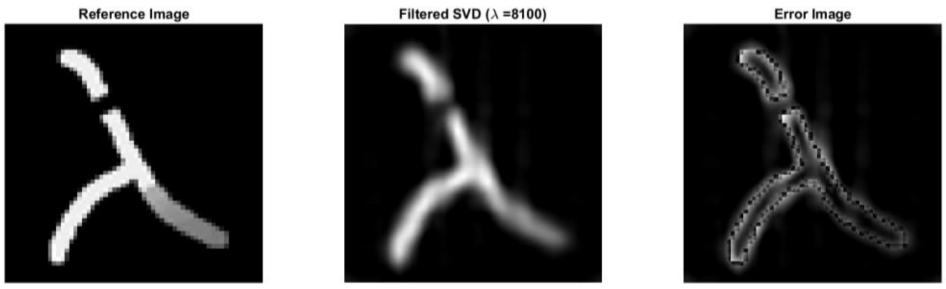


Figure 18. Filtered SVD with Lambda = 8100

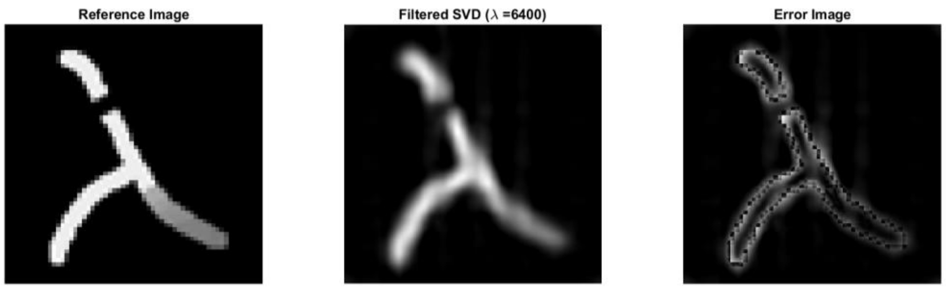


Figure 19. Filtered SVD with Lambda = 6400

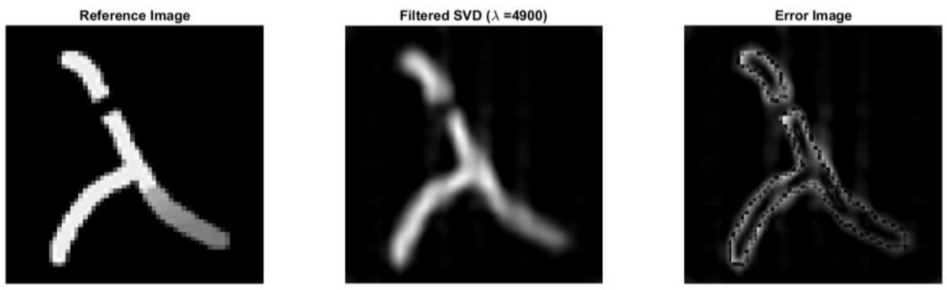


Figure 20. Filtered SVD with Lambda = 4900

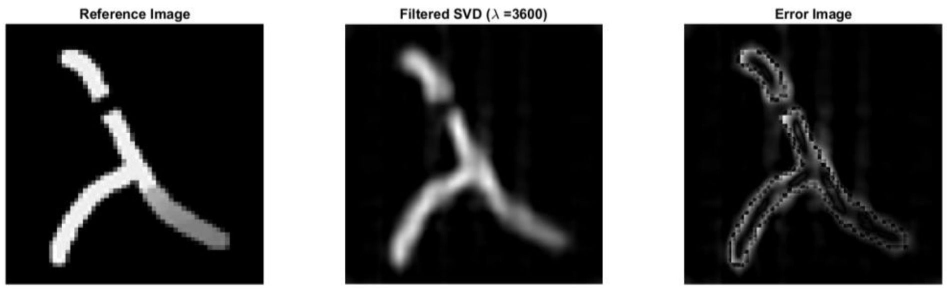


Figure 21. Filtered SVD with Lambda = 3600

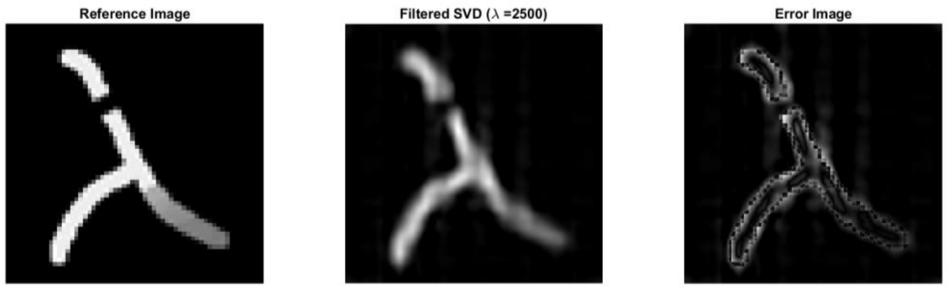


Figure 22. Filtered SVD with Lambda = 2500

<u>Optimum Regularization Parameter:</u>

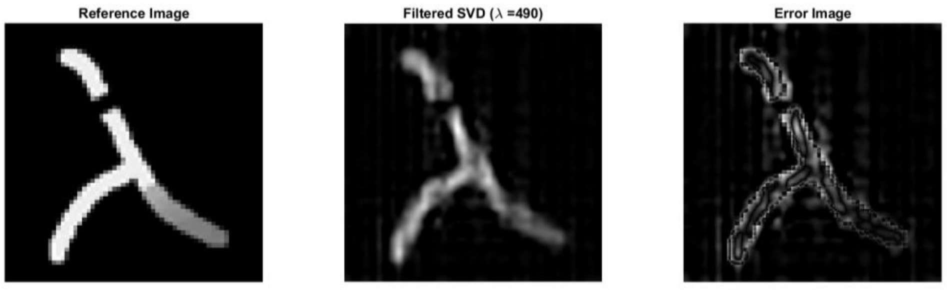


Figure 23. Filtered SVD with Lambda = 490

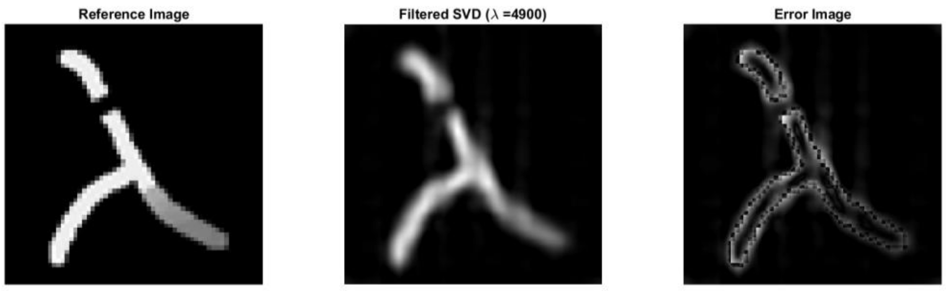


Figure 24. Filtered SVD with Lambda = 4900

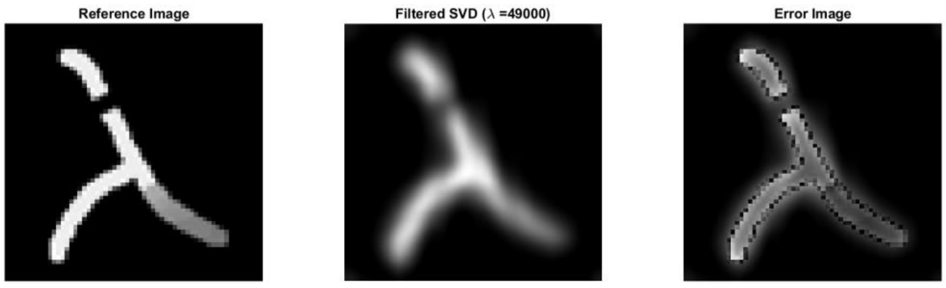


Figure 25. Filtered SVD with Lambda = 49000

Code

```
disp('1.8');
disp("Filtered SVD with Different Lambdas");

load("S.mat"); load("u.mat");
load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');

%show steps for finding optimum lambda
for lambda = [100:-10:50].^2
    disp(strcat('Regularization Parameter: ', num2str(lambda)));

    [U,Sigma,V] = svd(S,'econ');
    new_Sigma =  Sigma + (lambda./Sigma);
    c = V*((U'*u)./diag(new_Sigma));

    ima = reshape(c,40,40);
    ima = imresize(ima, [280 280]);
    ima(ima<0) = 0;

    figure;set(gcf, 'WindowState', 'maximized');
    subplot(1,3,1); imshow(phantom,[]);
    title('Reference Image');
    subplot(1,3,2); imshow(ima,[]);
    title(strcat('Filtered SVD ({\lambda} = ', num2str(lambda), ')'));
    subplot(1,3,3); imshow(abs(phantom-ima),[]);
    title('Error Image');
    saveas(gcf,strcat('1.8_step', num2str(lambda), '.png' ));
    %IQA measurements
    PSNR = psnr(ima,phantom);
    SSIM = ssim(ima,phantom);
    disp(strcat('PSNR for Regularization Parameter=',
num2str(lambda),':',num2str(PSNR)));
    disp(strcat('SSIM for Regularization Parameter=',
num2str(lambda),':',num2str(SSIM)));

end

%lambda (regularization parameter)
opt_lambda = 4900;
disp(strcat('Optimum Regularization Parameter: ', num2str(opt_lambda)));

for lambda = [opt_lambda/10 opt_lambda 10*opt_lambda]
    disp(strcat('Regularization Parameter: ', num2str(lambda)));

    [U,Sigma,V] = svd(S,'econ');
    new_Sigma =  Sigma + (lambda./Sigma);
    c = V*((U'*u)./diag(new_Sigma));

    ima = reshape(c,40,40);
    ima = imresize(ima, [280 280]);
    ima(ima<0) = 0;

    figure;set(gcf, 'WindowState', 'maximized');
    subplot(1,3,1); imshow(phantom,[]);
    title('Reference Image');
    subplot(1,3,2); imshow(ima,[]);
    title(strcat('Filtered SVD ({\lambda} = ', num2str(lambda), ')'));
    subplot(1,3,3); imshow(abs(phantom-ima),[]);
    title('Error Image');
    %saveas(gcf,strcat('1.8_', num2str(lambda), '.png' ));

    %IQA measurements
    PSNR = psnr(ima,phantom);
    SSIM = ssim(ima,phantom);
    disp(strcat('PSNR for Regularization Parameter=',
num2str(lambda),':',num2str(PSNR)));
    disp(strcat('SSIM for Regularization Parameter=',
num2str(lambda),':',num2str(SSIM)));

end
```

Q1.9– L-Curve

Optimum regularization parameter is found to be 1700. The closest point to the left-corner gives a close optimum lambda value, but as explained before, ractically it is in the more right side of the closest point. Therefore, the previous result with my visual assessments was different than this result.

Results

PSNR for Regularization Parameter=1700:20.431

SSIM for Regularization Parameter=1700:0.65429

Plots

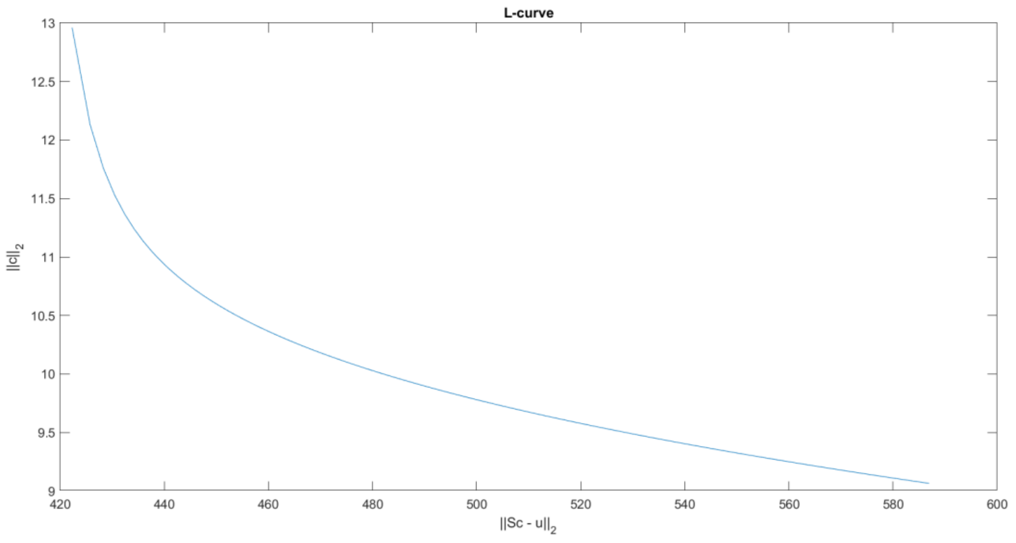


Figure 26. L-curve

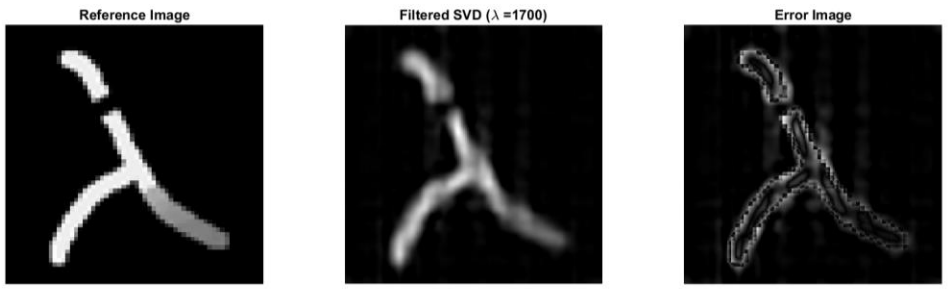


Figure 27. Images of Filtered SVD with Lambda = 1700

Code

```
disp('1.9');
disp("L-curve");

load("S.mat"); load("u.mat");
disp('1.9');
disp("L-curve");

load("S.mat"); load("u.mat");
load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');

[U,Sigma_ref,V] = svd(S,'econ');

lambda = 100:100:10000;
for i = 1:length(lambda)
    Sigma =  Sigma_ref + (lambda(i)./Sigma_ref);
    c = V*((U'*u)./diag(Sigma));
    residual_norm(i) = norm((S*c - u),2);
    solution_norm(i) = norm(c,2);
end
```

```
figure;set(gcf, 'WindowState', 'maximized');
plot(residual_norm,solution_norm);
title('L-curve');
xlabel('||Sc - u||_2'); ylabel('||c||_2');
saveas(gcf,'1.9_L-curve.png');

%find the optimum lambda
opt_lambda = lambda(max(find(residual_norm<=447.5)));
[U,Sigma,V] = svd(S,'econ');
new_Sigma =  Sigma + (opt_lambda./Sigma);
c = V*((U'*u)./diag(new_Sigma));

ima = reshape(c,40,40);
ima = imresize(ima, [280 280]);
ima(ima<0) = 0;

figure;set(gcf, 'WindowState', 'maximized');
subplot(1,3,1); imshow(phantom,[]);
title('Reference Image');
subplot(1,3,2); imshow(ima,[]);
title(strcat('Filtered SVD ({\lambda} = ', num2str(opt_lambda), ')'));
subplot(1,3,3); imshow(abs(phantom-ima),[]);
title('Error Image');
saveas(gcf,strcat('1.9.png' ));

%IQA measurements
PSNR = psnr(ima,phantom);
SSIM = ssim(ima,phantom);
disp(strcat('PSNR for Regularization Parameter=',
num2str(opt_lambda),':',num2str(PSNR)));
disp(strcat('SSIM for Regularization Parameter=',
num2str(opt_lambda),':',num2str(SSIM)));
```

PART 2 – KACZMARZ METHOD

Q2.1 - Row-norm Thresholding

Size of S is 763x1600 and size of u is 763x1.

Plots

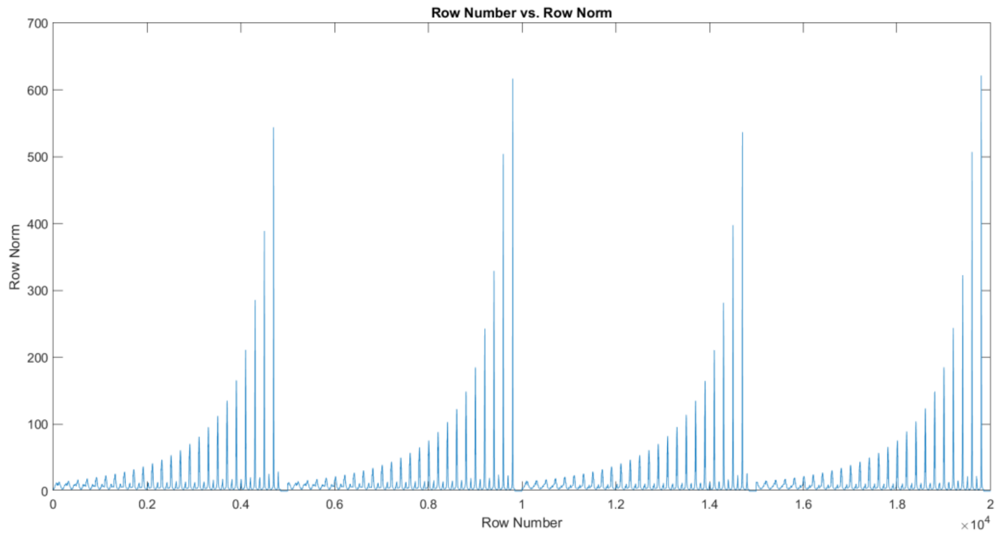


Figure 28. Row Number vs. Row Norm

Code

```
disp('2.1');
disp("Row-norm Thresholding");

load("S.mat"); load("u.mat");

treshold = 50;
[original_norm_S,norm_S,S,u] = row_norm_treshold(treshold,S,u);

figure;set(gcf, 'WindowState', 'maximized');
plot(original_norm_S);
title('Row Number vs. Row Norm');
xlabel('Row Number');ylabel('Row Norm');
saveas(gcf,'2.1.png');

size_S = size(S);
size_u = size(u);
disp(strcat('Size of S: ', num2str(size_S(1)), 'x',num2str(size_S(2)) ));
disp(strcat('Size of u: ', num2str(size_u(1)), 'x',num2str(size_u(2)) ));
```

row norm treshold function (only shown here)

```
function [original_norm_S,norm_S,S,u] = row_norm_treshold(treshold,S,u)

[row_no col_no] = size(S);
original_norm_S = zeros(1,row_no);
for i = 1:row_no
    original_norm_S(i) = norm(S(i,:),2);
end

index = find(original_norm_S<treshold);
S(index,:) = [];
u(index,:) = [];
```

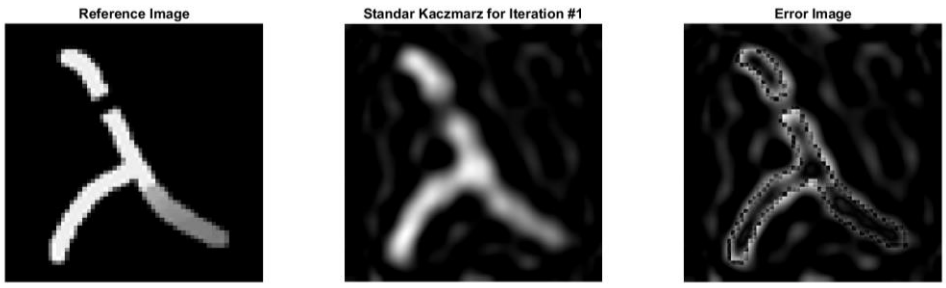
```
[row_no col_no] = size(S);
norm_S = zeros(1,row_no);
for i = 1:row_no
    norm_S(i) = norm(S(i,:),2);
end

end
```

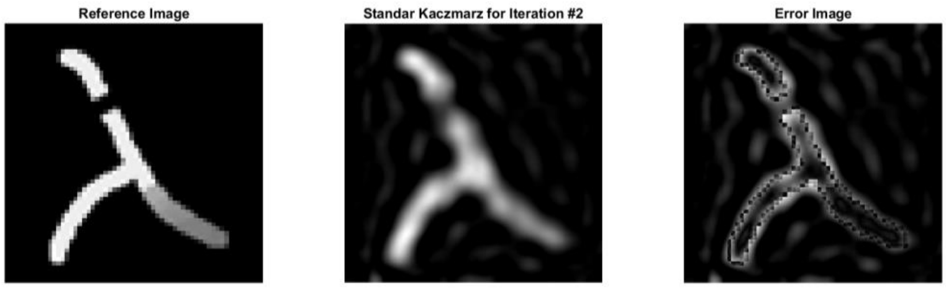
Q2.2 - . Standard Kaczmarz Method

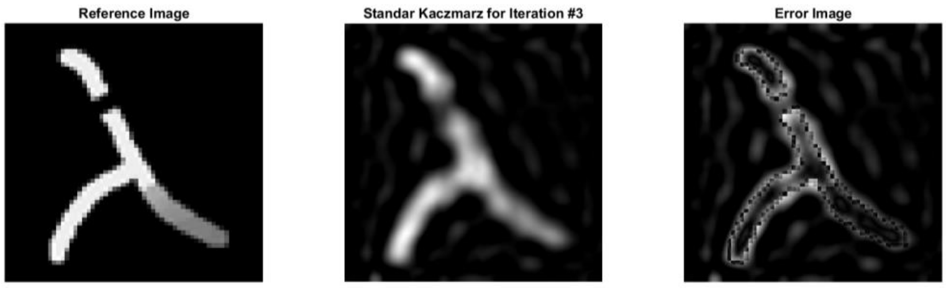
Results

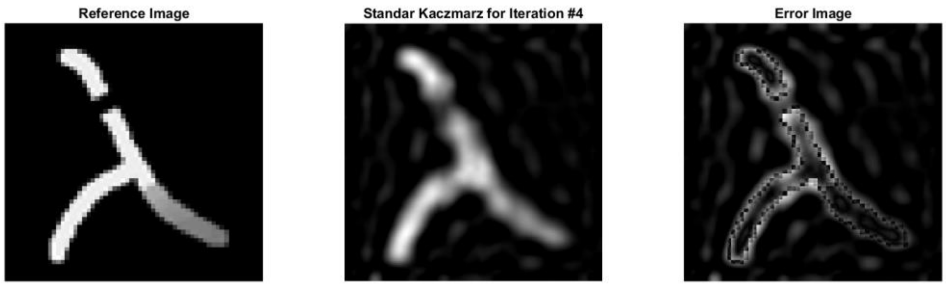
```
Standard Kaczmarz Method

PSNR for Iteration #1:18.4953

SSIM for Iteration #1:0.54247

PSNR for Iteration #2:18.7654

SSIM for Iteration #2:0.55691

PSNR for Iteration #3:18.7954

SSIM for Iteration #3:0.54704

PSNR for Iteration #4:18.7656

SSIM for Iteration #4:0.53522

PSNR for Iteration #5:18.7131

SSIM for Iteration #5:0.52601

PSNR for Iteration #6:18.6481

SSIM for Iteration #6:0.51824

PSNR for Iteration #7:18.5746

SSIM for Iteration #7:0.51157

PSNR for Iteration #8:18.495

SSIM for Iteration #8:0.50536

PSNR for Iteration #9:18.4106

SSIM for Iteration #9:0.49935

PSNR for Iteration #10:18.3223

SSIM for Iteration #10:0.49352
```

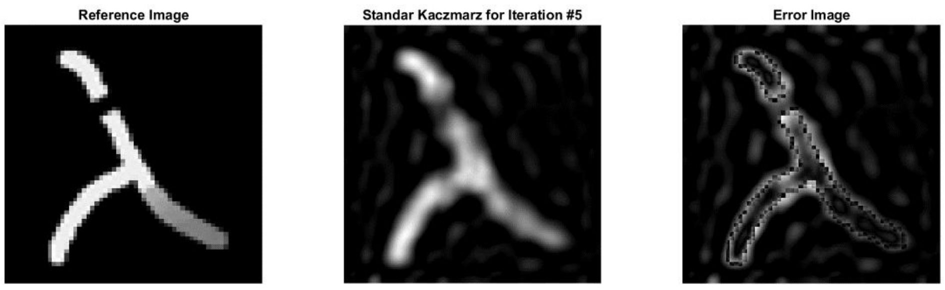
Plots

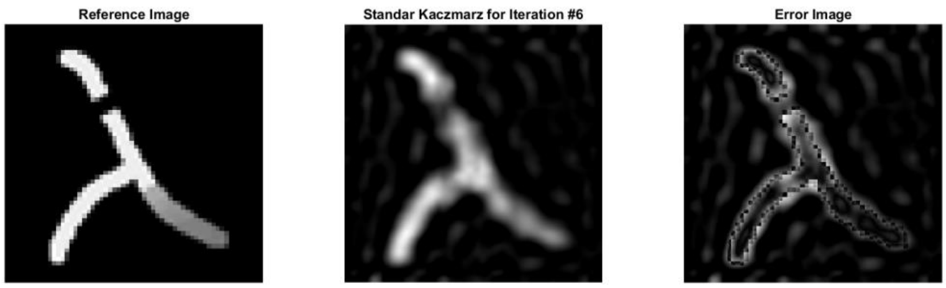

Reference Image
Standar Kaczmarz for Iteration #1
Error Image

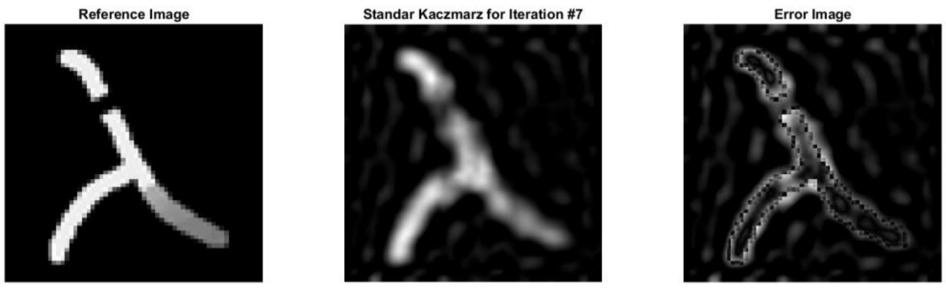

Reference Image
Standar Kaczmarz for Iteration #2
Error Image

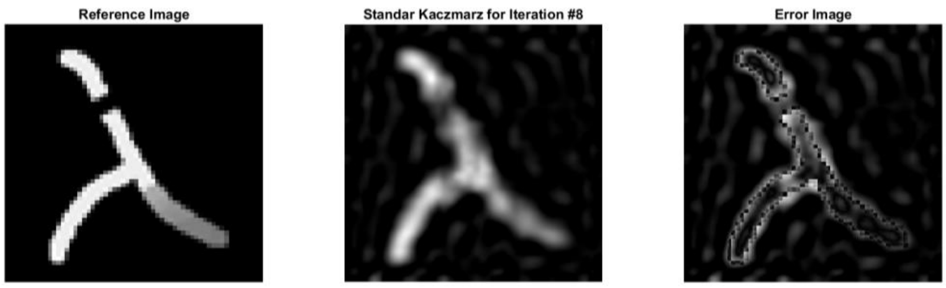

Reference Image
Standar Kaczmarz for Iteration #3
Error Image

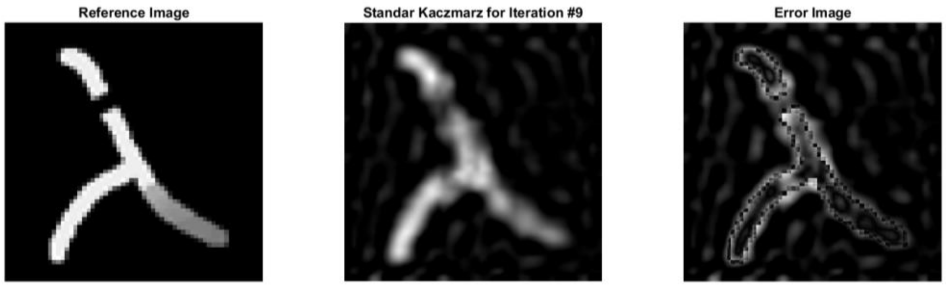

Reference Image
Standar Kaczmarz for Iteration #4
Error Image

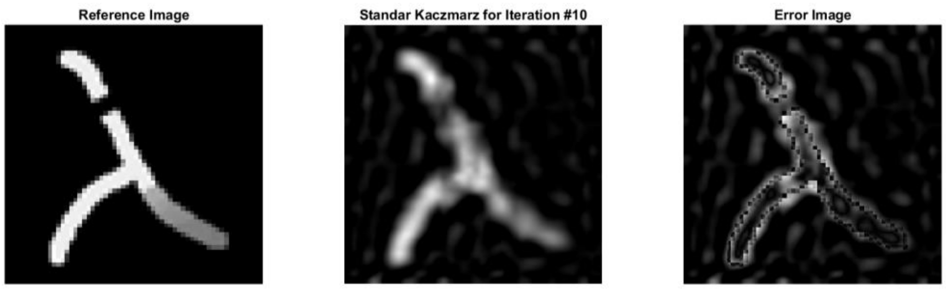

Reference Image
Standar Kaczmarz for Iteration #5
Error Image


Reference Image
Standar Kaczmarz for Iteration #6
Error Image


Reference Image
Standar Kaczmarz for Iteration #7
Error Image


Reference Image
Standar Kaczmarz for Iteration #8
Error Image


Reference Image
Standar Kaczmarz for Iteration #9
Error Image



Code

```
disp('2.2');
disp("Standard Kaczmarz Method");

load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');
load("S.mat"); load("u.mat");

treshold = 50;
[original_norm_S,norm_S,S,u] = row_norm_treshold(treshold,S,u);

%randomize rows proportional to ||si||^2
%take sorting_order as the order of sub-iterations
[sorted_norm_S,sorting_order] = sort((norm_S.^2),'descend');

[row_no col_no] = size(S);

%start from origin as first estimate solution
c = zeros(col_no,1);
for iter = 1:10
    for sub_iter = row_no:-1:1
        ind = find(sorting_order == sub_iter);
        si=S(ind,:);
        ui=u(ind);
        si_H = si';
        %When A and B are both
        %column vectors, dot(A,B)
        %is the same as A'*B.
        c = c + ((ui-dot(si,c))/(norm(si,2)^2))*si_H;
    end
    ima = reshape(c,40,40);
    ima = imresize(ima, [280 280]);
    ima(ima<0) = 0; %set negative values to zero

    figure;set(gcf, 'WindowState', 'maximized');
    subplot(1,3,1); imshow(phantom,[]);
    title('Reference Image');
    subplot(1,3,2); imshow(ima,[]);
    title(strcat('Standar Kaczmarz for Iteration #', num2str(iter)));
    subplot(1,3,3); imshow(abs(phantom-ima),[]);
    title('Error Image');
    saveas(gcf,strcat('2.2_iter', num2str( iter ), '.png' ));

    %IQA measurements
    PSNR = psnr(ima,phantom);
    SSIM = ssim(ima,phantom);
    disp(strcat('PSNR for Iteration #',num2str(iter),': ', num2str(PSNR)));
    disp(strcat('SSIM for Iteration #',num2str(iter),': ', num2str(SSIM)));

end
```

Q2.3 - Standard Kaczmarz Method with Different Treshold Values

When treshold is too small, we cannot get rid of noise. When trehsold is too large, similar to filtered SVD for different lambda values, result image looks blurry and low-pass filtered. Therefore, between values are giving better results. 30 and 50 are better treshold values in this question. 50 is the best in terms of PSNR, SSIM and visual assessments.

Results

Standard Kaczmarz Method with Different Tresholds

PSNR for Treshold=1:10.0821

SSIM for Treshold=1:0.16695

PSNR for Treshold=10:9.2985

SSIM for Treshold=10:0.1361

PSNR for Treshold=30:16.7844

SSIM for Treshold=30:0.40021

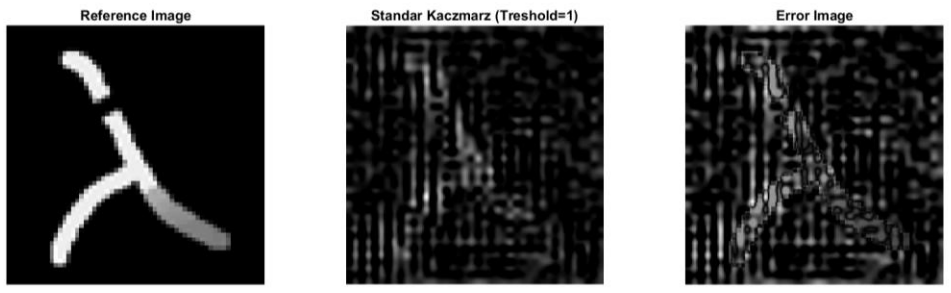
PSNR for Treshold=50:18.3223

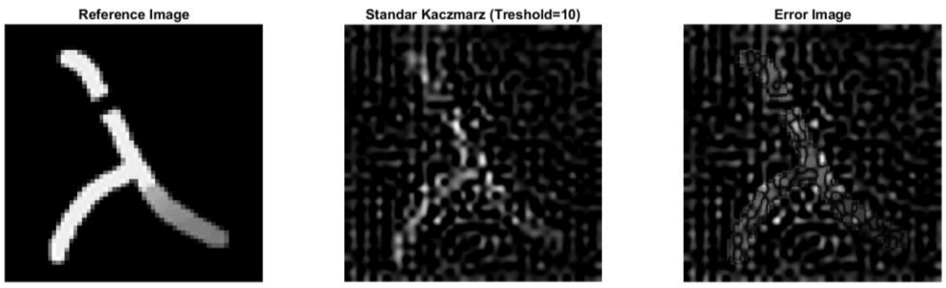
SSIM for Treshold=50:0.49352

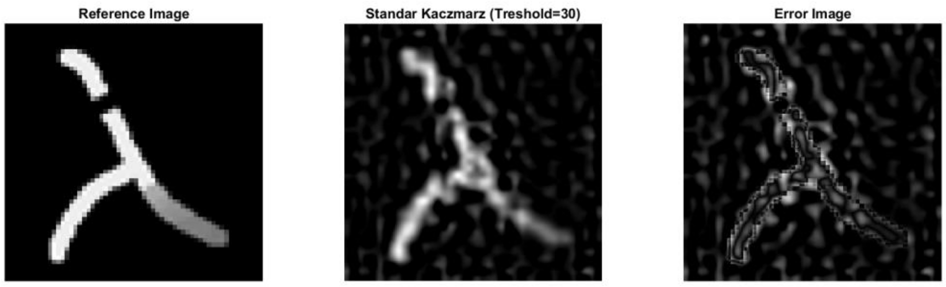
PSNR for Treshold=100:16.319

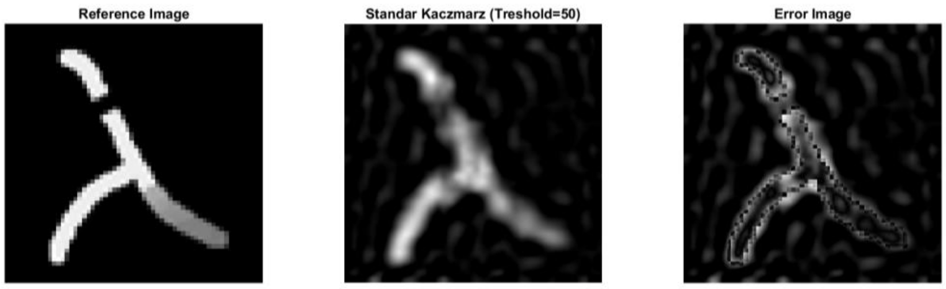
SSIM for Treshold=100:0.50745

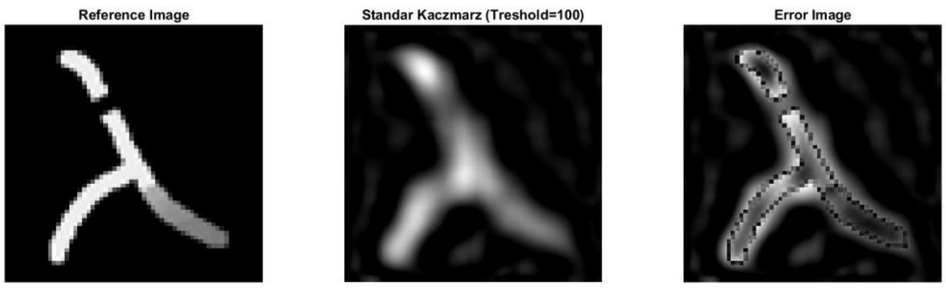
Plots











Code

```
disp('2.3');
disp("Standard Kaczmarz Method with Different Tresholds");

load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');
load("S.mat"); load("u.mat");
S_ref = S; u_ref = u;

for treshold = [1 10 30 50 100]

    %randomize rows proportional to ||si||^2
    %take sorting_order as the order of sub-iterations
    [original_norm_S,norm_S,S,u] = row_norm_treshold(treshold,S_ref,u_ref);
    [sorted_norm_S,sorting_order] = sort((norm_S.^2),'descend');

    [row_no col_no] = size(S);

    %start from origin as first estimate solution
    c = zeros(col_no,1);
    for iter = 1:10
        for sub_iter = row_no:-1:1
            ind = find(sorting_order == sub_iter);
            si=S(ind,:);
            ui=u(ind);
            si_H = si';
            c = c + ((ui-dot(si,c))/(norm(si,2)^2))*si_H;
        end
    end

    ima = reshape(c,40,40);
    ima = imresize(ima, [280 280]);
    ima(ima<0) = 0; %set negative values to zero

    figure;set(gcf, 'WindowState', 'maximized');
    subplot(1,3,1); imshow(phantom,[]);
    title('Reference Image');
    subplot(1,3,2); imshow(ima,[]);
    title(strcat('Standar Kaczmarz (Treshold=', num2str(treshold), ')'));
    subplot(1,3,3); imshow(abs(phantom-ima),[]);
    title('Error Image');
    saveas(gcf,strcat('2.3_treshold', num2str( treshold ), '.png' ));

    %IQA measurements
    PSNR = psnr(ima,phantom);
    SSIM = ssim(ima,phantom);
    disp(strcat('PSNR for Treshold= ',num2str(treshold), ': ',
num2str(PSNR)));
    disp(strcat('SSIM for Treshold= ',num2str(treshold), ': ',
num2str(SSIM)));

end
```

Q2.4 - Regularized Kaczmarz Method

In previous question, result with treshold of 1 was very noisy and the image could be barely understood. With regularized Kaczmarz method, we reach to a closer result image to reference image even with first iteration. Still, the last image is not as desired because of low tresholding. PSNR and SSIM values are shown in results.

Results

Regularized Kaczmarz Method

PSNR for Iteration #1:11.1732

SSIM for Iteration #1:0.81774

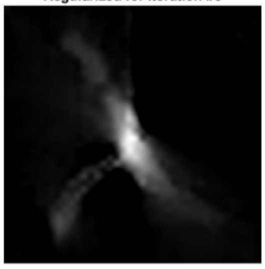
PSNR for Iteration #2:11.171

```
SSIM for Iteration #2:0.81824
PSNR for Iteration #3:11.1711
SSIM for Iteration #3:0.81823
PSNR for Iteration #4:11.171
SSIM for Iteration #4:0.81823
PSNR for Iteration #5:11.171
SSIM for Iteration #5:0.81823
PSNR for Iteration #6:11.171
SSIM for Iteration #6:0.81823
PSNR for Iteration #7:11.171
SSIM for Iteration #7:0.81823
PSNR for Iteration #8:11.171
SSIM for Iteration #8:0.81823
PSNR for Iteration #9:11.171
SSIM for Iteration #9:0.81823
PSNR for Iteration #10:11.171
SSIM for Iteration #10:0.81823
```

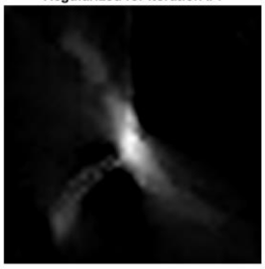
Plots

Reference Image
Regularized for Iteration #1
Error Image

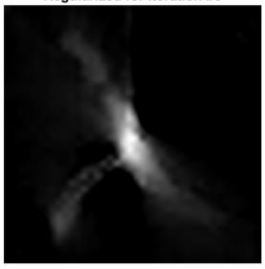
Reference Image
Regularized for Iteration #2
Error Image

Reference Image

Regularized for Iteration #3

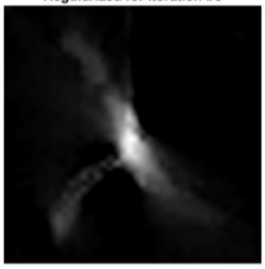
Error Image

Reference Image

Regularized for Iteration #4

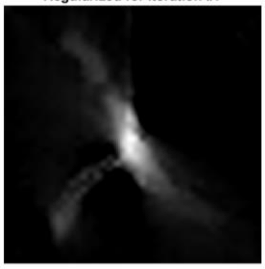
Error Image

Reference Image

Regularized for Iteration #5

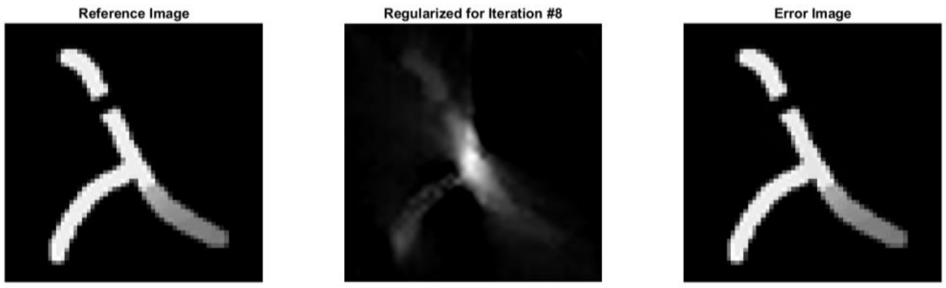
Error Image

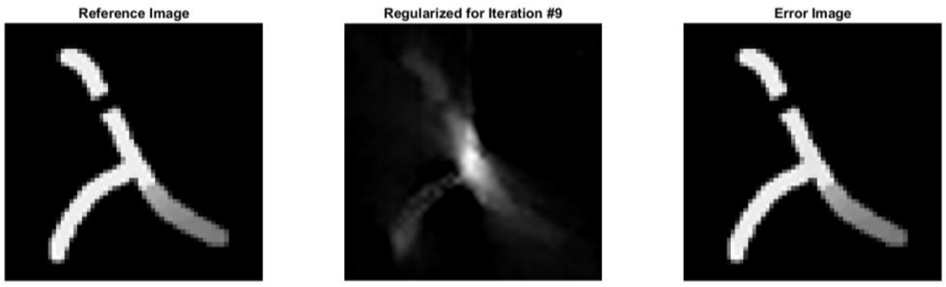
Reference Image

Regularized for Iteration #6

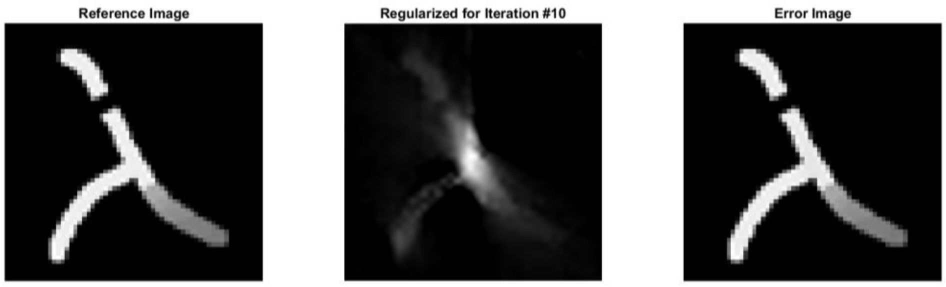
Error Image

Reference Image

Regularized for Iteration #7

Error Image







Code

```
disp('2.4');
disp("Regularized Kaczmarz Method");

load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');
load("S.mat"); load("u.mat");

treshold = 1;
[original_norm_S,norm_S,S,u] = row_norm_treshold(treshold,S,u);
%randomize rows proportional to ||si||^2
[sorted_norm_S,sorting_order] = sort((norm_S.^2),'descend');

%weighting function
w = 1./(norm_S.^2);
W = diag(w);

[U,Sigma,V] = svd(S,'econ');
singular_values = diag(Sigma);
lambda = singular_values(1)* singular_values(end);

[row_no, col_no] = size(S);
identity = eye(row_no);
```

```
%start from origin for first estimate solution
c = zeros(col_no,1);
v = zeros(row_no,1);
for iter = 1:10
    for sub_iter = row_no:-1:1
        ind = find(sorting_order == sub_iter);%find index
        %parameters
        si=S(ind,:);
        ui=u(ind);
        si_H = si';
        wi = w(ind);
        ei = identity(:,ind);
        vi = v(ind);
        %algorithm
        a = ( ui - dot(si_H,c) - sqrt(lambda/wi)*vi )./( norm(si,2)^2 +
(lambda/wi) );
        c = c + a*si_H;
        v = v + a*sqrt(lambda/wi)*ei;
    end

    ima = reshape(c,40,40);
    ima = imresize(ima, [280 280]);
    ima(ima<0) = 0; %set negative values to zero

    figure;set(gcf, 'WindowState', 'maximized');
    subplot(1,3,1); imshow(phantom,[]);
    title('Reference Image');
    subplot(1,3,2); imshow(ima,[]);
    title(strcat('Regularized for Iteration #', num2str(iter)));
    subplot(1,3,3); imshow(abs(phantom-ima),[]);
    title('Error Image');
    saveas(gcf,strcat('2.4_iter', num2str( iter ), '.png' ));

    %IQA measurements
    PSNR = psnr(ima,phantom);
    SSIM = ssim(ima,phantom);
    disp(strcat('PSNR for Iteration #',num2str(iter),': ', num2str(PSNR)));
    disp(strcat('SSIM for Iteration #',num2str(iter),': ', num2str(SSIM)));

end
```

Q2.5 - Regularized Kaczmarz Method with Diffferent Tresholds

Best visuals results are with lambda = 127.6172 and lambda = 1276.1729. SSIM is better for 1276.1729, therefore I chose 1276.1729 to be optimum lambda value for next question.

I was expecting to see better results, since I mainly got the central part of result image and the rest is not visible. I could not understand its reason. The reason is low-tresholding. Later, when we have better tresholds like 50, we will see the improvement in our result images.

Results

Regularized Kaczmarz Method with Different Lambdas

PSNR for Lambda=127.6173:12.1806

SSIM for Lambda=127.6173:0.64081

PSNR for Lambda=1276.1729:11.2121

SSIM for Lambda=1276.1729:0.81147

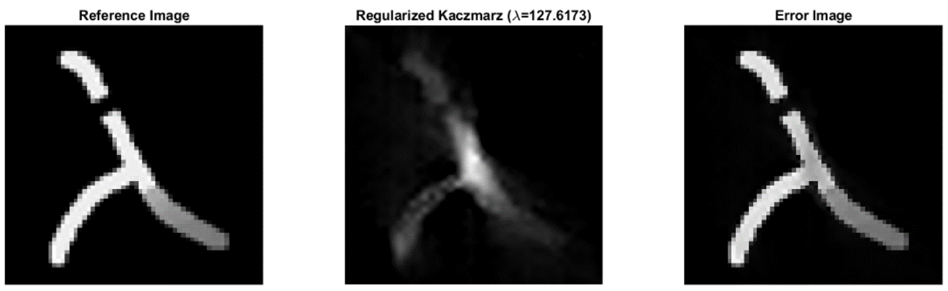
PSNR for Lambda=12761.7286:11.073

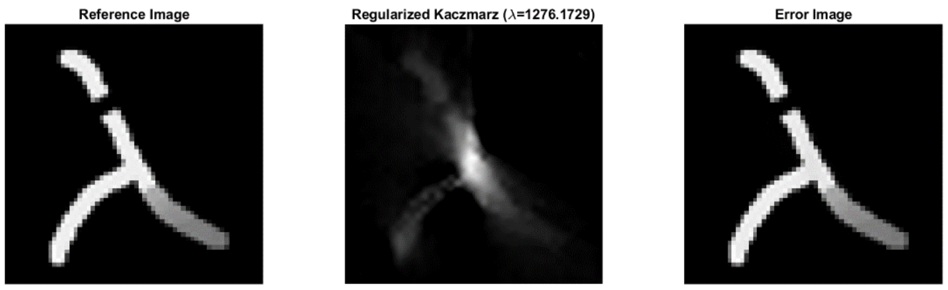
SSIM for Lambda=12761.7286:0.82616

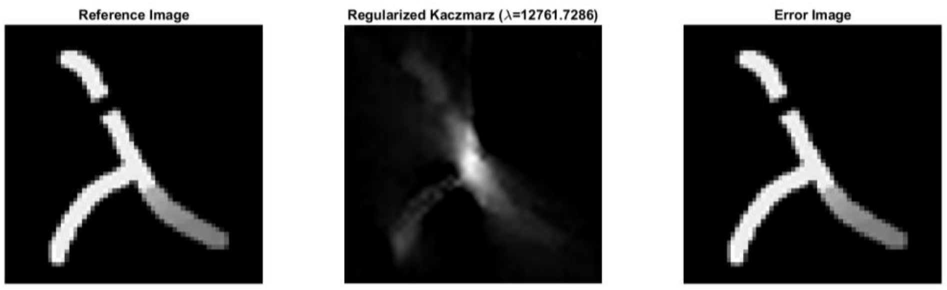
PSNR for Lambda=127617.286:11.0583

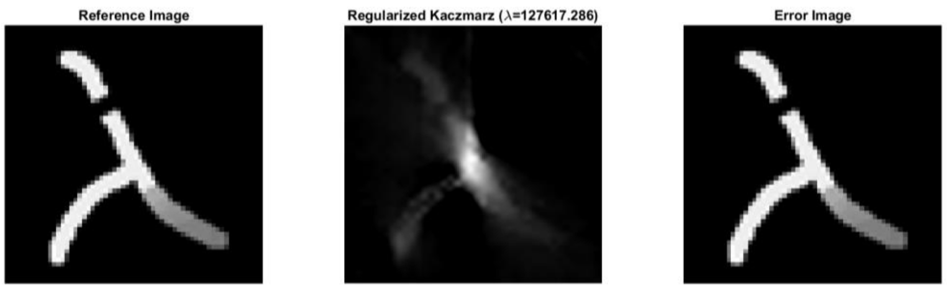
SSIM for Lambda=127617.286:0.82577

Plots









Code

```
disp('2.5');
disp("Regularized Kaczmarz Method with Different Lambdas");

load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');
load("S.mat"); load("u.mat");
S_ref = S; u_ref = u;

treshold = 1;
[original_norm_S,norm_S,S,u] = row_norm_treshold(treshold,S,u);
%randomize rows proportional to ||si||^2
[sorted_norm_S,sorting_order] = sort((norm_S.^2),'descend');

%weighting function
w = 1./(norm_S.^2);
W = diag(w);

[U,Sigma,V] = svd(S,'econ');
singular_values = diag(Sigma);
lambda_rel = [1e-4 1e-3 1e-2 1e-1];

[row_no, col_no] = size(S);
identity = eye(row_no);


for lambda = (singular_values(1)^2).*lambda_rel

    %randomize rows proportional to ||si||^2
    [original_norm_S,norm_S,S,u] = row_norm_treshold(treshold,S_ref,u_ref);
    [sorted_norm_S,sorting_order] = sort((norm_S.^2),'descend');
    [row_no col_no] = size(S);
    %start from origin for first estimate solution
    c = zeros(col_no,1);
    v = zeros(row_no,1);
    for iter = 1:10
        for sub_iter = row_no:-1:1
            ind = find(sorting_order == sub_iter);%find index
            %parameters
            si=S(ind,:);
            ui=u(ind);
            si_H = si';
            wi = w(ind);
            ei = identity(:,ind);
            vi = v(ind);
            %algorithm
            a = ( ui - dot(transpose(si_H),c) - sqrt(lambda/wi)*vi )./(
norm(si,2)^2 + (lambda/wi) );
            c = c + a*si_H;
            v = v + a*sqrt(lambda/wi)*ei;
        end

    end

    ima = reshape(c,40,40);
    ima = imresize(ima, [280 280]);
    ima(ima<0) = 0; %set negative values to zero

    figure;set(gcf, 'WindowState', 'maximized');
    subplot(1,3,1); imshow(phantom,[]);
    title('Reference Image');
    subplot(1,3,2); imshow(ima,[]);
    title(strcat('Regularized Kaczmarz ({\lambda}=',num2str(lambda),')'));
    subplot(1,3,3); imshow(abs(phantom-ima),[]);
    title('Error Image');
    saveas(gcf,strcat('2.5_lambda', num2str( lambda ), '.png' ));

    %IQA measurements
    PSNR = psnr(ima,phantom);
    SSIM = ssim(ima,phantom);
    disp(strcat('PSNR for Lambda=',num2str(lambda),': ', num2str(PSNR)));
    disp(strcat('SSIM for Lambda=',num2str(lambda),': ', num2str(SSIM)));

end
```

Q2.6 - Regularized Kaczmarz Method with Diffferent Tresholds

The best result comes with threshold value of 30. PSNR and SSIM values are very close to each other, therefore the best image is chosen visually. When threshold is low, we lose information about our image. When threshold is large, we start to discard desired high frequencies. Hence, we lose resolution.

There is a trade-off noise and resolution here. When threshold is low, we have noise. When threshold is high, we lose resolution and image becomes blurry and smooth.

Results

2.6

Regularized Kaczmarz Method with Different Tresholds for the Best Lambda

PSNR for Treshold=1:11.2121

SSIM for Treshold=1:0.81147

PSNR for Treshold=10:11.1132

SSIM for Treshold=10:0.82573

PSNR for Treshold=30:11.0904

SSIM for Treshold=30:0.82648

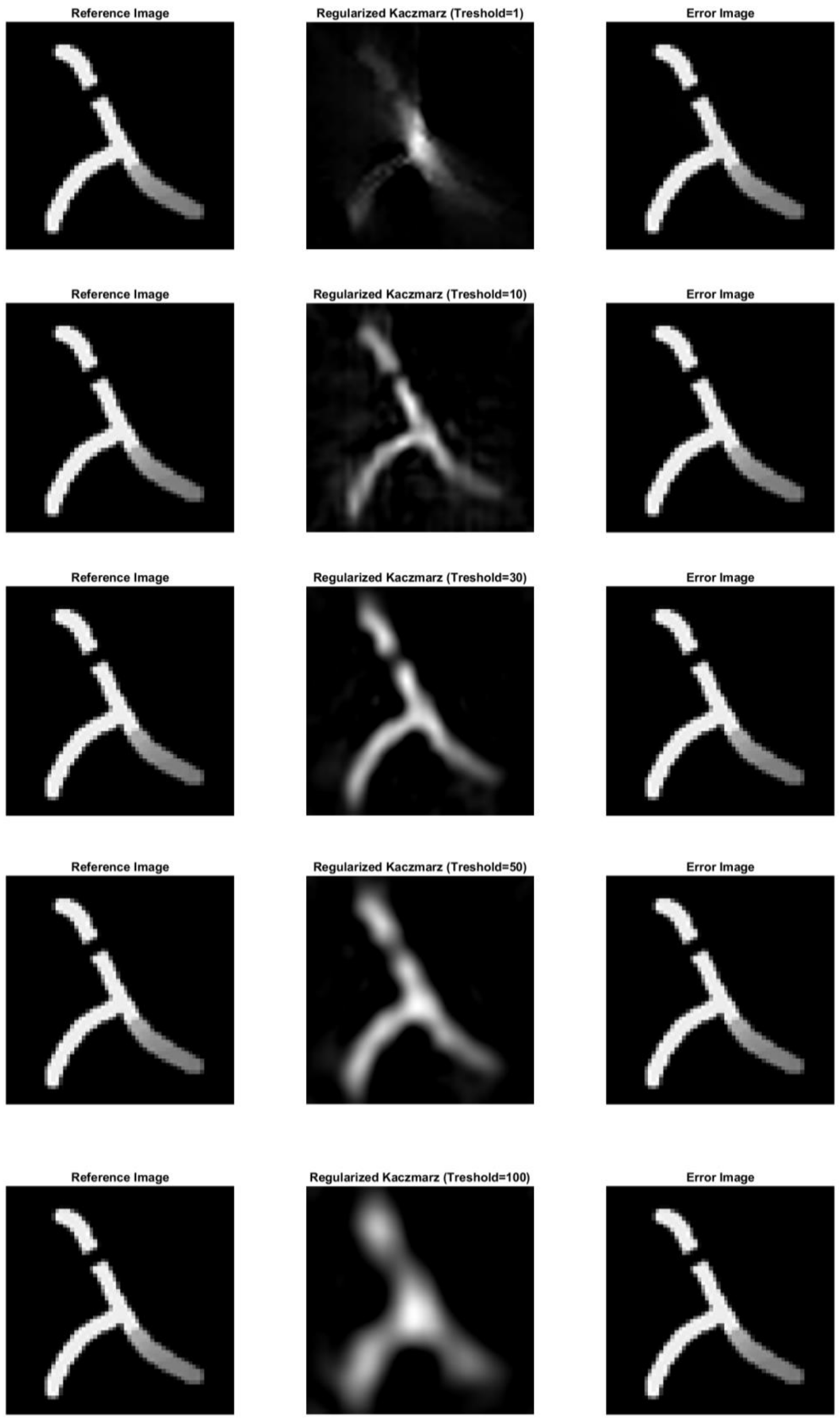
PSNR for Treshold=50:11.0842

SSIM for Treshold=50:0.82668

PSNR for Treshold=100:11.0756

SSIM for Treshold=100:0.8263

Plots


Reference Image
Regularized Kaczmarz (Treshold=1)
Error Image
Reference Image
Regularized Kaczmarz (Treshold=10)
Error Image
Reference Image
Regularized Kaczmarz (Treshold=30)
Error Image
Reference Image
Regularized Kaczmarz (Treshold=50)
Error Image
Reference Image
Regularized Kaczmarz (Treshold=100)
Error Image

Code

```
disp('2.6');
disp("Regularized Kaczmarz Method with Different Tresholds for the Best
Lambda");

load("measurement_phantom.mat",'meas_coil1','meas_coil2','phantom');

load("S.mat");load("u.mat");
S_ref = S;u_ref = u;

lambda = 1276.1729;

for treshold = [1 10 30 50 100]

    %randomize rows proportional to ||si||^2
    [original_norm_S,norm_S,S,u] = row_norm_treshold(treshold,S_ref,u_ref);
    [sorted_norm_S,sorting_order] = sort((norm_S.^2),'descend');

    [U,Sigma,V] = svd(S,'econ');

    %weighting function
    w = 1./(norm_S.^2);
    W = diag(w);

    [row_no col_no] = size(S);
    identity = eye(row_no);
    %start from origin for first estimate solution
    c = zeros(col_no,1);
    v = zeros(row_no,1);
    for iter = 1:10
        for sub_iter = row_no:-1:1
            ind = find(sorting_order == sub_iter);%find index
            %parameters
            si=S(ind,:);
            ui=u(ind);
            si_H = si';
            wi = w(ind);
            ei = identity(:,ind);
            vi = v(ind);
            %algorithm
            a = ( ui - dot(transpose(si_H),c) - sqrt(lambda/wi)*vi )./(
norm(si,2)^2 + (lambda/wi) );
            c = c + a*si_H;
            v = v + a*sqrt(lambda/wi)*ei;
        end
    end

    ima = reshape(c,40,40);
    ima = imresize(ima, [280 280]);
    ima(ima<0) = 0; %set negative values to zero

    figure;set(gcf, 'WindowState', 'maximized');
    subplot(1,3,1); imshow(phantom,[]);
    title('Reference Image');
    subplot(1,3,2); imshow(ima,[]);
    title(strcat('Regularized Kaczmarz (Treshold=',num2str(treshold),')'));
    subplot(1,3,3); imshow(abs(phantom-ima),[]);
    title('Error Image');
    saveas(gcf,strcat('2.6_treshold', num2str( treshold ), '.png' ));

    %IQA measurements
    PSNR = psnr(ima,phantom);
    SSIM = ssim(ima,phantom);
    disp(strcat('PSNR for Treshold=',num2str(treshold),': ',
num2str(PSNR)));
    disp(strcat('SSIM for Treshold=',num2str(treshold),': ',
num2str(SSIM)));

end

case '2.7'

disp('2.7');
disp("Effects of Measurement Noise");

end
```

PART 3 – OPEN MPI DATA

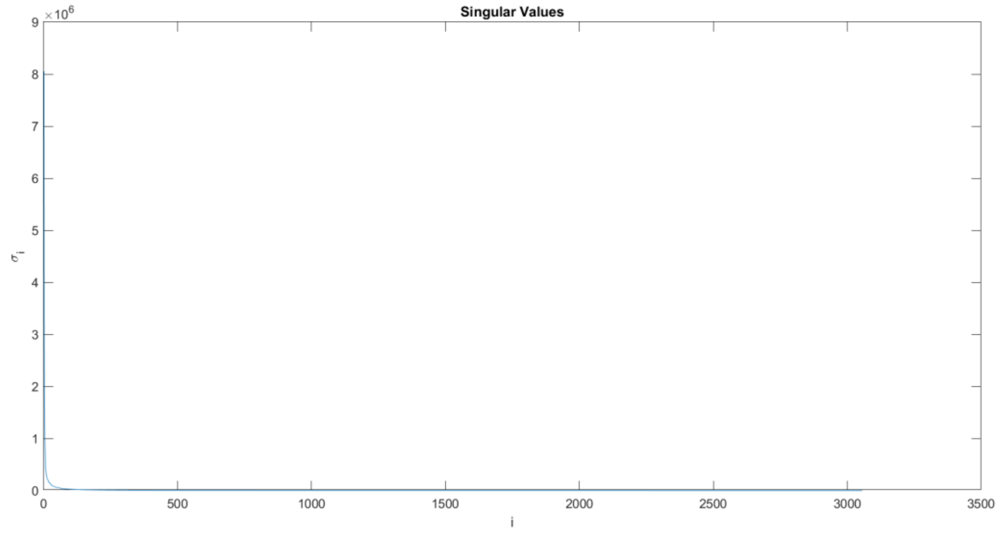
Q3.1

Condition number is a lot higher than in Part 2. Our system will be unstable for noise.

Results

Condition Number:106761830400

Plots



Code

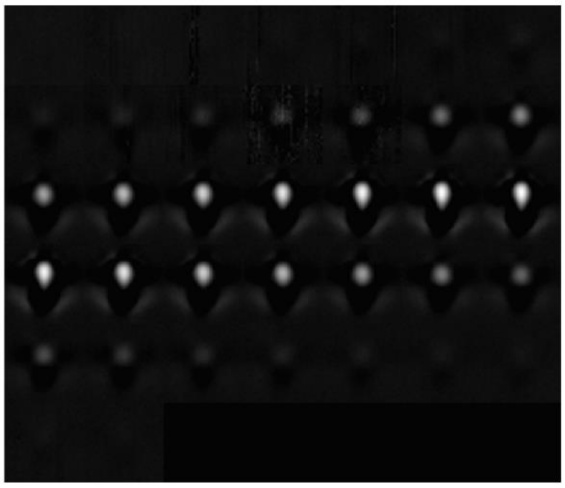
```
disp('3.1');
load("su_openmpi.mat",'S','u');

[U,Sigma,V] = svd(S,'econ');
%plot the singular values
singular_values = diag(Sigma);
figure;set(gcf, 'WindowState', 'maximized');
```

```
plot(singular_values); title('Singular Values');
xlabel('i'); ylabel('{\sigma}_i');
saveas(gcf,'3.1_singular_values.png');

condition_no = singular_values(1)/singular_values(end);
disp(strcat('Condition Number: ', num2str(condition_no)));
```

Q3.2

We see the effect of high condition number as a background noise. The system is unstable for noise. We should have high regularization parameter to compensate this instability.

Plots



Code

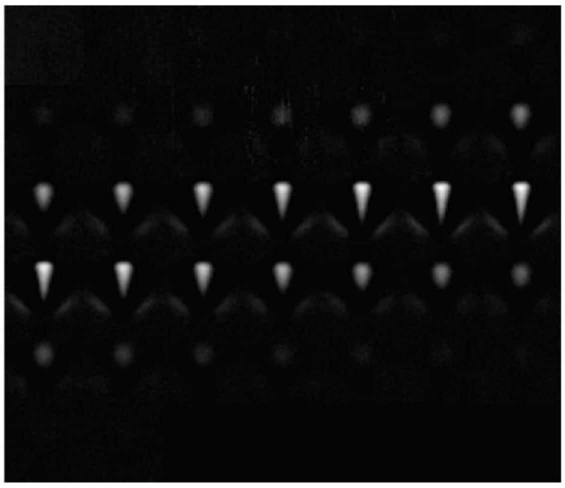
```
disp('3.2');
load("su_openmpi.mat",'S','u');

treshold = 30;
[original_norm_S,norm_S,S,u] = row_norm_treshold(treshold,S,u);
%randomize rows proportional to ||si||^2
[sorted_norm_S,sorting_order] = sort((norm_S.^2),'descend');

%weighting function
w = 1./(norm_S.^2);
W = diag(w);

[U,Sigma,V] = svd(S,'econ');
singular_values = diag(Sigma);
lambda = singular_values(1)* singular_values(end);

[row_no, col_no] = size(S);
identity = eye(row_no);

%start from origin for first estimate solution
c = zeros(col_no,1);
v = zeros(row_no,1);
```

```
for iter = 1:10
    for sub_iter = row_no:-1:1
        ind = find(sorting_order == sub_iter);%find index
        %parameters
        si=S(ind,:);
        ui=u(ind);
        si_H = si';
        wi = w(ind);
        ei = identity(:,ind);
        vi = v(ind);
        %algorithm
        a = ( ui - dot(si_H,c) - sqrt(lambda/wi)*vi )./( norm(si,2)^2 +
(lambda/wi) );
        c = c + a*si_H;
        v = v + a*sqrt(lambda/wi)*ei;
    end
end

ima = reshape(c,37,37,37);
ima(ima<0) = 0; %set negative values to zero
figure;set(gcf, 'WindowState', 'maximized');
montage(reshape(ima,[37, 37, 1, 37]),'displayRange',[]);
saveas(gcf,'3.2.png' );
```

Q3.3

When lambda is 1, we have blurring in our image due to instability of system matrix.

Plots



Code

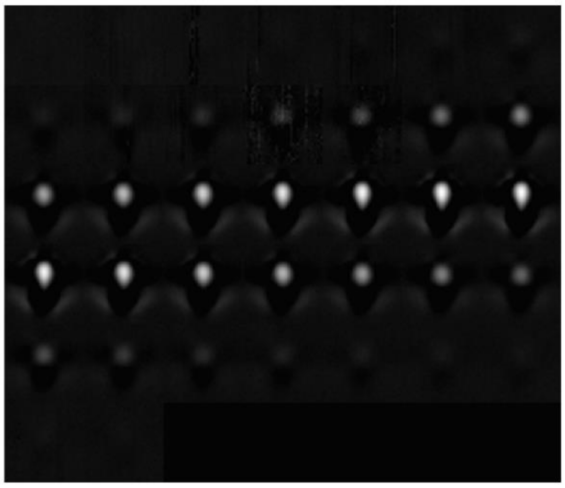
```
disp('3.3');
load("su_openmpi.mat",'S','u');

treshold = 30;
[original_norm_S,norm_S,S,u] = row_norm_treshold(treshold,S,u);
%randomize rows proportional to ||si||^2
```

```
[sorted_norm_S,sorting_order] = sort((norm_S.^2),'descend');

%weighting function
w = 1./(norm_S.^2);
W = diag(w);

[U,Sigma,V] = svd(S,'econ');
singular_values = diag(Sigma);
lambda = 1;

[row_no, col_no] = size(S);
identity = eye(row_no);

%start from origin for first estimate solution
c = zeros(col_no,1);
v = zeros(row_no,1);
for iter = 1:10
    for sub_iter = row_no:-1:1
        ind = find(sorting_order == sub_iter);%find index
        %parameters
        si=S(ind,:);
        ui=u(ind);
        si_H = si';
        wi = w(ind);
        ei = identity(:,ind);
        vi = v(ind);
        %algorithm
        a = ( ui - dot(si_H,c) - sqrt(lambda/wi)*vi )./( norm(si,2)^2 +
(lambda/wi) );
        c = c + a*si_H;
        v = v + a*sqrt(lambda/wi)*ei;
    end
end

ima = reshape(c,37,37,37);
ima(ima<0) = 0; %set negative values to zero
figure;set(gcf, 'WindowState', 'maximized');
montage(reshape(ima,[37, 37, 1, 37]),'displayRange',[]);
saveas(gcf,'3.3.png' );
```

Q3.4

We have better images than Q3.2 and Q3.3. We need high lambda values to compensate instability of system matrix. However, it should not be very high as explained in Part 2. Because we have blurry, smooth images with less resolution when we have very high regularization parameter.

Plots



Code

```
disp('3.4');
load("su_openmpi.mat",'S','u');

treshold = 30;
[original_norm_S,norm_S,S,u] = row_norm_treshold(treshold,S,u);
%randomize rows proportional to ||si||^2
[sorted_norm_S,sorting_order] = sort((norm_S.^2),'descend');

%weighting function
w = 1./(norm_S.^2);
W = diag(w);

[U,Sigma,V] = svd(S,'econ');
singular_values = diag(Sigma);
lambda = 1e10;

[row_no, col_no] = size(S);
identity = eye(row_no);

%start from origin for first estimate solution
c = zeros(col_no,1);
v = zeros(row_no,1);
for iter = 1:10
    for sub_iter = row_no:-1:1
        ind = find(sorting_order == sub_iter);%find index
        %parameters
        si=S(ind,:);
        ui=u(ind);
        si_H = si';
        wi = w(ind);
        ei = identity(:,ind);
        vi = v(ind);
        %algorithm
        a = ( ui - dot(si_H,c) - sqrt(lambda/wi)*vi )./( norm(si,2)^2 +
(lambda/wi) );
        c = c + a*si_H;
        v = v + a*sqrt(lambda/wi)*ei;
    end
end

ima = reshape(c,37,37,37);
ima(ima<0) = 0; %set negative values to zero
figure;set(gcf, 'WindowState', 'maximized');
montage(reshape(ima,[37, 37, 1, 37]),'displayRange',[]);
saveas(gcf,'3.4.png' );
```