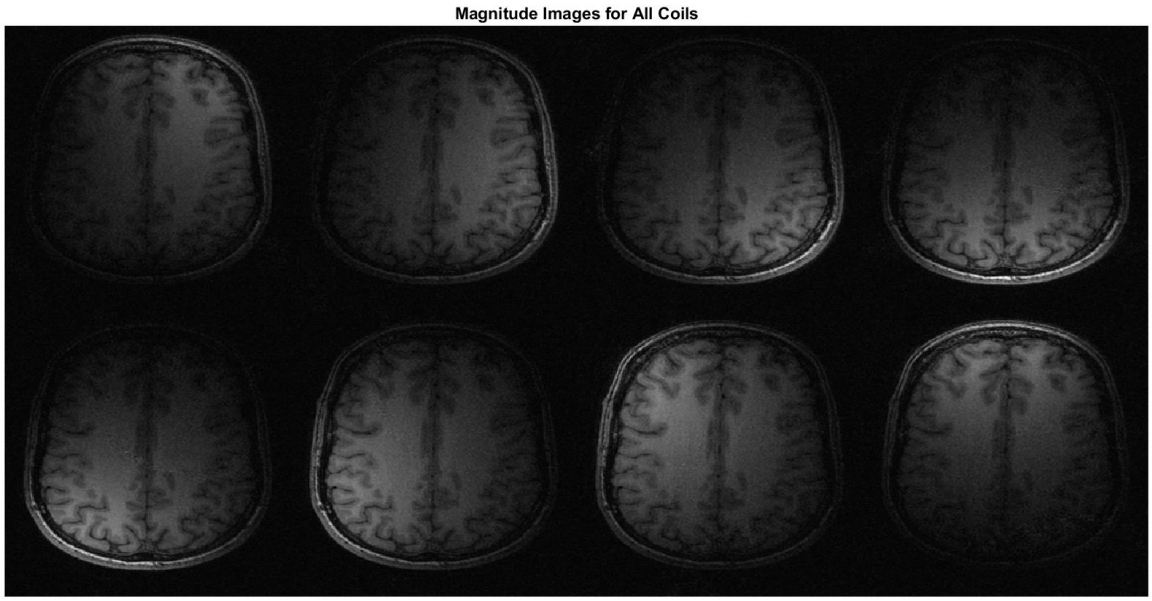
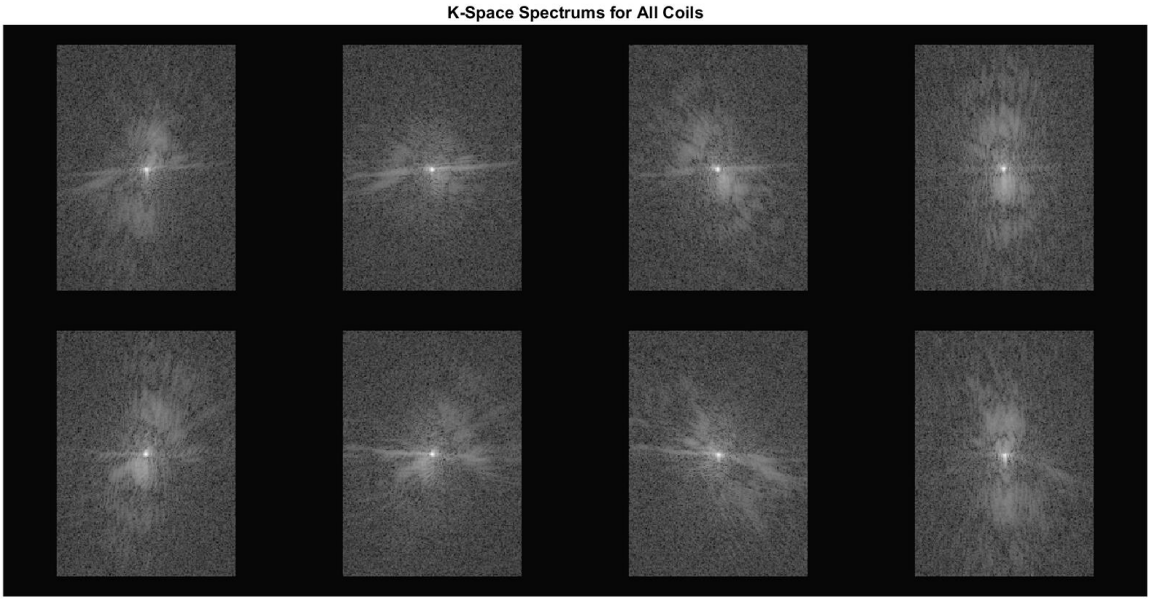
Berfin Kavşut
21602459
EEE475
May 25, 2020

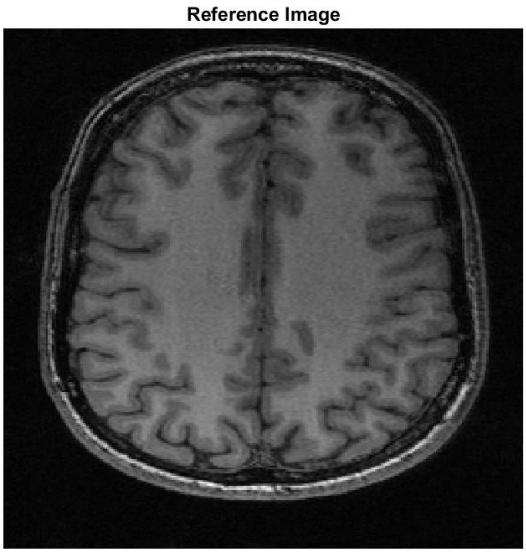
REPORT FOR HOMEWORK #5

Q1 – Display the Data

Images & Plots







MATLAB Code

```
disp('1.1')
disp('Display the Data');

load('multicoil-random.mat');

%take 2DFT of image
for coil= 1:8
    d(:,:,coil) = fft2c(im(:,:,coil));
end

%SoS reconstruction
m_sos = sos(im);
ref = m_sos;
%save reference image
save('reference.mat','ref');

%magnitude images for all coils
figure;set(gcf, 'WindowState', 'maximized');
montage(abs(im),'DisplayRange', [], 'Size', [2 4]);
title('Magnitude Images for All Coils');
saveas(gcf,'1.1_mag.png');

%k-space spectrums for all coils
figure; set(gcf, 'WindowState', 'maximized');
montage(log(abs(d)+1),'DisplayRange', [], 'Size', [2 4]);
title('K-Space Spectrums for All Coils');
saveas(gcf,'1.1_kspace.png');

%magnitude image for SoS
figure;set(gcf, 'WindowState', 'maximized');
imshow(abs(m_sos),[]);
title('Reference Image');
saveas(gcf,'1.1_sos.png');
```

m = sos(im) function (only written here)

```
function  m = sos(im)

% sum of squares (sos) function
% im: images from all coils
% map: coil sensitivities
% m: result image

%map is ideally uniform over the entire object
map = ones(size(im));
%find mss from coil images
mss = sqrt( sum( im.*conj(im), 3) );
%find weights from coil sensitivities
coil_sens= sqrt( sum( abs(map).^2,3 ) );
%find image from mss and calculated weights
m = mss./coil_sens;

%correct image by setting inf and nan values to zero
m(isinf(m)) = 0;
m(isnan(m)) = 0;

end
```

Q2 - Sampling Mask and Random Undersampled Images

When we have uniformly undersampled images, which means we acquire our k-space data by skipping lines, artifacts due to undersampling appear as coherent aliased images. Artifacts appear as repating with respect to Nyquist theorem. Here, since we have randomly undersampled images, artifacts are noise-like artifacts, which means they are incoherent artiacts.

Results

1.2

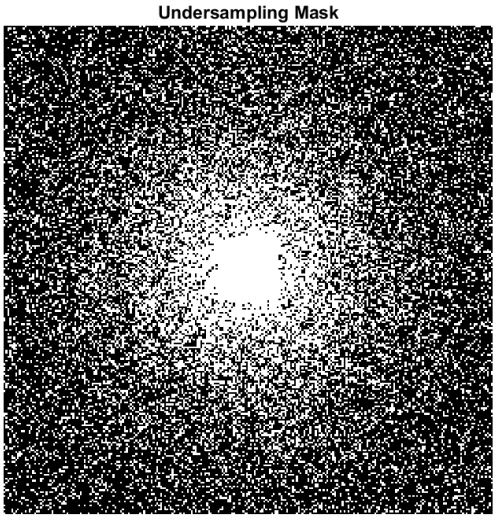
Sampling Mask and Random Undersampled Images

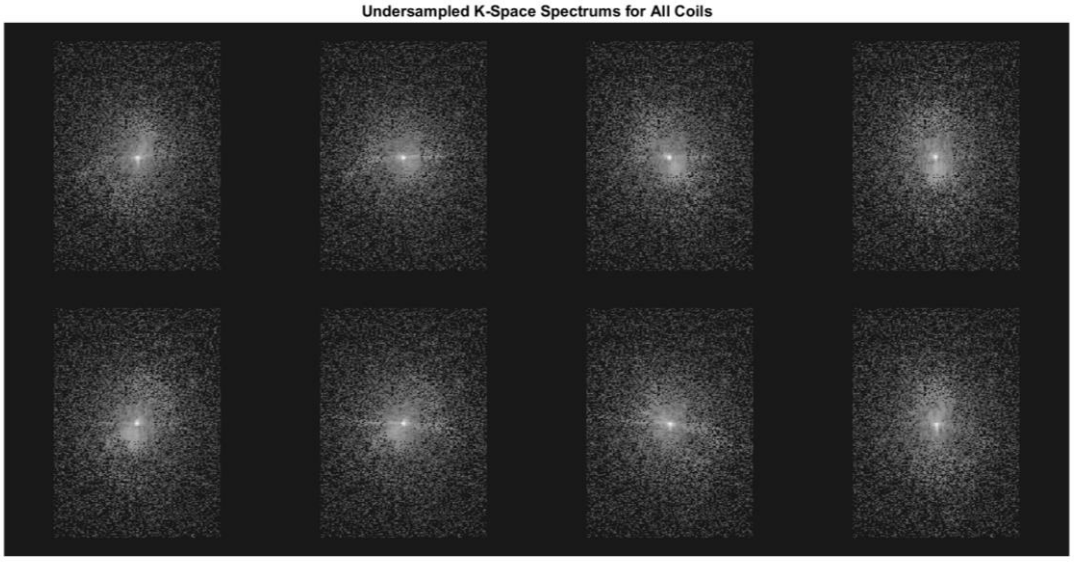
The overall acceleration corresponding to the given mask:3.03

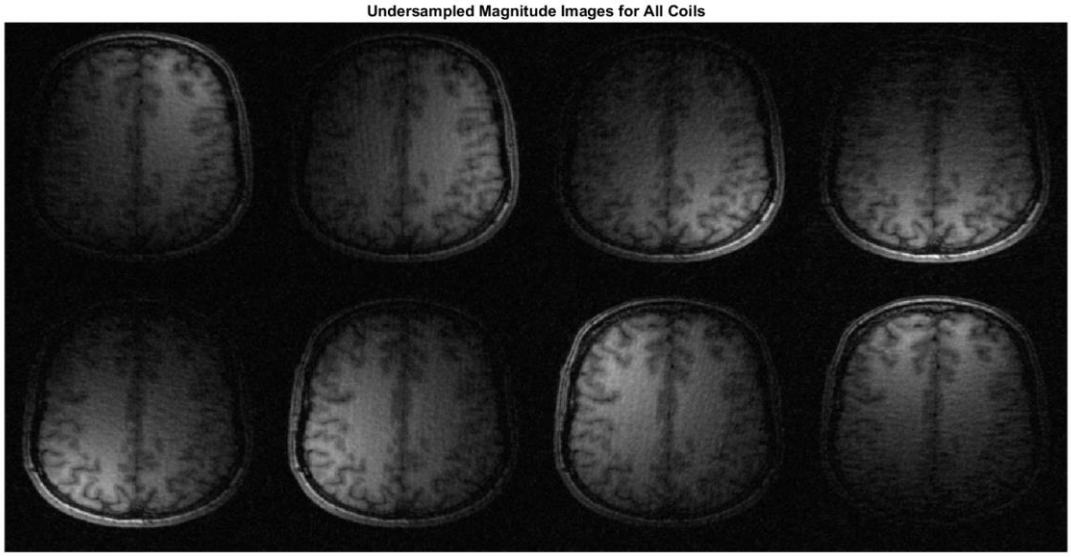
PSNR:22.4526

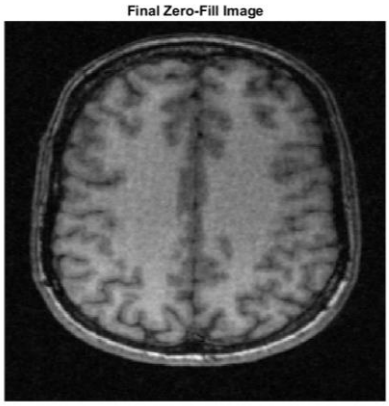
SSIM:0.83054

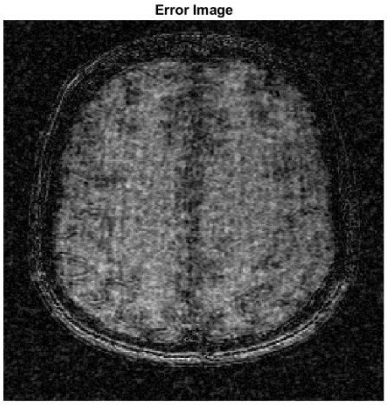
Images & Plots











MATLAB Code

```
disp('1.2')
disp('Sampling Mask and Random Undersampled Images');

load('multicoil-random.mat');
load('reference.mat','ref');

%magnitude image for SoS
figure;set(gcf, 'WindowState', 'maximized');
imshow(log(abs(mask)+1),[]);
title('Undersampling Mask');
saveas(gcf,'1.2_mask.png');

%acceleration
%R_overall = 1/(ratio of k-space acquired)
acquired_kspace = sum(mask(:));
all_kspace = numel(mask(:));
R_overall = 1/(acquired_kspace/all_kspace);
disp(strcat('The overall acceleration corresponding to the given
mask:',num2str(R_overall)));

%take 2DFT of image, and undersample k-space datas with mask
[datau,imu] = undersample(im,mask);

%k-space spectrums for all coils
figure; set(gcf, 'WindowState', 'maximized');
montage(log(abs(datau)+1),'DisplayRange', [], 'Size', [2 4]);
title('Undersampled K-Space Spectrums for All Coils');
saveas(gcf,'1.2_kspace.png');

%magnitude images for all coils
figure;set(gcf, 'WindowState', 'maximized');
montage(abs(imu),'DisplayRange', [], 'Size', [2 4]);
title('Undersampled Magnitude Images for All Coils');
saveas(gcf,'1.2_mag.png');

%SoS reconstruction
```

```
m_sos = sos(imu);

%magnitude image for SoS
figure;set(gcf, 'WindowState', 'maximized');
imshow(abs(m_sos),[]);
title('Final Zero-Fill Image');
saveas(gcf,'1.2_sos.png');

%normalize reference image
ref = abs(ref);
ref = ref/max(ref(:));
%normalize sos result image
m_sos = abs(m_sos); % actually, m_sos is already real-valued image
m_sos = m_sos/max(m_sos(:));

%IQA results
PSNR= psnr(m_sos,ref);
SSIM= ssim(m_sos,ref);
disp(strcat('PSNR:',num2str(PSNR)));
disp(strcat('SSIM:',num2str(SSIM)));
```

[datau,imu] = undersample(im,mask) function

```
function [datau,imu] = undersample(im,mask)
    for coil= 1:8
        datau(:,:,coil) = fft2c(im(:,:,coil));
        datau(:,:,coil) = datau(:,:,coil).*mask;
        imu(:,:,coil) = ifft2c(datau(:,:,coil));
    end
end
```

Q3 - SPIRiT Kernel Weights

32x32 calibration regions from k-space data for each coil is taken by [data_calib] = imcalib(im, calib) function. data_calib has the size of 32x32 and displayed in Images&Plots part of Q3. This function is used inside kernel = calibrate(data_calib,lambda) function.

For the solution of kernel, the sample points in neighborhood for one missing data point is taken by starting from the top, left sample point, going down in the same column, and then skipping to the next column. If the missing point is in the same coil with the sample points in neighbourhood, the central point for 3x3 kernel is skipped, which means there are 8 sample points for the coil of missing point. There are 9 sample points for the missing point coming from the other coils. Size of kernel is 71x8.

Kernel is calculated for lambda = 0.

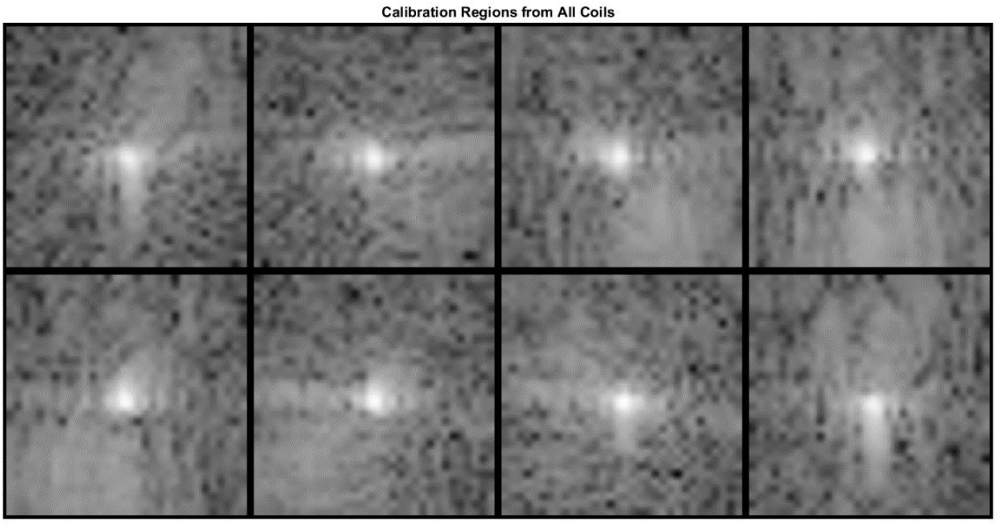
Results

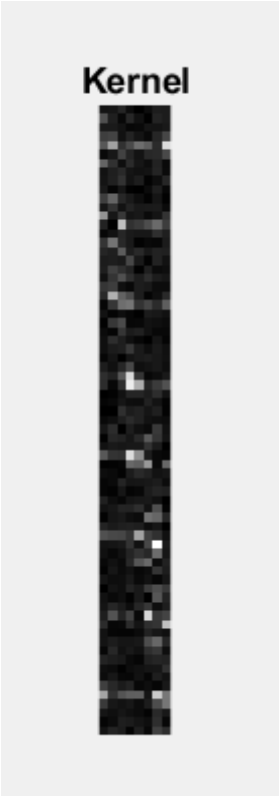
1.3

SPIRiT Kernel Weights

size of kernel:71x8

Images & Plots





MATLAB Code

```
disp('1.3')
disp('SPIRiT Kernel Weights');

load('multicoil-random.mat');

[data_calib] = imcalib(im, calib);
lambda = 0;
kernel =calibrateSpirit(data_calib,lambda);
```

```
%k-space spectrums of calibration regions from all coils
figure;set(gcf, 'WindowState', 'maximized');
montage(log(abs(data_calib)+1),'DisplayRange', [], 'Size', [2 4],
'Border', 5);
title('Calibration Regions from All Coils');
saveas(gcf,'1.3_calib.png');

%magnitude image of SPIRiT kernel
figure;
imshow(abs(kernel),[]);
title('Kernel');
saveas(gcf,'1.3_kernel.png');
size_kernel = size(kernel);
disp(strcat('size of
kernel:',num2str(size_kernel(1)),'x',num2str(size_kernel(2))));
```

[data calib] = imcalib(im, calib) function

```
function [data_calib] = imcalib(im, calib)
%
% inputs
% im : kspace data for all coils
% calib : calibration region size, [calibx caliby]
%
% outputs
% data_calib : calibration region data for all coils, 2D matrix with
size of (32,32)
%

calibx = calib(1); caliby = calib(2);

for coil_no = 1:8
    %2D FFT
    d = fft2c(im(:,:,coil_no));
    data_calib(:,:,coil_no) = d( (end/2 - calibx/2 + 1):(end/2 +
calibx/2) , ...
                                 (end/2 - caliby/2 + 1):(end/2 +
caliby/2) );
end

end
```

kernel =calibrateSpirit(data_calib,lambda)

```
function kernel = calibrateSpirit(data_calib,lambda)

% calibrateSpirit function
%
% inputs
% data_calib : calibration region of  k-space data for all coils,
%              size of(32x32)
% lambda : regularization parameter
%
% outputs
```

```
% kernel :  2D matrix with size of ( 9 x Nc -1 ) x Nc
%
% pre-condition : (calibx,caliby) = (32,32)

[calibx caliby Nc] = size(data_calib);

Ma_coils = [];

%collect coefficients for every coil,
% and every data point inside calibration region of that coil

for coil = 1:Nc

    Ma_coil = [];

    %collect coefficients for every data point
    %inside calibration region,
    %except for first and last row/column
    for col = 2:caliby-1
        for row = 2:calibx-1

            Ma_row = zeros(1,9*Nc-1);

            %go through 71 data points for one missing
            %k-space data point
            ind = 0;

            %take coefficients from all coils
            for coil_no = 1:Nc
                for j = [-1 0  1]
                    for i = [-1 0 1]
                        %take 8 coefficients from the same coil
                        %take 9 coefficients from other coils
                       if( ~( (coil_no==coil) && (i==0 && j== 0) ) )
                           ind = ind + 1; %increase index by 1
                           Ma_row(ind) = ...
                                data_calib(row+i,col+j,coil_no);
                       end
                    end
                end
            end
            %repeat for every data point inside calibration region
            Ma_coil = [Ma_coil; Ma_row];
        end
    end
    %restore coefficient matrix for each coil separately
    Ma_coils(:,:,coil) = Ma_coil;
end

%for each coil, we will find different kernels
%each column of kernel will be to be used in one coil
for k = 1:Nc

    %take data points for each coil from calibration region
    %create vector of data points
    Mkc = [];
```

```
    Mkc = data_calib((2:calibx-1),(2:caliby-1),k);
    Mkc = Mkc(:);

    %take coefficients for each coil
    Ma = Ma_coils(:,:,k);

    %solve linear system of equations
    %with regularized least squares estimation
    ak = inv(Ma'*Ma + lambda*eye(size(Ma'*Ma))) * Ma' * Mkc;

    %set nan and inf values to zero
    ak(isnan(ak)) = 0;
    ak(isinf(ak)) = 0;

    %each column of kernel has coefficients for each coil
    a(:,k) = ak(:);

end

%set a to kernel
kernel = a;

end
```

Q4 - SPIRiT Reconstruction

I have plotted PSNR and SSIM values fort he lambdas from 0 to 1 with the step size of 0.1 and quantitative results are deteriorating as lambda increases. This can be seen in of PSNR and SSIM in Image&Plots part of Q4. Therefore, I have chosen a small value for lambda, which is 0.1.

In SPIRiT reconstruction, kernel is applied everywhere, including calibration region. I also tried by entering inside the calibration region. There was no change in PSNR and SSIM, which shows that we have consistency with calibration region. Applying SPIRiT reconstruction inside calibration region do not change our results, because there should be no change for missing points when the k-space data is full acquired.

Error image is between reference image and SoS reconstruction of SPIRiT reconstructed coil images.

PSNR increased to 22.4693 from 22.4526, and SSIM increased to 0.84198 from 0.83054 for the randomly undersampled case. When we apply data consistency criterion, PSNR and SSIM values will improve more than these obtained values.

Results

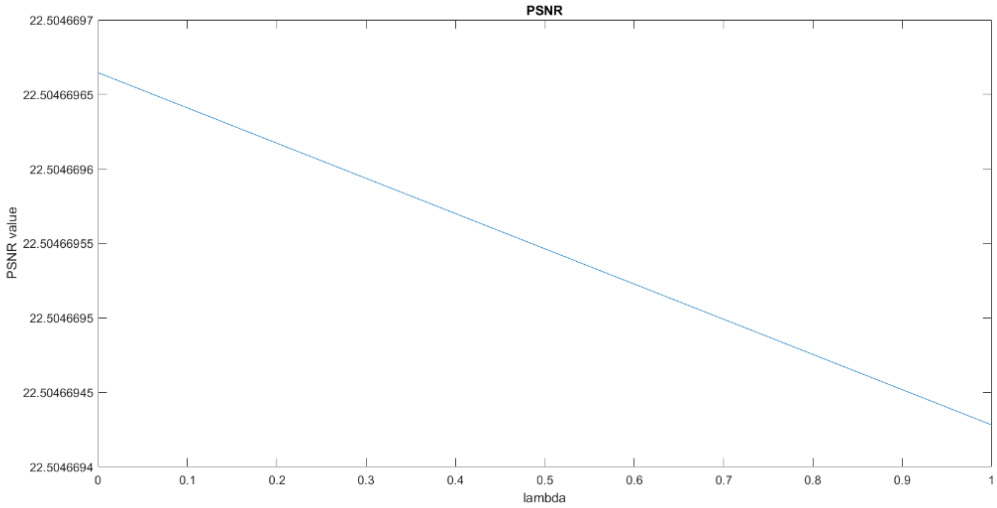
1.4

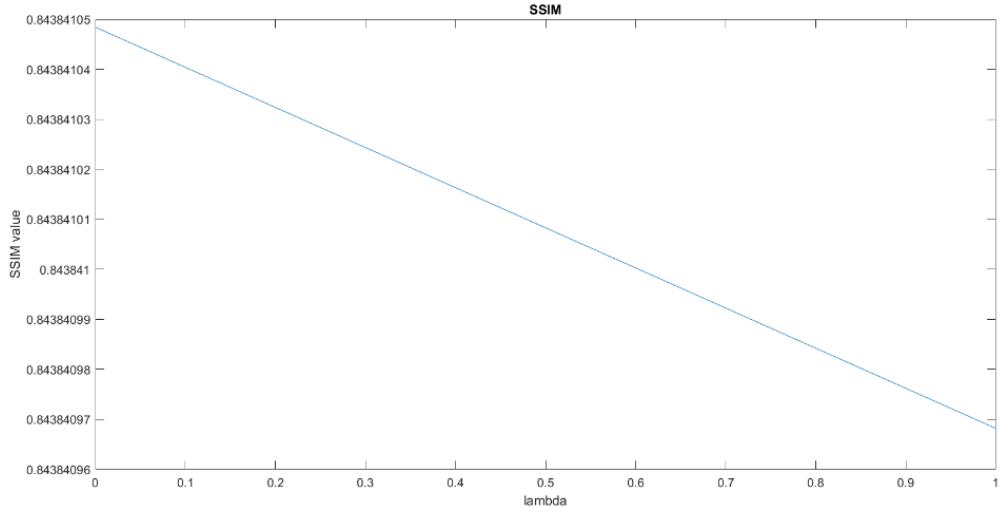
SPIRiT Reconstruction

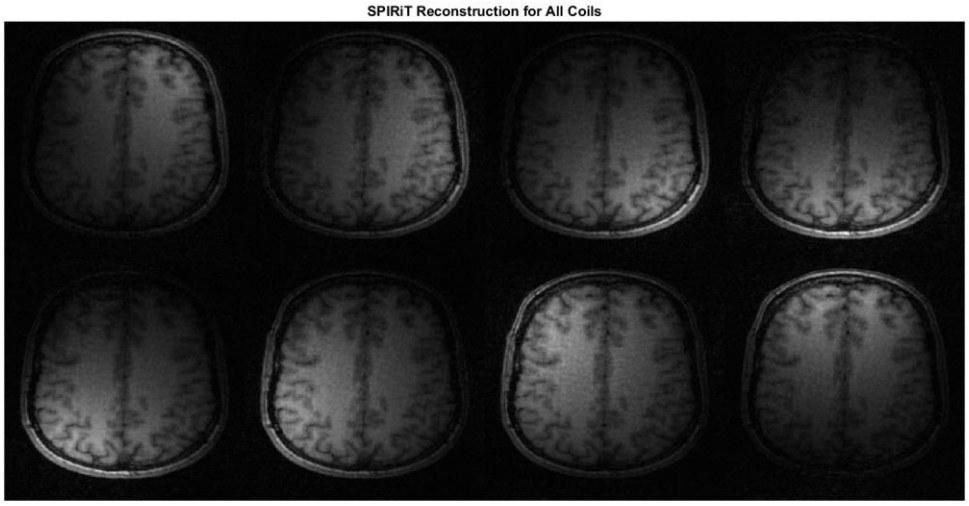
PSNR:22.4693

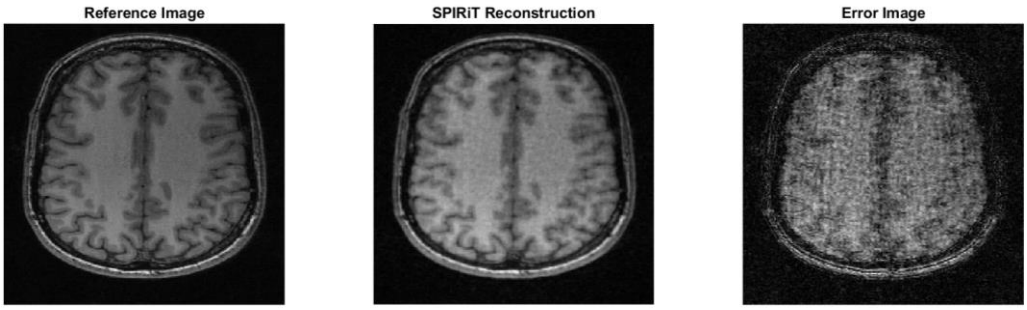
SSIM:0.84198

Images & Plots









MATLAB Code

```
disp('1.4')
disp('SPIRiT Reconstruction')

load('multicoil-random.mat');
load('reference.mat');

lambda = 1e-1; % by trial-and-error

%spirit reconstruction
[datau,imu] = undersample(im,mask);
[data_calib] = imcalib(im, calib);
kernel =calibrateSpirit(data_calib,lambda);
imr = spirit(datau, kernel);

%sos reconstruction of coil images found by spirit reconstruction
m_sos = sos(imr);

%magnitude images for all coils by SPIRiT reconstruction
figure;set(gcf, 'WindowState', 'maximized');
montage(abs(imr),'DisplayRange', [], 'Size', [2 4]);
title('SPIRiT Reconstruction for All Coils');
saveas(gcf,'1.4_spirit.png');

%normalize reference image
ref = abs(ref);
ref = ref/max(ref(:));

%normalize sos result image
m_sos = abs(m_sos); % actually, m_sos is already real-valued image
m_sos = m_sos/max(m_sos(:));

%IQA results
PSNR= psnr(m_sos,ref);
SSIM= ssim(m_sos,ref);
disp(strcat('PSNR:',num2str(PSNR)));
disp(strcat('SSIM:',num2str(SSIM)));

%reference image
figure;set(gcf, 'WindowState', 'maximized');
subplot(1,3,1); imshow(abs(ref), []);
title('Reference Image');
```

```
%SoS reconstruction
subplot(1,3,2); imshow(abs(m_sos), []);
title('SPIRiT Reconstruction');
%error image
subplot(1,3,3);imshow(abs(m_sos-ref), []);
title('Error Image');
saveas(gcf,'1.4.png');
```

imr = spirit(datau, kernel) function

```
function imr = spirit(datau, kernel)

% spirit function
% this function applies spirit reconstruction without data
consistency
%
% inputs
% datau : undersampled k-space data with calibration region
% kernel : spirit kernel weights
%
% outputs
% imr : size of (Nx x Ny x Nc)
%
% pre-condition : (calibx,caliby) = (32,32)

%calibration region
calibx = 32; caliby = 32;

%take sizes of k-space data
[Nx Ny Nc] = size(datau);

%set Mr to zero matrix
Mr = zeros(Nx, Ny, Nc);

%find spirit reconstruction for all coils
for k = 1:Nc
    for col = 2:Ny-1
        for row = 2:Nx-1

            %take coefficients for one coil from kernel,
            %each column is for one coil
            kernel_temp = kernel(:,k);

            %multiplications of samples in neighbourhood with
            %their corresponding coefficients
            sum = 0;

            %go thorugh 71 samples in the neighbourhood of
            %the calculated data point
            ind = 1;

            %take coefficients calculated for all coils
            for coil = 1:Nc
```

```
                for j = [-1 0 1]
                    for i = [-1 0 1]

                        %take 8 samples from the same coil
                        %take 9 samples from other coils
                        if( ~( (coil==k) && (i==0 && j== 0) ) )

                            %take corresponding coefficient
                            %for the sample point

                            coefficient = kernel_temp(ind);
                            %sum up the multiplications of sample
                               %data point and
                            %and their corresponding coefficients

                            sum = sum + coefficient * ...
                                datau(row+i,col+j,coil);

                            %index to go through all coefficients
                            ind = ind + 1;
                        end
                    end
                end
            end

            %put the found value to its position
            Mr(row,col,k) = sum;

        end
    end
end

for coil = 1:Nc
    %find spirit reconstructed images for all coils
    imr(:,:,coil) = ifft2c(Mr(:,:,coil));
end

end
```

Q5 - l1 Regularization in Wavelet Domain

In *l*1 regularization in wavelet domain for medical image reconstruction, there is a trade-off between effective noise removal and loss of resolution. As beta increases, the thresholded parts in wavelet domain increases, which means there is loss of resolution in image domain.

The loss of resolution can be seen when Beta is 1. We lost the high frequency components in our image and this is seen in error image when Beta is 1. When Beta is 0.1, there is loss of resolution again, but not as much as when Beta is 1. Best result comes when Beta is 0.01. When tryring different betas in Q6, we should remember that too small Beta value will not be working for noise removal, therefore we should see how much we should decrease Beta value by trial-and-error.

SSIM value is higher than Q2 when beta is 0.01, but the PSNR value is still lower than Q2. Beta may be still higher than we desire and this fact results in loss of resolution. Hence, PSNR value might have decreased for 0.01.

Results

1.5

l1 Regularization in Wavelet Domain

PSNR:21.9719, Beta:0.01

SSIM:0.84556, Beta:0.01

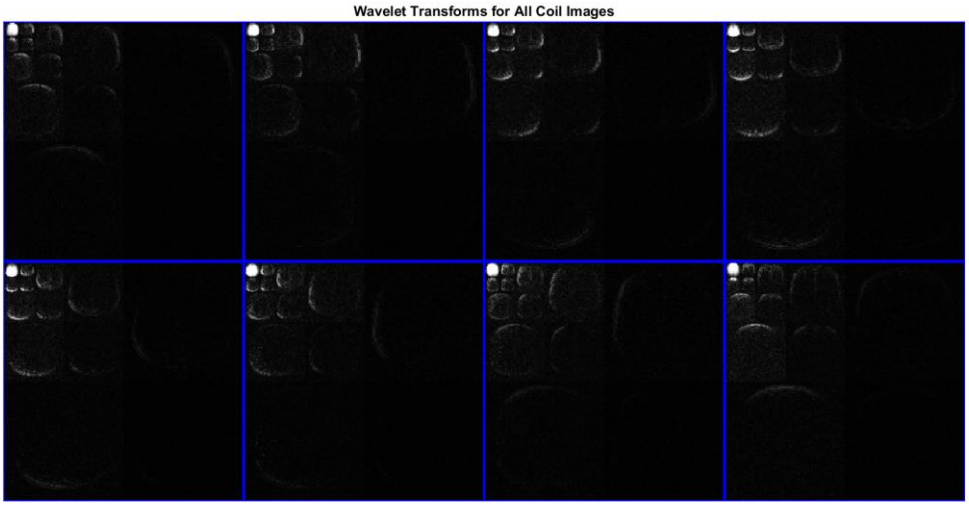
PSNR:18.4201, Beta:0.1

SSIM:0.59727, Beta:0.1

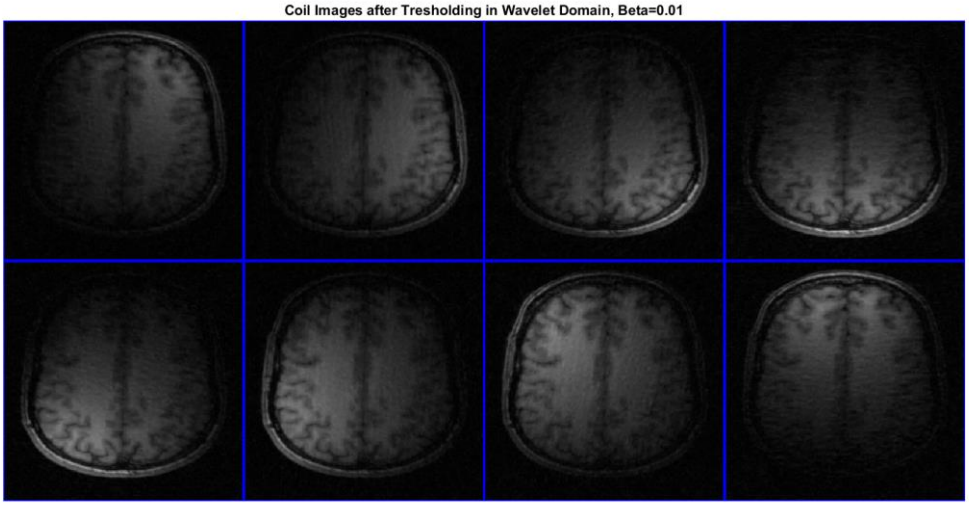
PSNR:16.6136, Beta:1

SSIM:0.33688, Beta:1

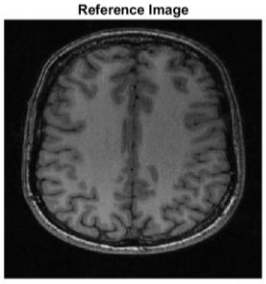
Images & Plots



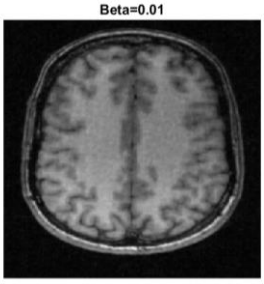
Coil Images after Tresholding in Wavelet Domain, Beta=0.01



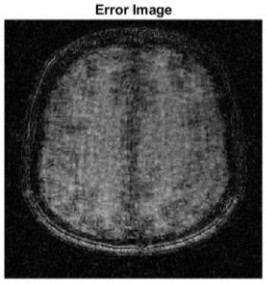
Reference Image



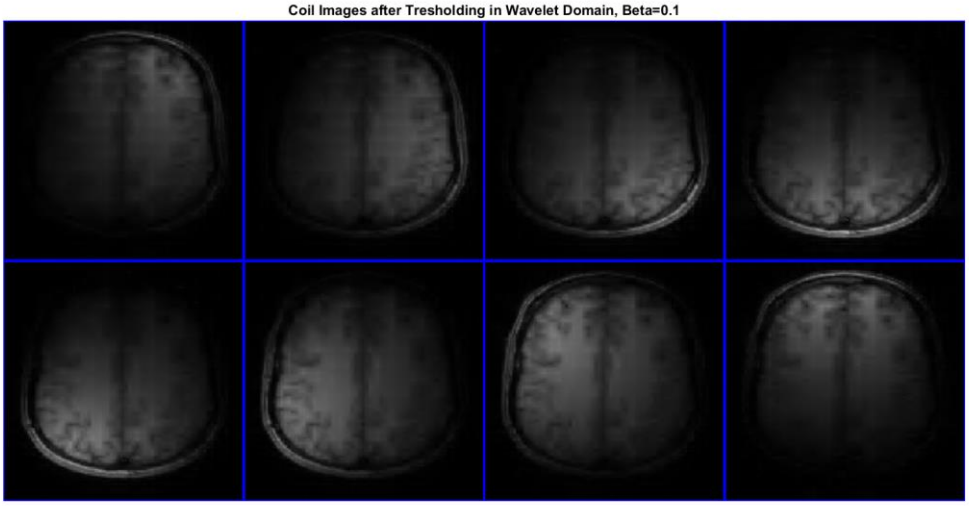
Beta=0.01

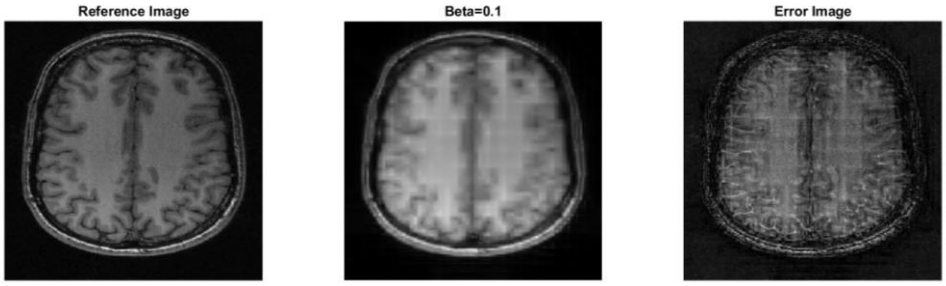


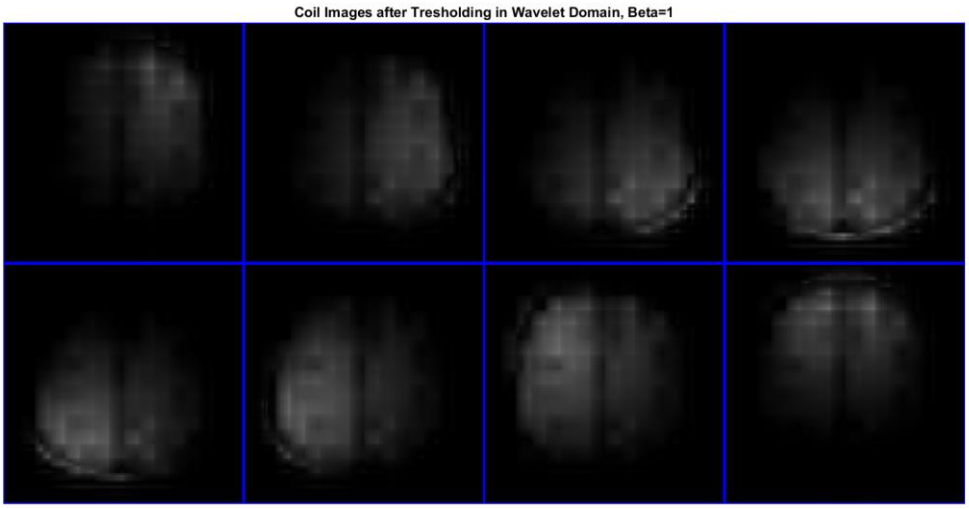
Error Image

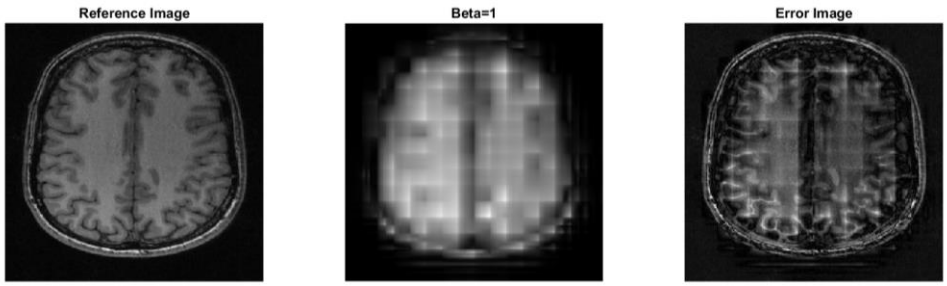


Coil Images after Tresholding in Wavelet Domain, Beta=0.1









MATLAB Code

```
disp('1.5');
disp('l1 Regularization in Wavelet Domain');

load('multicoil-random.mat');

load('reference.mat');
%normalize reference image
ref = abs(ref);
ref = ref/max(ref(:));

%undersampled images for all coils
```

```
[datau,imu] = undersample(im,mask);

%generate wavelet operator
wv = Wavelet('Daubechies',4,4);
for i = 1:8
    imr = imu(:,:,i);
    imwv(:,:,i) = wv*imr; % take forward wavelet transform
end

%wavelet transform images for all coils
figure;set(gcf, 'WindowState', 'maximized');
montage(log(abs(imwv)+1),'Size', [2 4], 'BorderSize',
2,'BackgroundColor', 'blue');
title('Wavelet Transforms for All Coil Images');
saveas(gcf,'1.5_wavelet.png');

%different beta values to see its effect on reconstruction images
for beta = [0.01, 0.1, 1]

    %apply l1 regularization in wavelet domain
    for i = 1:8
        imr = imu(:,:,i);
        imth(:,:,i) = l1wavelet(imr,beta);
    end

    %magnitude images from all coils after tresholding
    figure;set(gcf, 'WindowState', 'maximized');
    montage(abs(imth),'Size', [2 4], 'BorderSize',
2,'BackgroundColor', 'blue');
    title(strcat('Coil Images after Tresholding in Wavelet Domain,
Beta=',num2str(beta)));
    saveas(gcf,strcat('1.5_coils_beta',num2str(beta),'.png'));

    %SoS reconstruction
    m_sos = sos(imth);

    %normalize sos result image
    m_sos = abs(m_sos); % actually, m_sos is already real-valued
image
    m_sos = m_sos/max(m_sos(:));

    %IQA results
    PSNR = psnr(ref,m_sos);
    SSIM = ssim(ref,m_sos);
    disp(strcat('PSNR:',num2str(PSNR),', Beta:',num2str(beta)));
    disp(strcat('SSIM:',num2str(SSIM),', Beta:',num2str(beta)));

    %reference image
    figure;set(gcf, 'WindowState', 'maximized');
    subplot(1,3,1); imshow(ref, []);
    title('Reference Image');
    %SoS reconstruction
    subplot(1,3,2); imshow(m_sos, []);
    title(strcat('Beta=',num2str(beta)));
    %error image
    subplot(1,3,3);imshow(abs(m_sos-ref), []);
```

```
    title('Error Image');
    saveas(gcf,strcat('1.5_beta',num2str(beta),'.png'));

end
```

imth = l1wavelet(imr,beta) function

```
function imth = l1wavelet(imr,beta)
% l1wavelet function
%
% inputs
% imr : reconstructed 2D image
% beta : treshold in wavelet domain
%
% outputs
% imth :  wavelet regularized imr with beta constant
%

wv = Wavelet('Daubechies',4,4); % generate wavelet operator
coeffW = wv*imr; % take forward wavelet transform
%l1 regularization by soft tresholding
%apply soft tresholding in each elet coefficient
S_beta = (coeffW./(abs(coeffW))).*max(0,abs(coeffW)-beta);
imth = wv'*S_beta; % take inverse wavelet transform

end
```

Q6 - Iterative l1-SPIRiT Reconstruction

Lambda is 0.1 as it was chosen in Q4, and Beta is chosen as 0.001 by trial-and-error. When Beta was chosen smaller than 0.001, noise removal started to decrease and I could not see the effect of l1 regularization in wavelet domain as much as it was desired.

This iterative solution is quite fast and we almost converge to the real image around 10 iterations. In each iteration, there is decrease in magnitude image of error image, which was expected. The last iteration gives very good error image despite having some noise in the central region of image.

Until a certain point, there is increase in PSNR and SSIM values. Later, they start to decrease even if the visual results are improving. The reason might be as we converge to the result image by SPIRiT reconstruction, thresholding in wavelet domain might cause loss of image components. Or we are simply denoising the reconstructed, converged image. Still, this loss of image components are not noticeable by human eye.

Even if it cannot be noticed by human eye, there is background noise in reference image. By applying l1-SPIRiT reconstruction, we get rid of the background noise by applying thresholding in wavelet domain. This may cause PSNR and SSIM values to decrease after a certain point.

Results

1.6

Iterative l1-SPIRiT Reconstruction

PSNR for Iteration#1:26.3143

SSIM for Iteration#1:0.87876

PSNR for Iteration#2:28.7999

SSIM for Iteration#2:0.89616

PSNR for Iteration#3:30.7178

SSIM for Iteration#3:0.9037

PSNR for Iteration#4:32.1566

SSIM for Iteration#4:0.90743

PSNR for Iteration#5:33.2716

SSIM for Iteration#5:0.90944

PSNR for Iteration#6:34.0615

SSIM for Iteration#6:0.91029

PSNR for Iteration#7:34.6348

SSIM for Iteration#7:0.91078

PSNR for Iteration#8:34.9824

SSIM for Iteration#8:0.91069

PSNR for Iteration#9:35.2039

SSIM for Iteration#9:0.91063

PSNR for Iteration#10:35.285

SSIM for Iteration#10:0.91016

PSNR for Iteration#11:35.3121

SSIM for Iteration#11:0.9099

PSNR for Iteration#12:35.2582

SSIM for Iteration#12:0.90925

PSNR for Iteration#13:35.1961

SSIM for Iteration#13:0.90893

PSNR for Iteration#14:35.0847

SSIM for Iteration#14:0.90817

PSNR for Iteration#15:34.9898

SSIM for Iteration#15:0.90789

PSNR for Iteration#16:34.8604

SSIM for Iteration#16:0.90702

PSNR for Iteration#17:34.7611

SSIM for Iteration#17:0.90681

PSNR for Iteration#18:34.6347

SSIM for Iteration#18:0.90577

PSNR for Iteration#19:34.5424

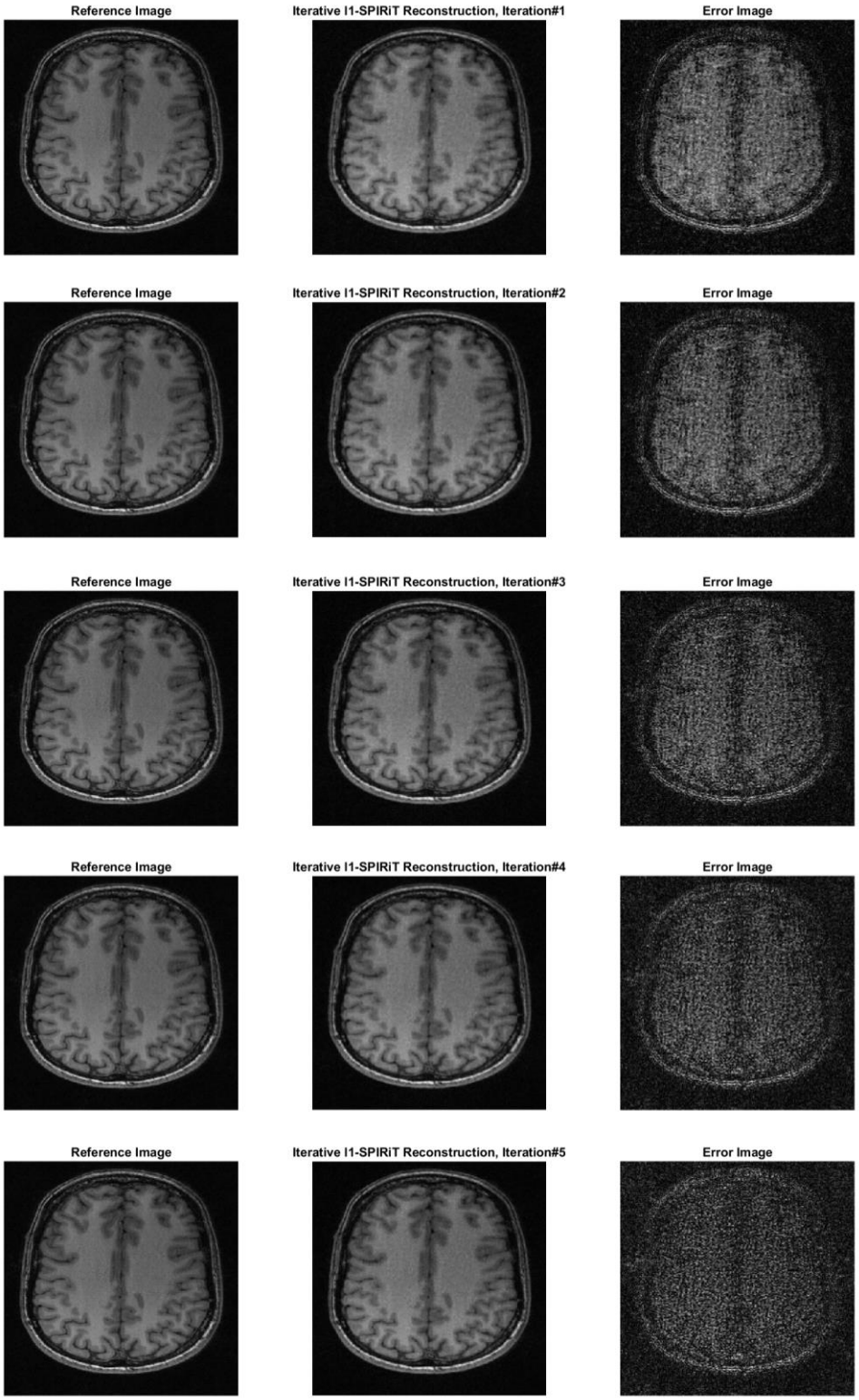
SSIM for Iteration#19:0.90552

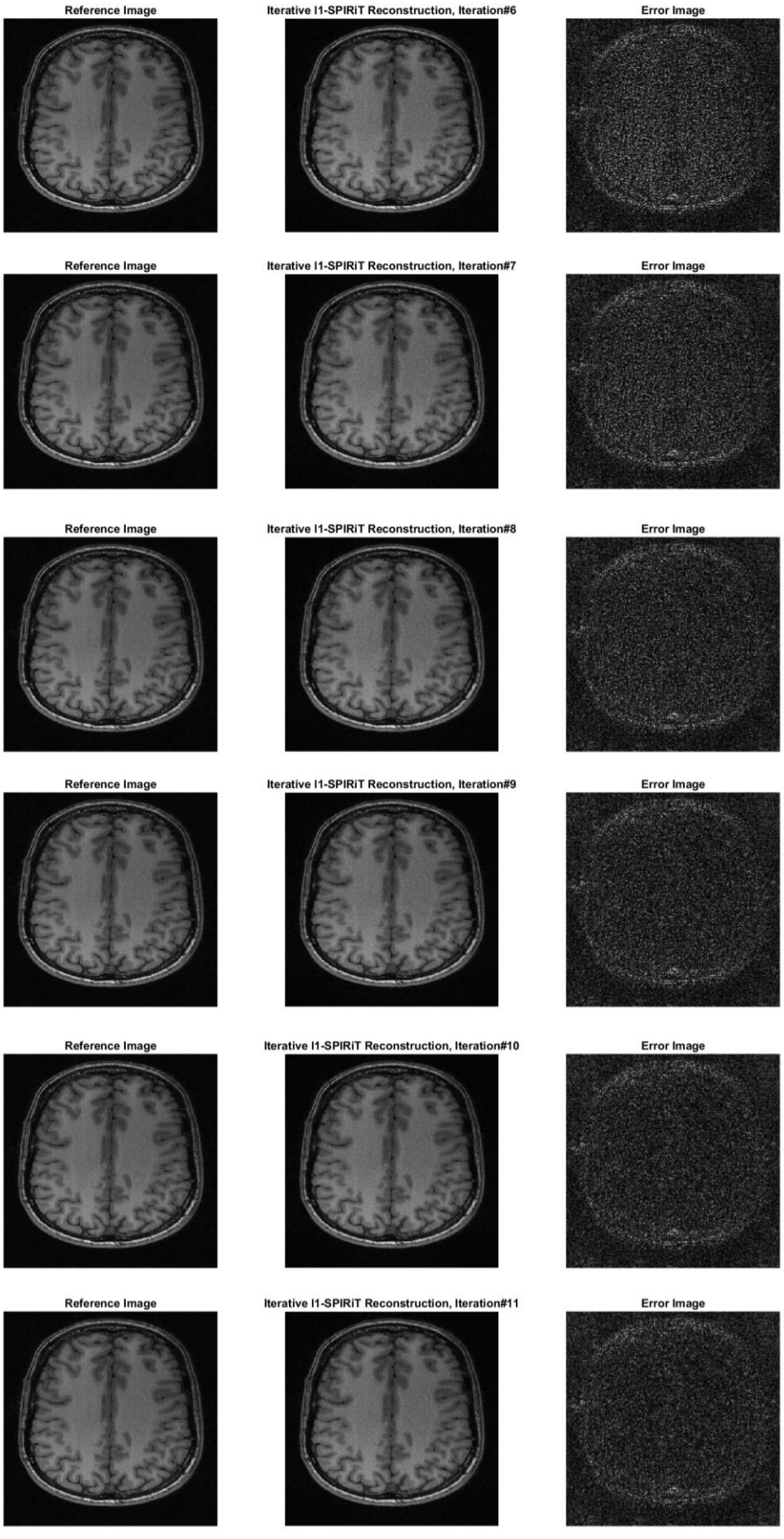
PSNR for Iteration#20:34.408

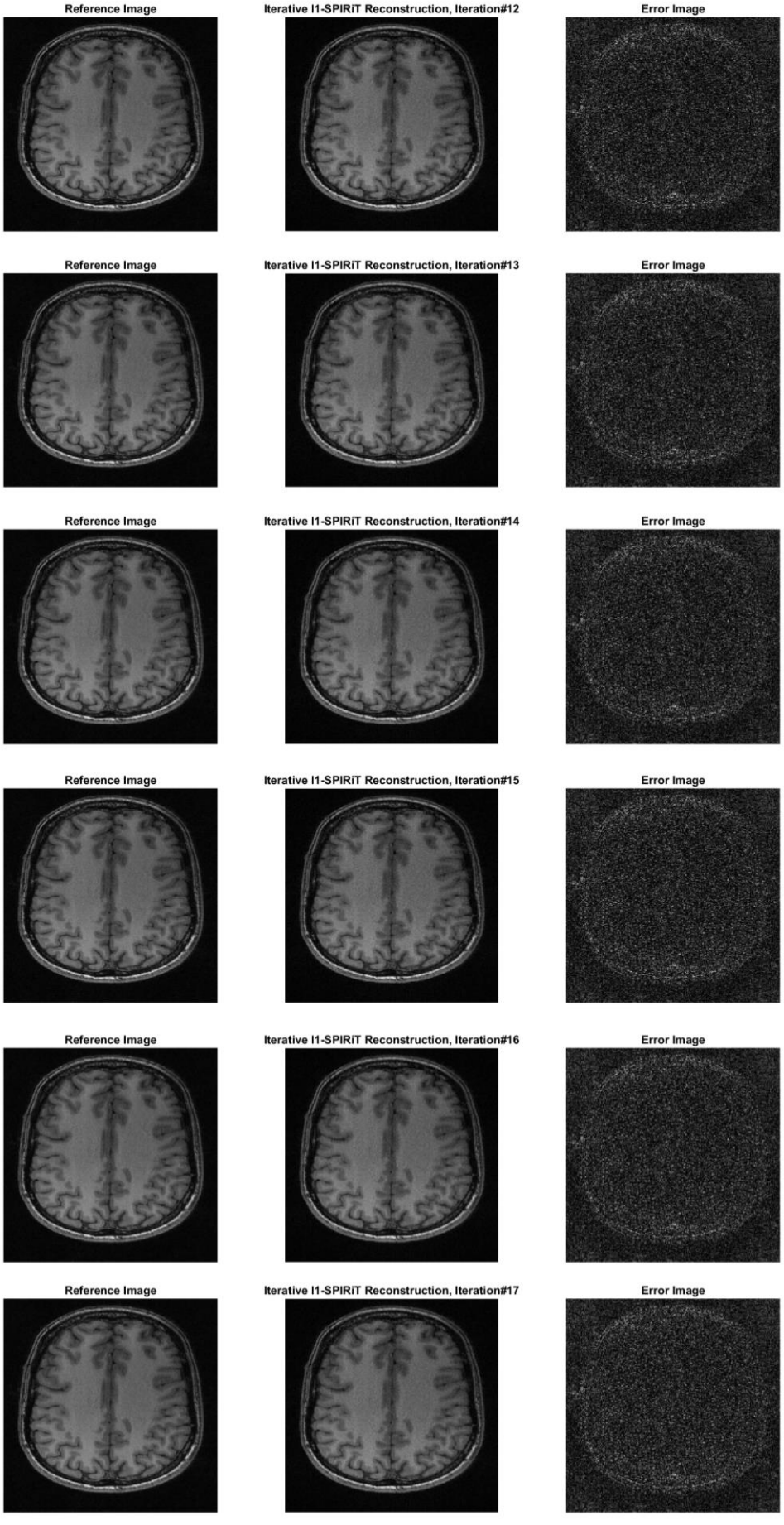
SSIM for Iteration#20:0.90384

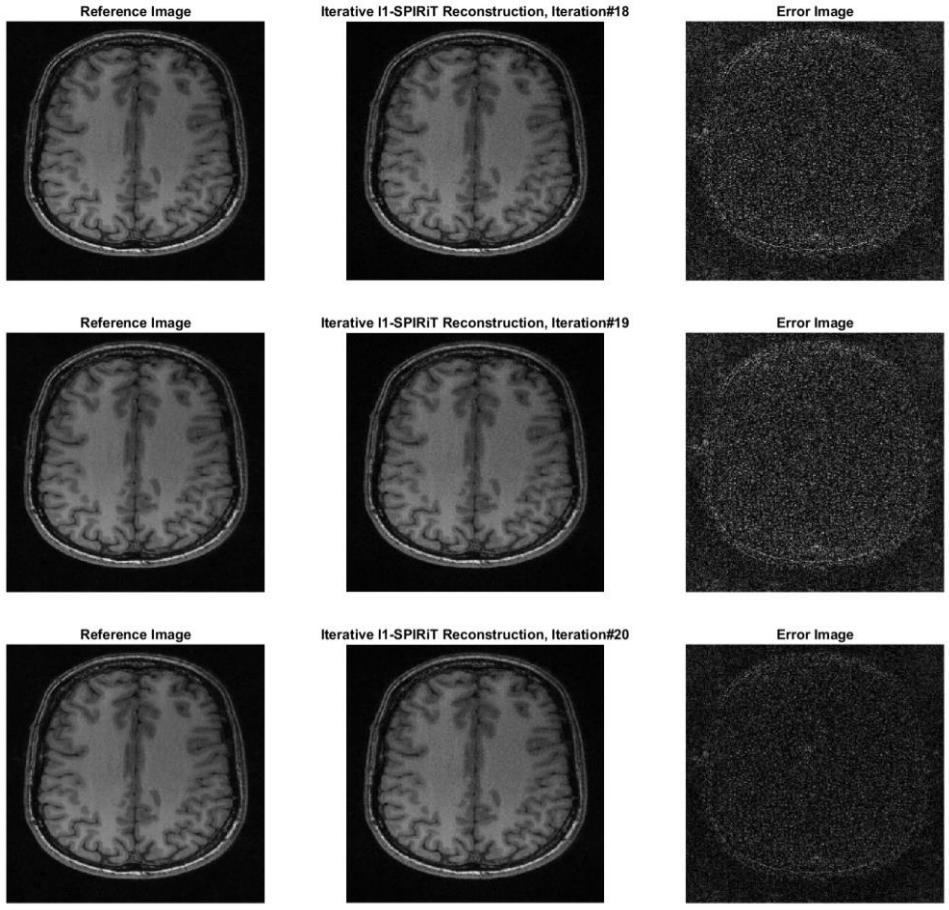
Beta:0.001

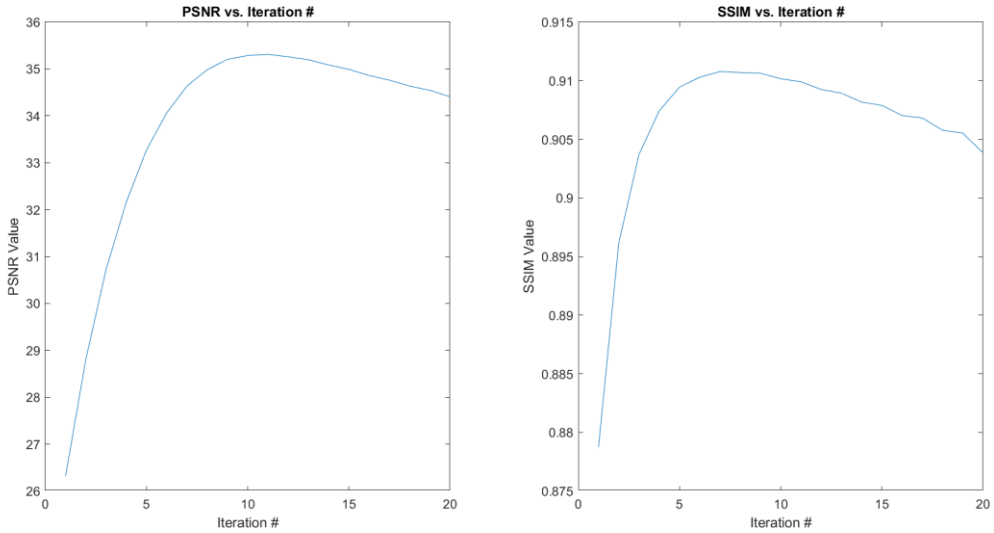
Images & Plots




Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#6
Error Image
Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#7
Error Image
Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#8
Error Image
Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#9
Error Image
Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#10
Error Image
Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#11
Error Image


Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#12
Error Image
Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#13
Error Image
Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#14
Error Image
Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#15
Error Image
Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#16
Error Image
Reference Image
Iterative l1-SPIRiT Reconstruction, Iteration#17
Error Image





MATLAB Code

```
disp('1.6')
disp('Iterative l1-SPIRiT Reconstruction');

load('multicoil-random.mat');

[datau,imu] = undersample(im,mask); %undersampled images from each
coil
```

```
[data_calib] = imcalib(im, calib); %calibration region  from each
coil
lambda = 1e-1; %regularization parameter
kernel = calibrateSpirit(data_calib,lambda); %kernel calculated for
each coil

%Iterative l1-SPIRiT Reconstruction
%selected beta value near to 1/500 of maximum coefficient from coil
coefficients
beta = 0.001; %by trial-and-error
numiter = 20; %iteration number
imr = l1spirit(datau,kernel, beta, numiter);

%Outputs are computed inside l1spirit function!
```

imr = l1spirit(datau,kernel, beta, numiter) function

```
function imr = l1spirit(datau,kernel, beta, numiter)
% l1spirit function
%
% inputs
% datau : randomly undersampled k-space data
% kernel :
% beta : treshold in wavelet domain
% numiter : iteration number
%
% outputs
% imr : iterative l1-SPIRiT reconstruction image
%
% STEPS
% 1. Apply SPIRiT to reconstruct full k-space images for each coil.
% 2. Apply l1 regularization in wavelet domain to each image.
% Choose a reasonable beta by trial and error, and keep it constant
across iterations.
% 3. Enforce data consistency in k-space by replacing the
reconstructed
% k-space data with the acquired data at sampled k-space locations.

load('reference.mat');
%normalize reference image
ref = abs(ref);
ref = ref/max(ref(:));

[Nx Ny Nc] = size(datau);

kspace = datau; % first iteration, undersampled acquired k-space
data

for iter = 1:numiter

    %spirit reconstruction
    imr = spirit(kspace, kernel);

    %l1 regularization in wavelet domain
    for coil = 1:Nc
```

```
        %wavelet regularized image
        imth(:,:,coil) = l1wavelet(imr(:,:,coil),beta);
        %k-space data of the wavelet regularized image
        kspace(:,:,coil) = fft2c(imth(:,:,coil));
    end

    %data consistency
    %replace the acquired (i.e. non-zero points)
    %from acquired randomly undersampled data (datau)
    kspace(datau~=0) = datau(datau~=0);

    %take inverse 2DFT
    for coil = 1:Nc
        imr(:,:,coil) = ifft2c(kspace(:,:,coil));
    end

    %final SoS reconstrucion
    m_sos = sos(imr);

    %normalize sos result image
    m_sos = abs(m_sos); % actually, m_sos is already real-valued
image
    m_sos = m_sos/max(m_sos(:));

    %reference image
    figure;set(gcf, 'WindowState', 'maximized');
    subplot(1,3,1); imshow(ref, []);
    title('Reference Image');
    %SoS reconstruction
    subplot(1,3,2); imshow(m_sos, []);
    title(strcat('Iterative l1-SPIRiT Reconstruction, Iteration#',
num2str(iter)));
    %error image
    subplot(1,3,3);imshow(abs(m_sos-ref), []);
    title('Error Image');
    saveas(gcf,strcat('1.6_iter#',num2str(iter),'.png'));

    PSNR(iter) = psnr(ref,m_sos);
    SSIM(iter) = ssim(ref,m_sos);
    disp(strcat('PSNR for
Iteration#',num2str(iter),':',num2str(PSNR(iter))));
    disp(strcat('SSIM for
Iteration#',num2str(iter),':',num2str(SSIM(iter))));

end

%PSNR and SSIM values vs. Iteration Number Plots
figure;set(gcf, 'WindowState', 'maximized');
subplot(1,2,1); plot((1:numiter),PSNR); title("PSNR vs. Iteration
#");
xlabel('Iteration #');ylabel('PSNR Value');
subplot(1,2,2); plot((1:numiter),SSIM); title("SSIM vs. Iteration
#");
xlabel('Iteration #');ylabel('SSIM Value');
saveas(gcf,'1.6_psnr_ssim.png');
end
```