EE369C: Assignment 5 Solutions

Due Thursday, Nov. 1

The problems this week will again be concerned with parallel imaging. We will look at coil compression, and k-space parallel reconstruction. The data set is the same as the one from last week.

1. Coil Compression Even though we have eight receive channels, we can only get a smaller acceleration factor. Here we will look at the reasons for this. We can exploit this to reduce the number of channels we need to reconstruct, and this can be important for large 3D array coils.

a. Eigencoils The coils sensitivities are not orthogonal, and span a space that has a lower dimension than the number of channels. To see this we'll compute the eigencoils that correspond to the set of coil sensitivities.

We first need to create a 2D matrix that describes the coils

```
[nx ny nc] = size(map);
m2 = reshape(map,nx*ny,nc)
```

This is a matrix that has an entire vectorized map in each column, with one column per coil. The eigen decomposition is

```
[ev ed] = eig(m2'*m2);
```

This eigenvalues are on the diagonal of `ed`, and the corresponding eigenvectors as columns of `ev`.

Each eigenvalue and eigenvector correspond to a combination of coils, with largest eigenvalue the most important combination. Reconstruct the eigencoil maps by

```
ecmap= reshape(m2*ev,nx,ny,nc);
```

Display the magnitude and phase of the eigencoil maps, in descending order of significance. The energy in one of the maps is the eigenvalue. How many terms do you need before we get to less than 1% of the energy? Note that the as the total energy decreases, the maps become more localized to the edge of the head. Also, note that the phase maps have phase wraps in the angular direction. How many wraps does the j^{th} map have?

Solution The magnitude and the phase of the eigencoils maps are shown in Fig. 1. The magnitudes are all scaled the same. Note that by the fourth eigencoil there is very little energy left. By the fifth coil we are down to 0.5% of the energy in the first eigencoil. For the coils that have significant energy, the number of phase wraps is the same as the coil order.

b. g-Factor It looks like there are only four or five significant eigencoils resulting from the eight array coils. Compare the g-factor maps for all eight array coils compared to the most significant 4, 5, and 6 eigencoils. Choose an acceleration of `Rx=Ry=2`.

Solution Using the g-factor m-file from last week, we get the g-factor maps shown in Fig. 2. Adding eigencoils beyond 5 doesn't improve the g-factor maps. Only a little improvement is seen in going from 4 to 5 eigencoils.

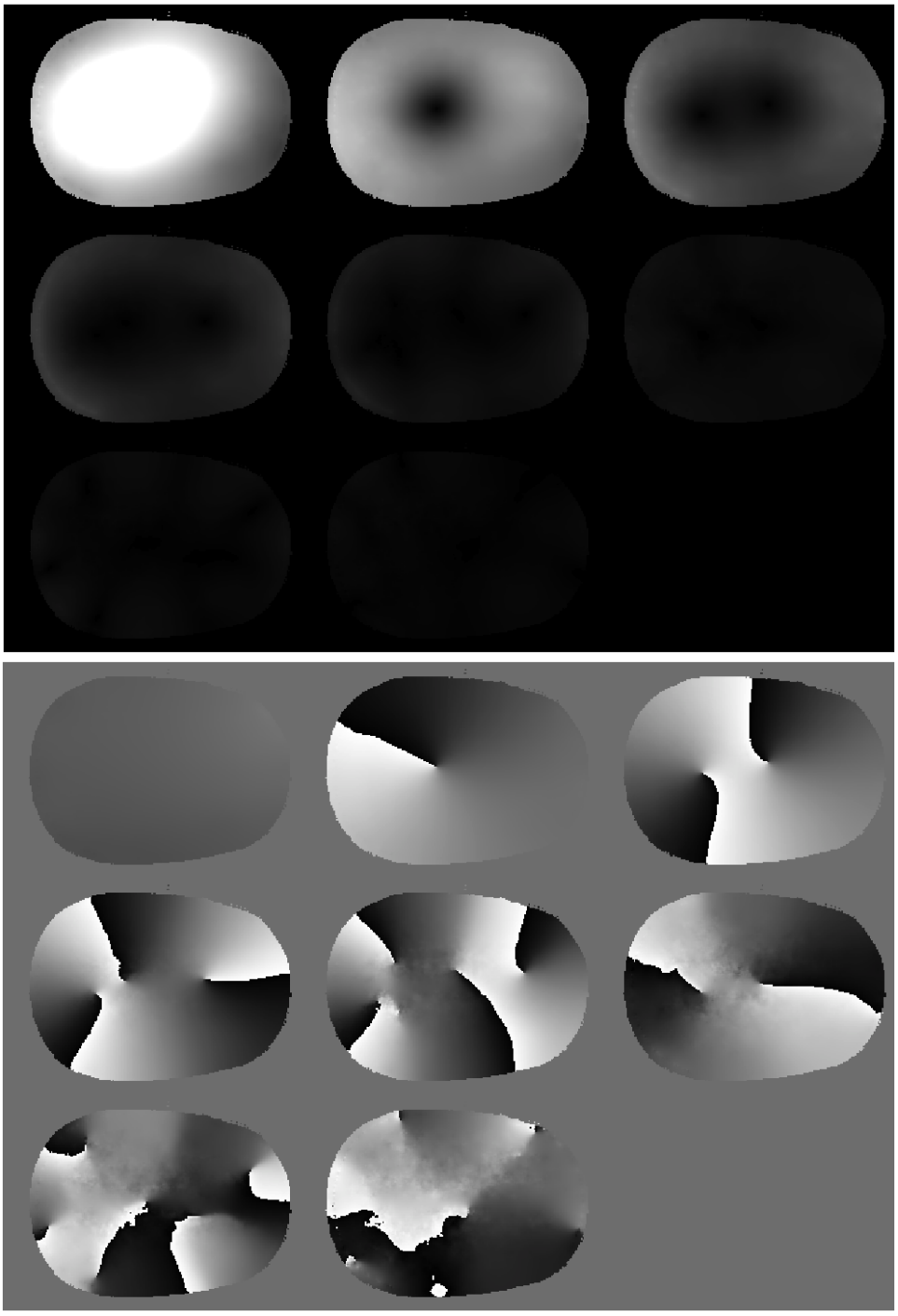


Figure 1: The magnitude (top) and phase (bottom) of the eigencoils. The magnitude data is all on the same scale. After four coils, the coil energy is below 1%.

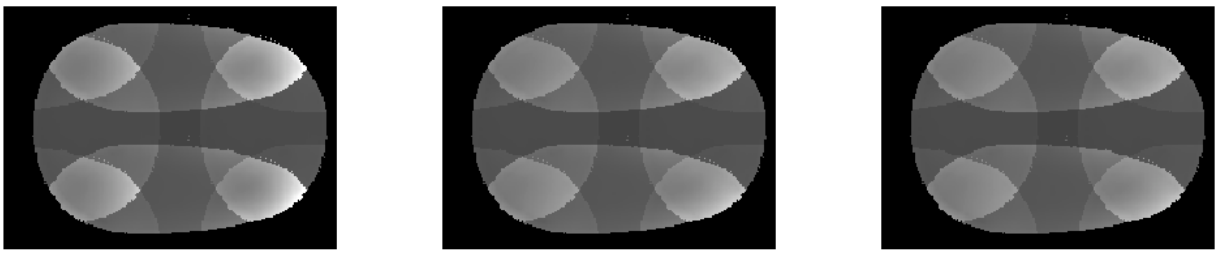


Figure 2: The g-factor maps for a 2×2 accelerations with 4, 5 and 6 eigencoils. Adding eignecoils beyond 5 doesn't improve the g-factor maps.

c. Coil Compression We can compute a reduced data set the same way as we computed the eigencoil maps. Make images corresponding to the different eigencoils, and display them on the same scale.

Solutions Using the same procedures as above for the coil maps, we get the following images corresponding to the eigencoils, shown in Fig. 3.

d. SENSE Reconstruction You can speed up your reconstruction by only using a subset of the eigencoils. Undersample the data from the previous part by `Rx=Ry=2`, and reconstruct it using 4, 5, and 6 of the eigencoils.

Solution Using the coil compressed data, and 4, 5 and 6 of the eigencoils, we get the reconstructions shown in Fig. 4.

2. Parallel Imaging with SPIRiT There are several different parallel reconstruction algorithms that operate in k-space. We'll look at the SPIRiT algorithm that we talked about in class. This is an iterative algorithm that doesn't require calibration for a specific sampling pattern. This will be useful for combining parallel imaging, and compressed sensing.

To help you get started, we'll give you two routines that are available on the web site. The first is

```
sk = spirit_kernel(mc,Nk)
```

This takes a calibration region `mc` and a kernel size `Nk`, and returns the SPIRiT kernel `sk`. The kernel has an `Nk` by `Nk` by `Nc` kernel for every coil. The second routine is

```
ms = apply_spirit(m,sk)
```

This takes k-space data `m` and applies the kernel `sk`, and returns the result.

a) Generate the SPIRiT kernel The data you have is in image space. Transform it to k-space and save it in a variable `m`. It may be useful to define functions

```
function y = fft2c(x)
y = fftshift(fft2(fftshift(x)))
```

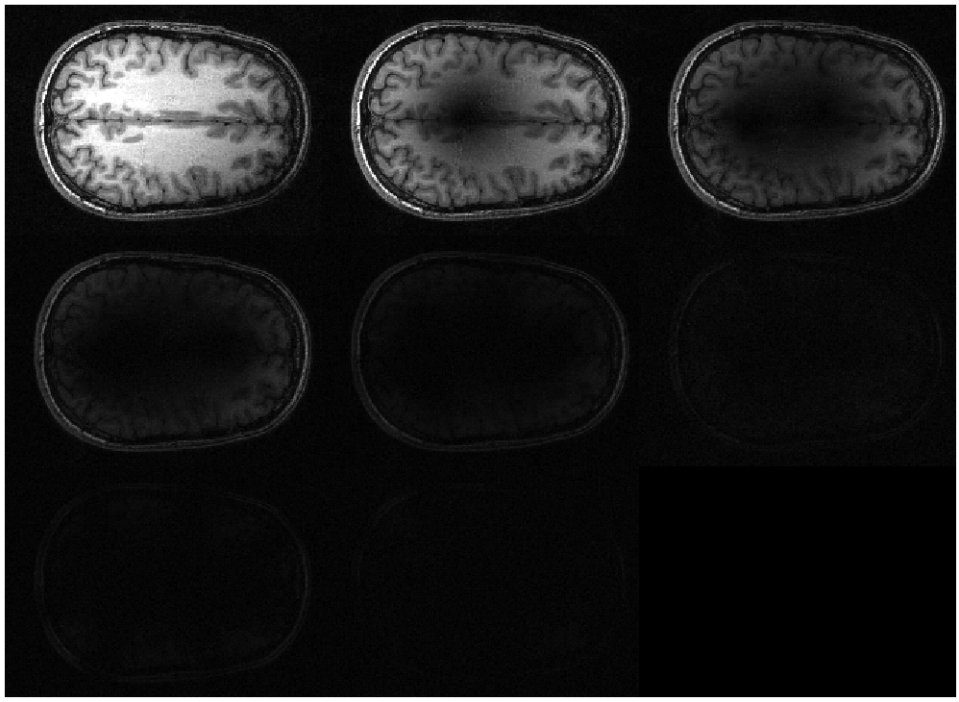


Figure 3: Images corresponding to the eignecoils.

and the corresponding `ifft2c()` to perform the centered transforms.

For the calibration region, extract the central 24 by 24 by 8 block of the k-space data, and call it `mc`. Compute a 5 by 5 SPIRiT kernel,

```
sk = spirit_kernel(mc,5)
```

Ideally if we apply this kernel to the full data set, we should get the same data back. Apply the kernel for 10 iterations. Display one of the original coil images, the reconstructed coil image after 10 iterations, and the difference image multiplied by 10.

Solution The original data for coil 4, the difference after applying the SPIRiT operator 10 times, and the difference multiplied by 10 are shown in Fig. 5.

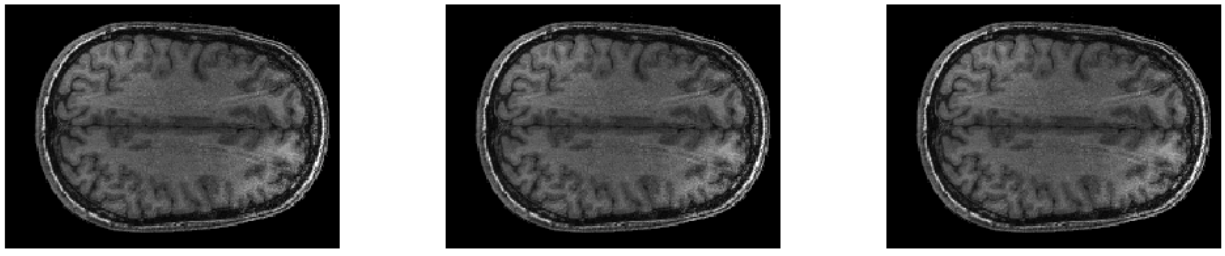


Figure 4: Coil compressed reconstructions for a 2×2 accelerations with 4, 5 and 6 eigencoils. The image quality is very similar for all of these.

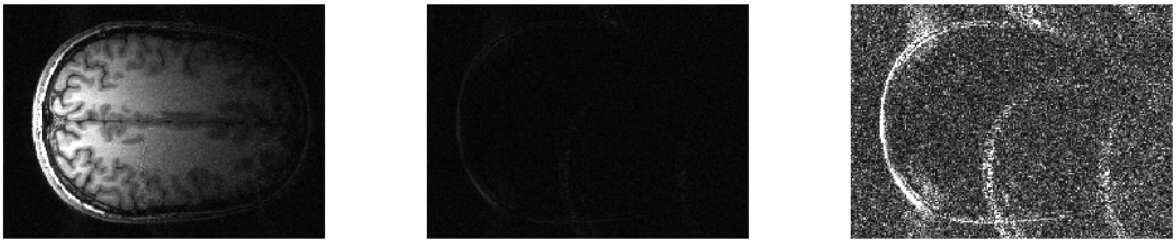


Figure 5: The original data for coil 4, the difference after applying the SPIRiT operator 10 times, and the difference multiplied by 10

b) SPIRiT Reconstruction Write a function that takes an undersampled k-space data set and a calibration data set, and computes the SPIRiT reconstruction. It will alternately apply data consistency, and consistency with the kernel. Since this is iterative, you need to choose a stopping criteria. Explain your reasoning.

Apply your function to the `Rx=Ry=2` data. Use the calibration region and kernel size from (a). Display the reconstructed coil data, the difference image multiplied by 10, and the square root of sum of squares reconstruction.

Usually the calibration region is included in the initial data. Repeat the reconstruction and see if this helps.

Solution An implementation of the spirit reconstruction is given here:

```
function [ mr msed] = spirit_recon(ma, mc, Nk, Ni)
%
% reconstruct an undersampled data set, using a calibration set mc, which
% need not be the same
%
%   ma -- k-space data
%   mc -- calibration data
%   Nk -- kernel size
%   Ni -- number of interations
%
%   mr -- reconstructed k-space data
%

% keep a mask of the known data
known_data = (ma ~= 0);
unknown_data = (ma == 0);

% perform the calibration
sk = spirit_kernel(mc, Nk);

%initialize reconstruction
```

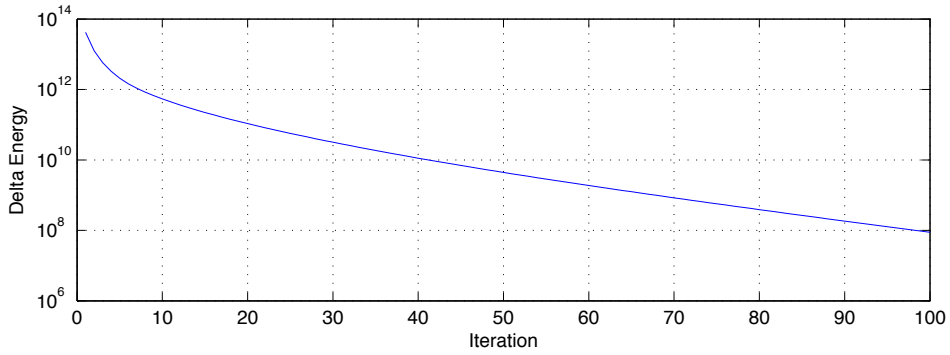


Figure 6: Energy of the difference between sequential iterations of the SPIRiT algorithm.

```
mr = ma;

msed = [];
mrp = mr;
% iterate applying kernel, and enforcing data consistency
for n =1:Ni,
    mr = apply_spirit(mr, sk);
    mr = (ma.*known_data) + mr.*unknown_data;
    msed = [msed sum(sum(sum(abs(mr-mrp).^2)))]; mrp = mr;
end
```

It iterates for a fixed number of iterations, and returns the result, as well as the energy in the difference between sequential iterations. This difference energy is plotted in Fig. 6.

This shows that the error between frames continues to drop, but at an exponentially decreasing rate. The question is then, when does this not matter any more. In this case we have the fully sampled data, so we can look at the true error, which we normally wouldn't have.

After 20 iterations there is still highly structured errors which are noticeable in the reconstruction. By 50 iterations, the errors are higher frequency, and much less noticeable. From 100 to 500 makes no noticeable difference at all. Practically, a threshold between 50 and 100 iterations would be reasonable.

Examples of the individual coil reconstructions after 100 iterations are shown in Fig. 8. The error, and the error times 10 is shown in Fig. 9.

Including the calibration region dramatically improves the convergence. The energy in the sequential frame difference decreases by a factor of 10, and the time to a given difference energy is halved. This is plotted in Fig. 10. The difference between the SPIRiT reconstruction after 100 iterations and the true data now shows even less structure, as shown in Fign 11.

c) Random Sampling Generate a randomly sampled data set with

```
% choose random phase encodes
msk1 = (rand(size(m(:,:,1))) > 0.5);
% copy to all coils
msk = repmat(msk1,[1 1 8]);
% mask the data off
mr = m.*msk;
```

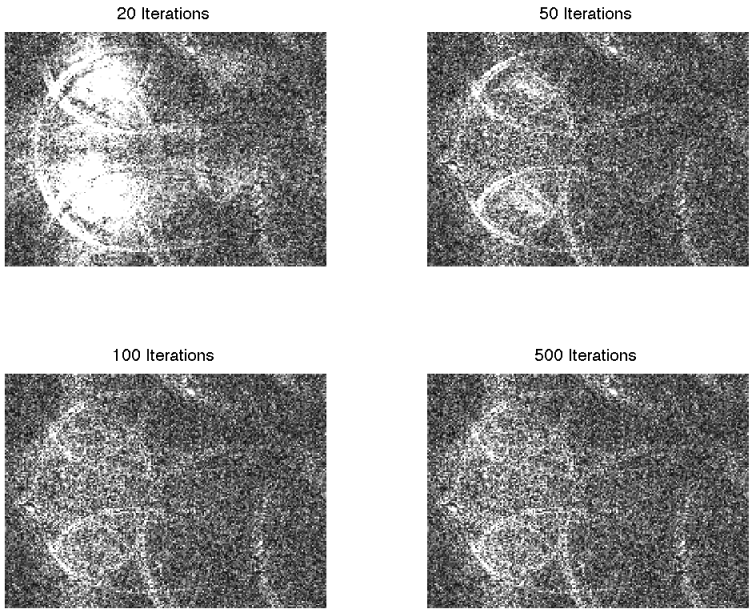


Figure 7: The error multiplied by 10 between the SPIRiT reconstruction and the true data after 20, 50, 100, and 500 iterations.

Figure 8: Individual coil images after 100 SPIRiT iterations

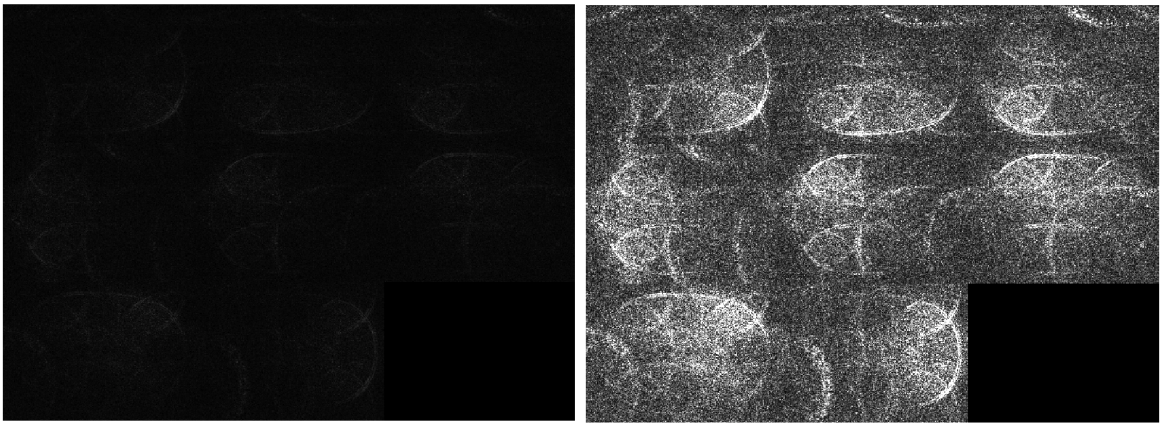


Figure 9: Error between the SPIRiT reconstruction and the true data, and the difference multiplied by 10. This is after 100 iterations.

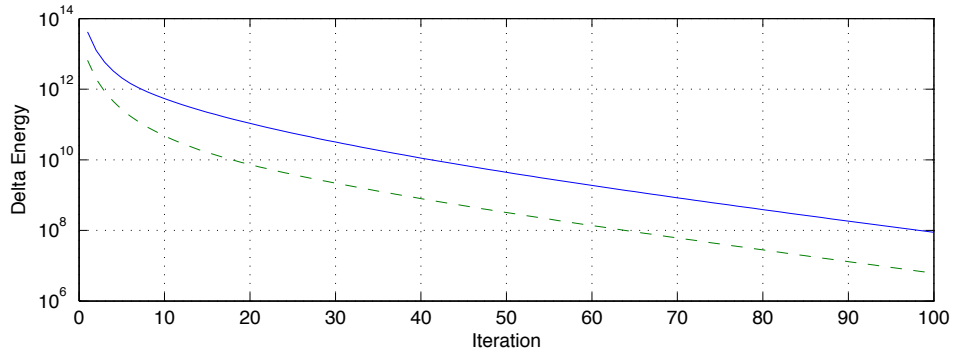


Figure 10: The energy in the difference between sequential frames for the SPIRiT reconstruction, with out and with the extra calibration data. The dashed line is for the accelerated data, while the dashed line also includes the calibration data.

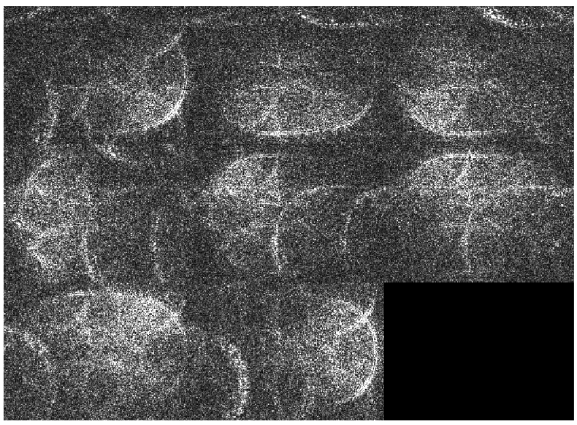
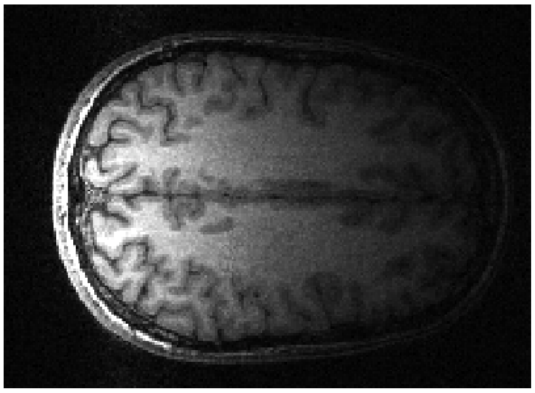


Figure 11: Error images between individual coil images after 100 SPIRiT iterations, and the true data. This includes the calibration region, as is usually the case in practice.



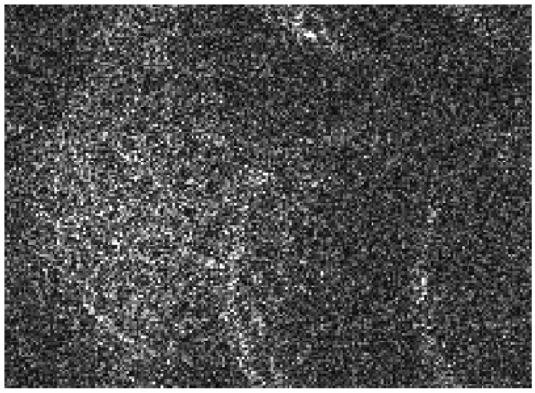


Figure 12: Reconstruction of the randomly sampled data, with a factor of 2 acceleration. The image is of coil 4. The error is multiplied by 10.

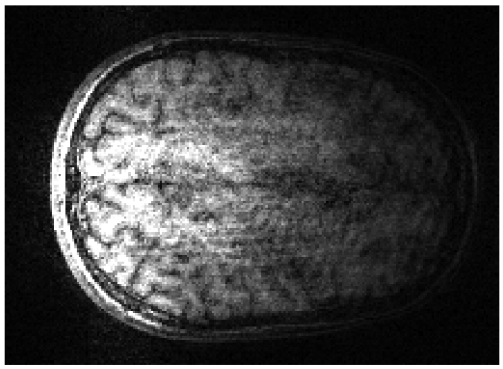


Figure 13: Reconstruction of the randomly sampled data, with a factor of 4 acceleration. The image is of coil 4.

This randomly deletes half of the samples. Reconstruct using the same calibration region as above. Display one of the coil images, and the difference multiplied by 10. This would be a difficult reconstruction to do with GRAPPA!

You can adjust the fraction of the data that is deleted by changing the threshold value from 0.5. Estimate the point at which the reconstruction become unacceptable. What happens to the number of iterations you need?

Solution

For the 50% acceleration case, the image for coil 4 and the error multiplied by 10 are shown in Fig. 12, after 50 iterations and not including the calibration data. This is an easy case. If we go to an acceleration of 4, or only 25% of the data, the reconstructions are certainly unacceptable, as shown in Fig. 13, and this is after 500 iterations.