

EE369c: Assignment 1

Due Thursday, Oct 11

Introduction

This problem set concerns partial k-space reconstruction methods. Since it is the first assignment, it also involves becoming familiar with Matlab, and displaying images. Many of you may already be familiar with this.

Displaying Images For displaying images the best option is “imshow” which is part of the image processing toolbox. If you have a complex image “im”, you display it with

```
>> imshow(abs(im), []).
```

This automatically scales the image from the image minimum to maximum. If you want to explicitly specify these, use

```
>> imshow(abs(im), [win_min win_max]).
```

This is quite common. Often the brightest part of the image is of little interest (a vessel, or fat, for instance). You will want to specify a narrower window to accentuate the part of the image that is of interest.

The built in “image” function in Matlab does a remarkably poor job of displaying images. Don’t use it if you can avoid it.

You can add a title to the plot, which is helpful to remind you what you were thinking when you generated it,

```
>> title('One cycle of phase correction in x');
```

Orientation Conventions In displaying medical images there are conventions about the orientations the images are presented. The axes are described in “subject” space. These are right-left (R/L) for subject right and left, anterior-posterior (A/P) for subject front and back, and superior-inferior (S/I) for subject up and down. For axial slices, the image is oriented as it would appear if you were looking at the subject from the feet. The subject’s left is on the right side of the image, and their front is up. For a sagittal slice (down the middle) you are viewing the subject from their left side, so up is up, the subject’s front is to the left, and back is to the right. For coronal slices you are viewing the subject from the front, and up is up, and left is right.

It is important to get the orientation right. From a medical perspective, the reason is fairly obvious. Think of someone using your images for surgical planning. Beyond that, people are conditioned to see images in a particular orientation, and have trouble understanding them if they are flipped or rotated. It is like looking at a face upside down, it is remarkably hard for your brain to make sense of it.

There are several ways to flip images in matlab. You can flip an image about the diagonal with the transpose

```
>> imt = im.';
```

Note the “.”. The default transpose (i.e., just “’”) also takes the conjugate, and that can cause problems when we care about the phase. You can flip images left-right and/or up-down by indexing the image in the reverse order

```
>> im_tb_flip = im(256:-1:1,:);
>> im_lr_flip = im(:,256:-1:1);
```

The “:” is a placeholder meaning that dimension is in the usual order. You can also rotate an image with the image processing toolbox function “imrotate” which rotates an image in the counter-clockwise direction

```
>> im_rot = imrotate(im,ang);
```

where “ang” is specified in degrees.

Once you have the image displayed, you can save it in eps format with

```
>> print -dpdf your_filename.pdf
```

This produces an eps or pdf file in the current directory. You can then splice that into your report.

Centered 2DFT’s Generally in medical image reconstruction, the origin for both the k-space data and the reconstructed image is the center of the image or data array. The usual convention for the fft is that the origin is at the beginning of the array, or the corner of a 2D array. To do a centered fft, you want to do an fftshift both before and after the fft. You will be doing this a lot in this course, so I’d recommend defining a matlab m-file that does this automatically

```
function im = fft2c(d)
% im = fft2c(d)
%
% fft2c performs a centered fft2
%
im = fftshift(fft2(fftshift(d)));
```

Save this in fft2c.m, and put it in a directory in your matlab path. Use addpath() if you need to add a directory to the path. From then on you can pretend that fft2c is actually part of matlab. The lines that start with “%” are comments that are printed when you ask for help

```
>> help fft2c
```

```
im = fft2c(d)
```

```
fft2c performs a centered fft2
```

Put something there that will remind you how to invoke the function if you forget. You will also want the corresponding ifft2c and the one dimensional versions fftc and ifftc.

Preparing Solutions The assignment is to perform a number of different reconstructions, orient them properly, save pdf versions, and mail them to me. In addition, there are a few questions to answer. Ideally you would use LaTeX to format them along with the answers (including your m-files), and email me a pdf file. Less ideally, you can use Word. Send me a pdf if you can. If neither of these is an option, you can send me a text file with pdf or jpg image attachments. Either way, mail them to pauly@stanford.edu. Please feel free to contact me if you have any problems.

Data Files

The MR data files are in

http://www.stanford.edu/class/ee369c/mr_data.

These will be used in this assignment, and others. Most of the data sets were acquired on a 0.5T interventional scanner using a head coil for transmit and receive. The files include

Spin Echo Data Sets

- **se_t1_sag_data.mat** Sagittal T1-weighted midline shot of the head of a normal volunteer. CSF is dark due to its very long T1, white matter is brighter than gray matter. Fat is very bright due to its short T1.
- **fse_t1_ax_data.mat** Axial T1-weighted shot of the head of a normal volunteer at the level of the eyes. Same contrast as the previous shot. This was acquired with a fast-spin-echo pulse sequence.
- **fse_t2_ax_data.mat** Axial T2-weighted shot of the head of a normal volunteer at the level of the eyes. Gray matter is brighter than white matter, CSF is bright due to its long T2. Fat is bright due to the suppression of J-coupling along the lipid molecules by the rapid refocusing of the fast-spin-echo pulse sequence, which lengthens the T2 significantly.

and

Gradient Echo Data Sets

- **gre_axial_phantom.mat** This is an axial T1 weighted gradient-recalled echo scan acquired at 1.5T.
- **gre_axial_head_ip.mat** This is a gradient recalled echo scan acquired at 0.5T. The echo timing has been chosen so that water and fat are in phase, at 13.8 ms. The data is in the variable `d_ip`.
- **gre_axial_head_op.mat** This is a gradient recalled echo scan acquired at 0.5T. The echo timing has been chosen so that water and fat are 180 degrees out of phase, at 20.2 ms. The data is in the variable `d_op`.

All are 256×256 data matrices. When you load these into matlab with

```
>> load se_t1_sag_data.mat
```

you will have a matlab variable `d` that contains the raw data for this slice.

For partial k-space reconstruction, the gradient recalled echo images are more interesting because there is more phase to correct. The readout direction in these data files is the first index, and the phase encode is the second. Hence, to plot the 128th phase encode,

```
>> plot(abs(d(:,128)));
```

To generate a 3/4th partial k-space data set from a full data matrix `d`, one alternative is

```
>> dp = zeros(nro,npe);  
>> dp(:,1:(npe*3/4)) = d(:,1:(npe*3/4));
```

where `nro` is the number of readout samples, and `npe` is the number of phase encodes (both 256 so far), and we have chosen the first 3/4ths of the phase encodes. Similarly, a partial echo data set is generated by

```
>> dp = zeros(nro,npe);
>> dp((nro/4+1):nro,:) = d((nro/4+1):nro,:);
```

where the first part of the echo has not been collected, which is the usual case.

Questions

1. Conventional 2DFT Reconstruction Choose one of the spin echo data sets, and answer the following questions:

1. Is the largest k-space sample at the origin? If not, which direction is it shifted? What does that imply about the acquisition?
2. Typically the magnitude is displayed. Ideally MR data should be real. Pick an image and manually phase correct it for linear terms to see if you can get all of the signal into the real channel. How many cycles of shift did you need to correct for? What remains in the imaginary channel? Send images of `abs(real(im_pc))` and `abs(imag(im_pc))`. Set the brightness of each image individually, and indicate what the ratio of intensities are between the two. To generate a linear phase correction matrix, use something like

```
>> w = [-128:127]*pi/128;
>> pcm = exp(i*w'*nx)*exp(i*w*ny);
>> im_pc = im.*pcm;
```

to correct for `nx` cycles in the readout direction and `ny` cycles in the phase encode direction. This is equivalent to shifting the origin in k-space by this many samples, but has the benefit that non-integer shifts are easily performed. Note that there is a constant (global) phase shift that also needs to be corrected.

2. Merging Filter Design Most of the partial k-space reconstruction algorithms use a weighting function $W(k_y)$, which is often called a merging filter. In this question we'll look at the effect of different choices for this function.

We want to compute the impulse response of the merging filter, $w(y)$. For simplicity, we will do this with matlab. We will assume a 9/16th k-space acquisition with 256 phase encodes. This means there are 112 unacquired phase encode, 32 symmetrically sampled phase encodes, and 112 phase encodes that are only sampled for positive spatial frequencies.

- a) In class, we talked about the impulse response of the partial Fourier acquisition. The weighting filter in this case is zero for the unacquired data, and one for the acquired data,

```
>> W = [zeros(1,112) ones(1, 32+112)];
```

The impulse response $w(y)$ is then

```
>> w = ifftc(W)
```

Plot the real and imaginary components of this function. Does it look as you expect? What about the high frequency oscillation in $Im\{w(y)\}$?

- b) One solution to the problem is (a) is to apodize $W(k_y)$ by multiplying by a window function, such as a Hamming window

```
>> Ky = [-128:127]/256;  
>> Hw = 0.54+0.46*cos(2*pi*Ky);  
>> whw = ifftc(W.*Hw);
```

Plot the real and imaginary components of this function. Does it look as you expect?

- c) Now, consider the step merging filter

```
>> Ws = [zeros(1,112) ones(1,32) 2*ones(1,112)];
```

Plot the real and imaginary components of the impulse response for this k-space weighting. Does it look as you expect? How many samples do you need to go from the origin in image space before the magnitude of $w(y)$ remains below 1% of the peak?

- d) Now, consider a ramp filter over the same ranges, and plot the real and imaginary components of the impulse response. How does it compare to the step weighting above? How many samples do you need to go from the origin in image space before the magnitude of $w(y)$ remains below 1% of the peak?
- e) Come up with your own better weighting function. Plot the function and it's impulse response. How does it compare to the step and ramp weighting?
- f) In practice, we don't use the apodization function. Should we?

3. Partial k-Space Reconstruction Program each of the three partial k-space algorithms. Each should require no more than 20 lines of matlab code. Test these on the phantom data set (`gre_axial_phantom.mat`). This is a high contrast object with sharp edges, which makes artifacts easier to detect. It also has some phase variation, especially around the bubble at the top and the "GE" logo. Submit 9/16th k-space reconstructions (zero out 7/16ths of the data file) for each method, and the difference images compared to the full k-space reconstructions, `abs(im_fk)-abs(im_pk)`. The partial k-space methods are

1. Homodyne, using the weighting (merging filter) of your choice.
2. POCS. Monitor the convergence of the POCS algorithm by computing the mean squared error between iterations. State the termination you used, and how many iterations were needed.
3. Phase corrected conjugate synthesis using either direct conjugate synthesis, or the homodyne approach for implicitly synthesizing the missing conjugate data.