EE 591: Magnetic Resonance Imaging and Reconstruction Fall 2009
K. Nayak

Homework #9

due Tuesday, December 1st, 2009

Reading:

- Pauly J., "Reconstruction of Non-Cartesian Data", *(preliminary draft of a book chapter)*.

Assignment:

This assignment introduces the basic operations in gridding reconstruction. Starting from a simple routine, you will add pre-weighting density correction, k-space oversampling, and deapodization. The dataset from this problem is a simulated phantom using a spiral acquisition with 6 interleaves of 1536 samples. The k-space trajectory, pre-weighting function, and simulated data are in the file: http://mrel.usc.edu/class/591/data/rt_spiral.mat

The complex k-space data is in the matlab variable `d`, and the pre-weighting function is in the matlab variable `w`. The k-space trajectory (k_x,k_y) is stored as $k = k_x + ik_y$ in the complex matlab variable `k`. `k` is scaled relative to ±0.5. The k-space trajectory is scaled to produce the correct field-of-view when reconstructed as a 128x128 image.

We start with a very basic gridding algorithm that doesn't do density correction, uses a simple separable triangular kernel, uses a 1X grid, and doesn't do any deapodization. This m-file is available as `grid1.m`. Note that this only does the gridding, and you still need to do an inverse 2DFFT to produce an image.

```
function m = grid1(d,k,n)

% function m = grid1(d,k,n)
%     d -- k-space data
%     k -- k-trajectory, scaled -0.5 to 0.5
%     n -- image size

% convert to single column
d = d(:);
k = k(:);

% convert k-space samples to matrix indices
nx = (n/2+1) + n*real(k);
ny = (n/2+1) + n*imag(k);

% zero out output array
m = zeros(n,n);

% loop over samples in kernel
for lx = -1:1,
  for ly = -1:1,

    % find nearest samples
    nxt = round(nx+lx);
```

```
    nyt = round(ny+ly);

    % compute weighting for triangular kernel
    kwx = max(1-abs(nx-nxt),0);
    kwy = max(1-abs(ny-nyt),0);

    % map samples outside the matrix to the edges
    nxt = max(nxt,1); nxt = min(nxt,n);
    nyt = max(nyt,1); nyt = min(nyt,n);

    % use sparse matrix to turn k-space trajectory into 2D matrix
    m = m+sparse(nxt,nyt,d.*kwx.*kwy,n,n);
  end;
end;

% zero out edge samples, since these may be due to samples outside
% the matrix
m(:,1) = 0; m(:,n) = 0;
m(1,:) = 0; m(n,:) = 0;
```

The function `grid1.m` loops over the gridding kernel, which is relatively small (-1 to +1). For each sample of the kernel, the gridding operation for the entire data vector is done with the `sparse()` matrix call. This sets up an n × n sparse matrix with values `d.*kwx.*kwy` at matrix locations `(nx,ny)`. This automatically gets converted to a full 2D matrix when it is added to `m`.

This is one possible way to code the gridding algorithm, and happens to be exceptionally fast in Matlab. If you find a faster approach, let us know!

1. **Simple Gridding Reconstruction**. Reconstruct an 128 × 128 image of the simulated phantom. There is a dominant low frequency artifact. What is it due to? Display your reconstruction.

2. **Density Pre-Compensation.** Extend the algorithm to use the preweighting function `w` that has been provided. Display your reconstruction.

3. **Oversampling**. Extend the algorithm to reconstruct on a 2X grid. This is a grid that is sampled twice as finely in k-space, and has twice the FOV in image space. The kernel should now extend for ±2 samples on the 2X grid. Display the 2X reconstruction. What artifacts have been reduced or eliminated.

4. **Deapodization**. The kernel we are using is a separable triangle function in k_x and k_y. Compute the apodization produced by this kernel for the 2X oversampled reconstruction, and divide it out of the reconstructed image. Plot a cross-section through the phantom before and after correction. Display your corrected reconstruction.

At this point, you should have an algorithm that does a reasonable job of gridding!

You may earn **Extra Credit** by going beyond the call of duty on any of these parts. Feel free to suggest ideas to me, and I will approve them over e-mail. For example, you could: 1) improve the convolution kernel, or 2) compare different methods for density compensation.