UNIVERSITY OF CALIFORNIA, SANTA BARBARA
Department of Electrical and Computer Engineering

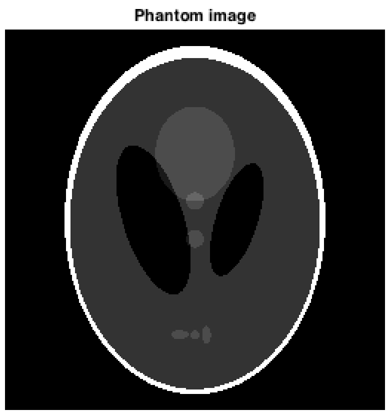
Prof. P. Sen **ECE 178 Digital Image Processing** Fall 2017

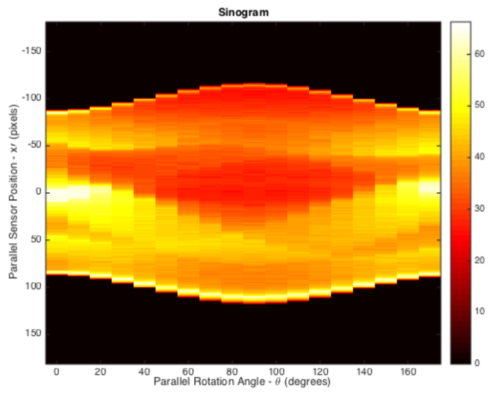
Homework 7

Due Tuesday, November 21. No Written portion. MATLAB portion due on GauchoSpace at 9:00pm.

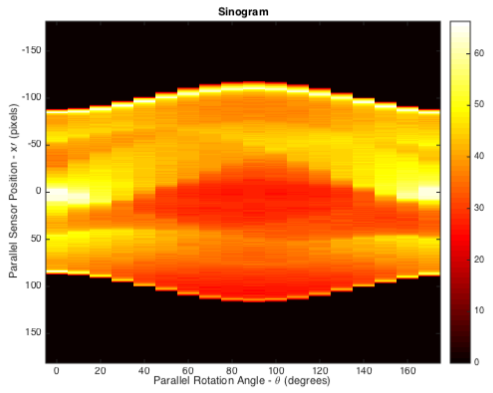
In this homework, we are going to implement Radon transform and filtered back projection.

1. **[5 pts]** In this part, one phantom image is given and we are going to do the Radon transform on this image. You have to implement *myRadon.m* function, which takes two input variables *img* (the input phantom image) and *theta_all* (angles for projection). It returns a $N \times K$ sinogram of Radon transform, where N is the length of diagonal line of the input image (in our case $ceil(sqrt(256^2 + 256^2)) = 363$) and K is the number of angles for the projection. After the transform is done, the main function will show the sinogram as following:



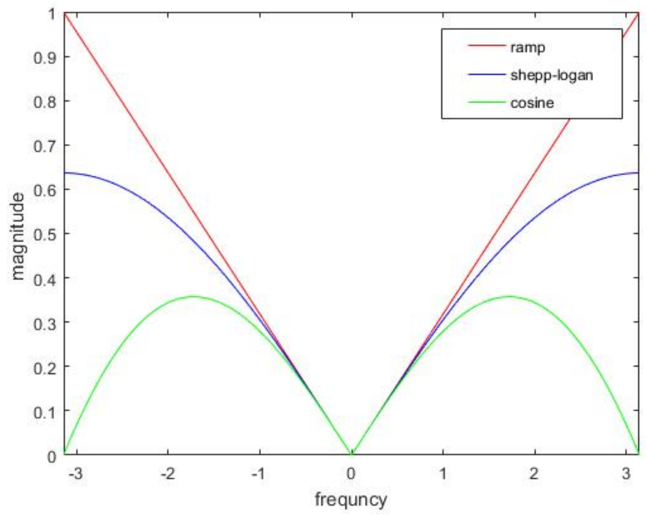


If you get a sinogram like this:



You have to take a look at the angle used for rotation, one negative sign may be needed. If you don't fix it at this part, you may get an upside down reconstructed image at the second part.

2. **[5 pts]** In the second part, we are going to implement the filtered back projection for Radon transform. You have to finish the blank part in *myInvRadon.m* function, which takes three input variables p (Radon transform results), *theta_all* (angles for projection), and *filter* (type of filter) and outputs the reconstructed result. Three types of filter are given for frequency domain filtering: *Ramp/Ram-Lak*, *Shepp-logan*, and *Cosine*. These three filters have frequency responses as shown the following figure.



The formula to implement these filters in the frequency domain is shown below:

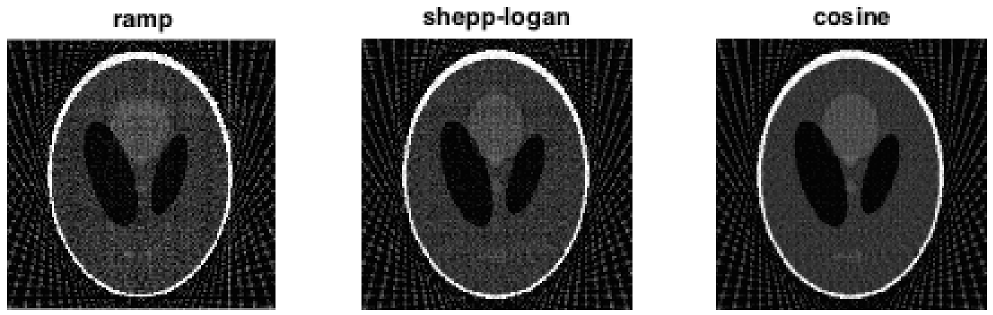
- *Ramp:* $f_r(\omega_i) \;=\; \frac{1}{\pi}|\omega_i|$ where, $\omega_i = -\pi + \frac{2\,\pi\,i}{N-1}$ for $i = 0 : N-1$
- *Shepp-logan:* $f_s(\omega_i) \;=\; f_r(\omega_i)\; sinc(\frac{\omega_i}{2\pi})$ where, $\omega_i = -\pi + \frac{2\,\pi\,i}{N-1}$ for $i = 0 : N-1$
- *Cosine:* $f_c(\omega_i) \;=\; f_r(\omega_i)\; cos(\frac{\omega_i}{2})$ where, $\omega_i = -\pi + \frac{2\,\pi\,i}{N-1}$ for $i = 0 : N-1$

You need to do the following to get filtered back-projection result:

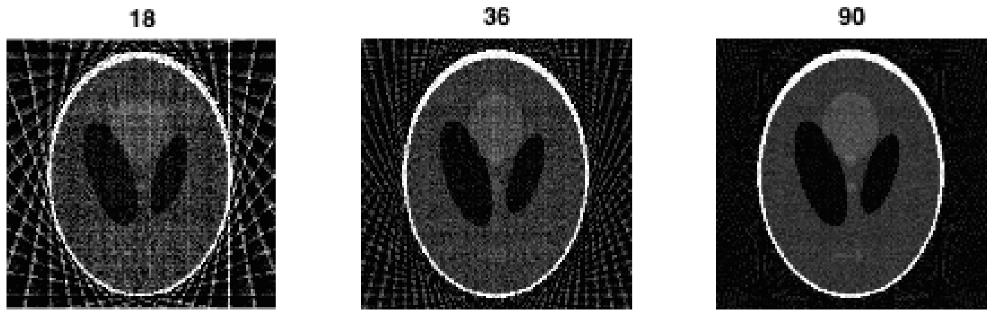
(a) Do the 1D filtering in frequency domain for each column of the output of Radon Transform. To do this, take FFT of the 1D signal and multiply with the filter implemented using the equations shown above. Note that you need to shift the filter before multiplying to make sure that you are multiplying correct frequency components. To do the shifting of filter, MATLAB's built-in *ifftshit* could be useful. After multiplying in the frequency domain, perform IFFT to get the filtered lines.

(b) Use the filtered lines for back projection to get the original phantom image.

To debug your back-projection code, maybe first try to implement it without using any filtering to get a reconstruction. After *myInvRadon.m* is done, the main function will do two comparisons for you. The first one is the comparison of different filters given the same number of angles for Radon transform.



We can see that the later two filters work better than ramp filter. The next comparison is to show the back-projected images of different number of angles by the smae filter (we use *ramp* at here).



As we increase the number of angles, the quality of the reconstructed image is getting better.

Note that built-in functions *radon* and *iradon* are **NOT ALLOWED TO USE**. For submitting your code, please upload one zipped file includes only 2 files : *myRadon.m*, *myInvRadon.m*. Name your zip file in this format: <Perm number>_<First name>_<Last name>_HW7. Good luck!