Mathematics 372 – Numerical Linear Algebra
Lab/Problem Set 11 – Landweber Iteration and Computed Tomography
April 23, 25, 27, 2007

Background

We are now ready to work with one iterative method that applies for overdetermined systems and that can be used for the image reconstruction problems from mathematical SPECT tomography. Recall that the image acquisition process can be modeled by the following equations. Let $\mathbf{f}$ represent the true concentrations of the radioactive tracer in a 2-dimensional axial slice of the subject (e.g. a patient's torso). We take a discretization ("pixelization") of this slice, so f will be a real vector with J components for some J (f can be thought of as either a single, "long," vector, or as a 2-dimensional array of pixel values which can be plotted as an image).

The process of data acquisition with multiple angular positions of the radiation detector camera is modeled by the matrix-vector equation

$$\mathbf{g} = H\mathbf{f} \tag{1}$$

where $\mathbf{g}$ is a (known) vector with I components that represents the measured data by the detector, and the *projection matrix* H is $I \times J$. $\mathbf{f}$ is the unknown here – we want to solve for it to reconstruct the image from the projection data $\mathbf{g}$. Typically in the applications, $I > J$, so the system above is *overdetermined*.

The problem is to find the "best fit" f for the data g, in the *least squares sense*, since (1) typically has no exact solution $\mathbf{f}$. Of course systems like this could be solved in theory either by solving the normal equations

$$(H^t H)\mathbf{f} = H^t\mathbf{g} \tag{2}$$

directly, or by using a QR-factorization, or the SVD of H, as we have discussed. However, in realistic applications, I and J are quite large (in the 1000's), so these approaches can be infeasible. Indeed, the MATLAB programs we will use never actually write down the whole *projection matrix* H or the whole *back-projection* matrix H^t explicitly.

Instead, we will use an iterative method called *Landweber iteration*. Given $\mathbf{f}^{(0)}$, we iterate

$$\mathbf{f}^{(k+1)} = \mathbf{f}^{(k)} + \alpha H^t\left(\mathbf{g} - H\mathbf{f}^{(k)}\right), \quad k \geq 0 \tag{3}$$

for some appropriate choice of the parameter $0 < \alpha < 1$, called the *relaxation parameter*. Note that (3) can also be written in the standard fixed-point form as follows:

$$\mathbf{f}^{(k+1)} = (I - \alpha H^t H)\mathbf{f}^{(k)} + \alpha H^t\mathbf{g}. \tag{4}$$

The equation (3) shows that Landweber can can be thought of as an *iterative technique for solving the normal equations* (2), since if $H^t\mathbf{g} = H^t H\mathbf{f}^{(k)}$, then $\mathbf{f}^{(k+1)} = \mathbf{f}^{(k)}$, and we are at a fixed point.

We will be using a series of MATLAB programs written by Prof. Soares of our department and a research student of his last year. To obtain copies of them, enter the following Linux commands in a Terminal window:

```
mkdir Tomo
cp /home/fac/little/public_html/NLA07/Tomo/* Tomo
cd Tomo
matlab &
```

The first line creates a new folder in your directory, the second copies the program files we will be using, the third changes your current directory to the new folder, and the last launches MATLAB.

Note: To be safe, always get into MATLAB this way when working on this lab assignment.

In the MATLAB command window, enter

```
Landweber_demo
```

This should launch a window containing three plots called Original Image, Projection Data, and Reconstructed Data.

- The default image shown on the left is a "pixelized" circular disk.
- There are several choices for the color map used in displaying images – I like "Bone" the best personally, but you can experiment to find one that "works for you."
- In the left portion, you also get to specify a number of projection angles, and the way the images are displayed. Experiment with these to see what they mean.
- The Project button on the left does the projection step

$$\mathbf{f} \mapsto \mathbf{g} = H\mathbf{f}.$$

  The resulting image can be hard to interpret, so don't spend a lot of time trying to understand it – just realize that the process used to create it is a version of the data acquisition process we discussed – multiple angular positions of the detector camera.
- The center inputs let you specify the relaxation parameter α for Landweber iteration directly. Or you can have the program estimate the maximum α for which the iteration will converge for you. If you press the Find Max Alpha button, you will need to *wait* for the output (this computation can take a while, depending on the size of the image, so be patient).
- You set the Maximum Iterations number by typing in a value. Try the value Maximum Iterations = 3 to start.
- We will not be using the Counts feature at all – just leave it with the default value of 100000.

- The Add Noise button adds random "noise" to the projection data – this is a fundamental aspect of using these techniques in the real world. Gamma rays can be deflected by other structures in the body, etc. and that can introduce this kind of noise in the data collected by the detector. You can press this button repeatedly to add "lots of noise."
- When you press the Reconstruct button, the Landweber iteration will be started and a message will be printed out in the MATLAB Command Window when each iteration is begun. The reconstructed image is displayed on the right.

The default image is contained in a file called

`disk_p32p32.mat`

which is one of the files you copied. There are several other files you should look at as well:

`gaussian_p64p64.mat`
`gaussian_offcenter_p64p64.mat`
`mcat_p64p64.mat`

The last of these is a "mathematical cardiac torso phantom" – an artificial image representing a slice through a human torso showing the chambers of the heart, spine, ribs, etc. You can use the Load Image button on the left to get these. (If you launch MATLAB from the directory containing these, they should show up in the Pick a File dialog box.) Experiment with all of these and try to get a good intuitive feeling for how the reconstruction quality is affected by things like the value of α, the maximum number of iterations used, whether noise is added, what "chopping negatives" means, and so forth.

Theoretical and "Conceptual" Questions

A) If you fix α, does increasing the number of iterations *always* improve the reconstructed image quality? If so, why? If not, what else does this depend on (give a specific example where it does not improve things and explain what is "going wrong").
B) If you fix the maximum number of iterations, how does the value of α used seem to affect the reconstructed image quality?
C) Should the maximum value of α computed by the program depend on *which image* you have loaded? If not, what *does it depend on*? (Note: the maximum α is computed by a program called `powermethod_LW.m` that is part of the "suite" of programs you copied. You can examine the source code with an editor if you like.)
D) In fact, prove using our analysis of convergence of iterative methods that Landweber iteration will converge for all α satisfying

$$0 < \alpha < \frac{2}{\sigma_1^2},$$

where σ_1 is the largest singular value of the projection matrix H. (Since we usually do not *have* the actual projection matrix H, this is primarily a theoretical result, not a way to determine what value of α to use.) Can you see what the `powermethod_LW.m` function is actually doing now?

E) You might have noticed that if we had the exact original object $\mathbf{f}$ and the exact projection data $\mathbf{g} = H\mathbf{f}$, then the system $H\mathbf{x} = \mathbf{g}$, while overdetermined, would have at least one *exact* solution $\mathbf{x} = \mathbf{f}$. So, strictly speaking, least squares is unnecessary. However in the real world, there *will* be noise in the projected data. We model the noise as an additional term added to $\mathbf{g}$, so we are actually looking at $H\mathbf{x} = \mathbf{g} + \delta\mathbf{g}$. Why will least squares usually actually be necessary in this sort of problem?

F) What is "chopping negatives" and why is it reasonable to do that in looking at the reconstructed image? (It may help

G) Since the projection data $\mathbf{g}$ and the final reconstructed image $\mathbf{f}^{(maxiter)}$ can be seen as "long" vectors, one way to measure the *quality* of the reconstructed image would be to use a vector norm and compute

$$\|\mathbf{g} - H\mathbf{f}^{(maxiter)}\|.$$

How would this idea depend on which vector norm we used? Which norm do you expect would be closest to saying "the reconstructed image looks like the original data" intuitively? For instance what would it mean if $\|\mathbf{g} - H\mathbf{f}^{(maxiter)}\|_\infty$ was small? Similarly, what would it mean if $\|\mathbf{g} - H\mathbf{f}^{(maxiter)}\|_1$, or $\|\mathbf{g} - H\mathbf{f}^{(maxiter)}\|_2$ was small. Which norm is Landweber iteration set up for?

April 25, 27 – Further MATLAB investigations

A) A "normalized" version of the measure of reconstructed image quality from conceptual question G above is called the *root mean-square error* (RMS). Let

$$\hat{\mathbf{g}} = H\mathbf{f}^{(maxiter)}$$

be the *estimated* projection data based on the final reconstructed image. Then we define:

$$RMS_{data} = \sqrt{\frac{1}{I}\sum_{i=1}^{I}(g_i - \hat{g}_i)^2} = \frac{1}{\sqrt{I}}\|\mathbf{g} - \hat{\mathbf{g}}\|_2.$$

To study this in more detail, we can use another one of the programs you copied at the start of the lab: `LWBRMSiter.m`.

To run this you enter a command like this from the MATLAB command window:

```
LWBRMSiter('mcat_p64p64.mat',20)
```

The first argument is the filename for the image you want, the second is the maximum number of iterations. The output will be a plot showing how RMS_{data} changes as

maxiter increases from 1 to 20. That is, the 20 in the input command is the "maximum *maxiter*." The value of α returned by `powermethod_LW.m` is used here. Generate plots for each of the 4 image files above and max *maxiter* 20. What is happening in each case?

B) (This is a somewhat open-ended question.) If you fix *maxiter* to begin with, what is the best strategy for choosing α to give the smallest RMS_{data} (intuitively, this would be the α with the fastest convergence rate)? You may assume if you like that H^tH is invertible. You can attack this either theoretically or experimentally. For the experimental way, for instance, you might create a modified version of the function from `LWBRMSiter.m` (call it something else) that lets you supply a value for α in addition to a *maxiter* and use it to gather data about the question. Note: you will have to think about how to set up the output to help you answer this. The output in the function supplied is geared to displaying a plot for RMS_{data} after 1 iteration, after 2 iterations, and so on up to a max *maxiter* number of iterations.

Note

This is the last regular lab/problem set for the semester. It will be due in class on Monday, April 30. In the course syllabus I said that the final lab project would count for 15% in the 40% of the course grade based on the labs and problem sets. But since we actually had two more regular problem sets than I originally planned (including the last two on eigenvalue and iterative methods), I am not going to do it that way. This will just be an "ordinary" problem set and it will carry the same weight as the others.