

Homework 4 (DRAFT)

David L. Olson

April 4, 2025

Contents

1	Problem 1	2
1.1	Solution	2
1.1.1	Part A	2
1.1.2	Part B	3
2	Problem 2	7
2.1	Solution	7

1 Problem 1

1.1 Solution

1.1.1 Part A

The `landweber()` function was implemented in MATLAB[®], and a screenshot of the code is provided in figure 1.

```

1 function X = landweber(G, m0, d, omega, k)
2 % Landweber Iteration for Steepest Decent Method
3 %
4 % Performs "k" iterations of the Landweber iteration on Gm = d, with
5 % initial solution m0, and parameter omega.
6 %
7 % On output, X is a matrix of size n by k. The k columns of X are the
8 % iterates m1, m2, ..., mk.
9 %
10 % For convergence, we need 0 < omega < 2/(s(1)^2). Outside of this range, you
11 % can expect the method to diverge.
12
13 %% Input and Output Arguments
14
15 arguments (Input)
16 G (:,:) {isfloat, mustBeReal, mustBeFinite}
17 m0 (:,1) {isfloat, mustBeReal, mustBeFinite}
18 d (:,1) {isfloat, mustBeReal, mustBeFinite}
19 omega (1,1) {isfloat, mustBeReal, mustBeFinite, mustBePositive}
20 k (1,1) {mustBeInteger, mustBePositive}
21 end
22
23 arguments (Output)
24 X (:,:) {isfloat, mustBeReal, mustBeFinite}
25 end
26
27
28 %% Error Checking
29
30 [m, n] = size(G);
31 assert(...
32     n == length(m0), ...
33     "Initial model m0 must have the same number of columns as G!");
34 assert(...
35     m == length(d), ...
36     "Data vector d must have the same number of rows as G!");
37
38 assert(...
39     omega < (2/(svds(G,1)^2)), ...
40     "This omega value will not let the iterations converge to a solution!")
41
42
43 %% Implementation
44
45 X = zeros(n, k);
46 X(:,1) = m0;
47 for ii = 2 : k
48     X(:,ii) = X(:,ii-1) - omega * G.' * (G*X(:,ii-1) - d);
49 end
50
51
52 end
53
54

```

Figure 1: MATLAB[®] Landweber Function Screenshot

1.1.2 Part B

MATLAB[®] code was modified from `ex_6_2.m` to load image data, apply noise, and plot the resulting images. The fixed step size

$$\omega = 0.95 \frac{2}{s_1^2} = 6190.5$$

was computed using the `svds()` function in MATLAB[®]. A total of 500 iterations were running using the `landweber()` function shown in figure 1. Iterations 10, 50, 100, 150, 250, and 500 are evaluated.

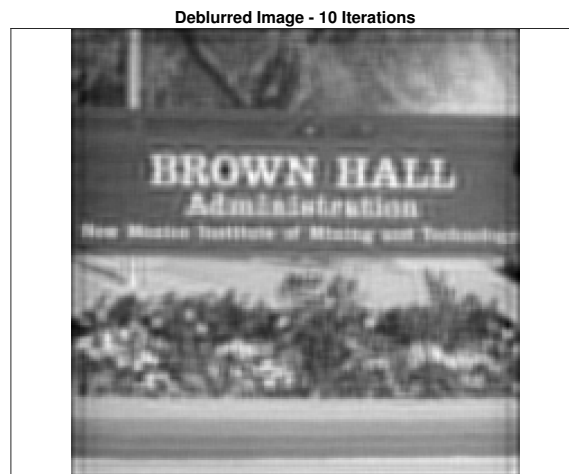


Figure 2: Deblurred Image - 10 Iterations

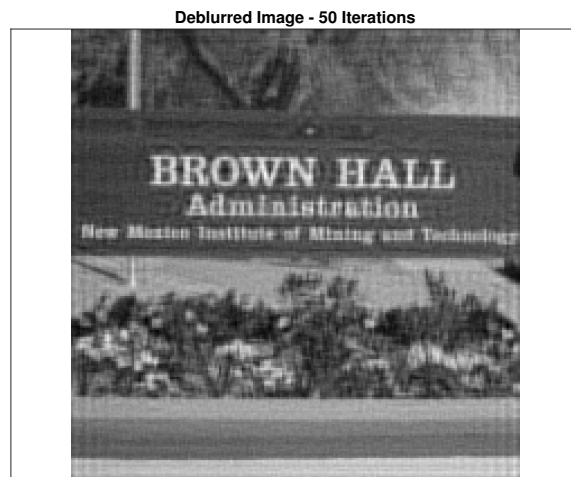


Figure 3: Deblurred Image - 50 Iterations

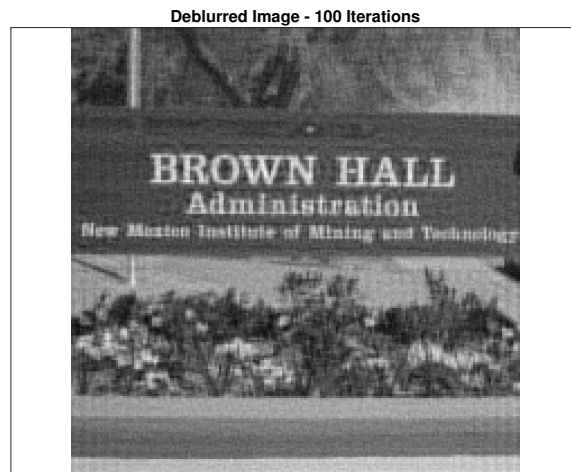


Figure 4: Deblurred Image - 100 Iterations

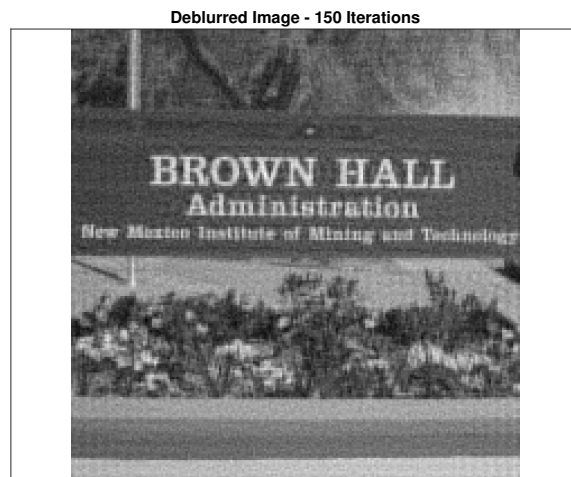


Figure 5: Deblurred Image - 150 Iterations

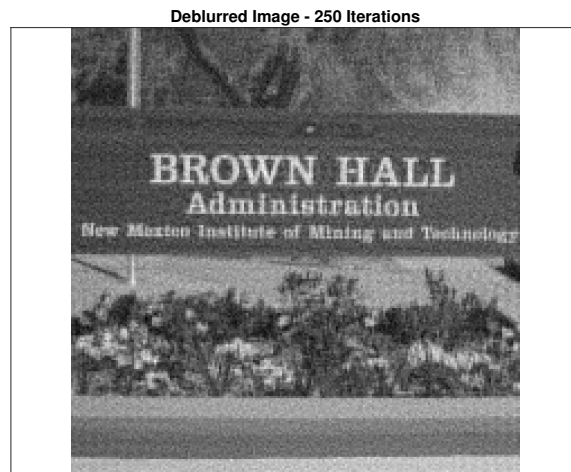


Figure 6: Deblurred Image - 250 Iterations

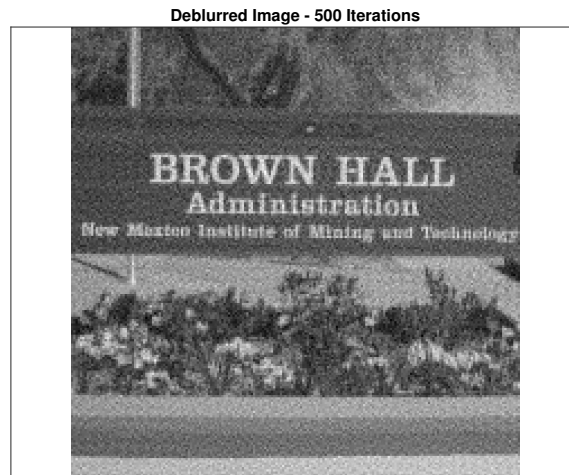


Figure 7: Deblurred Image - 500 Iterations

It becomes clear that there is a "diminishing rate of returns" on the number of iterations used to deblur the image. In my opinion, it seems that the sharpest image appears at 150 iterations. The images after this number of iterations are also as sharp, but from there I can no longer tell a difference.

It is less clear in this PDF, but on my screen in MATLAB it shows just a bit clearer. Unfortunately, you'll just have to take my word for it.

2 Problem 2

Exercise 6 in Section 7.8

2.1 Solution