Brandon Walters

CS 3200

Prof. Berzins

Homework 6 Report

1. For the given Hilbert matrices and the methods used to solve for them,
    a. We can start by looking at the accuracy of our \ solve method compared to the original x vector of 1s. As we can see from our first table, the 2 norm of this error depends on the size of the Hilbert matrix being used, with the norms starting at 5.29e-12 for the 5x5 matrix and getting as high as 216 for the 25x25 Hilbert matrix. As we can see from these norms, this method is quite effective on smaller Hilbert matrices, but it begins to struggle more as the size of the Hilbert matrix climbs.
    b. When using SVD to create the Hilbert matrix's inverse, we can see from the table that our error norms are quite close to those generated by the first \ solve method we used, although the method is generally slightly worse and has a large error spike on the 25x25 Hilbert matrix. Overall, we can conclude that using the SVD method gives us about the same level of accuracy as the initial \ solve method.
    c.

	Hilbert Matrix Size	\ Method Norm	SVD Inverse Norm
1	5	5.2949e-12	4.4296e-11
2	10	8.3663e-04	8.1744e-04
3	15	23.7834	25.7669
4	20	179.7727	287.0655
5	25	216.1554	2.2462e+03
6	30	42.9580	5.7889

d. When filtering the S matrix based on different tolerances to zero out parts of the S and S-inverse matrices, we can then use the filtered inverse with the b vector to see that we get accuracies starting generally as low as 1e-6 and going up to around 5 in the worst case. Overall, we can see that the trend is that with a higher tolerance value our norms go up, which makes sense as we are filtering out less and less of the matrices as the tolerance goes up, meaning that the preservation of values leads to a more ill-conditioned matrix. I would argue that this method is a good idea for larger Hilbert matrices, as we lose some of the initial accuracy afforded by the \ solve or regular SVD inverse calculation, but the ceiling on the norms is much lower than those aforementioned methods, meaning that it will be a big improvement on matrices where the other methods struggle, like the 25x25 and 30x30 Hilbert matrices.

e.

	Tolerances	5x5 Hilbert Matrix	10x10 Hilbert Matrix	15x15 Hilbert Matrix	20x20 Hilbert Matrix	25x25 Hilbert Matrix	30x30 Hilbert Matrix
1	1.0000e-10	4.4296e-11	1.7232e-05	2.8447e-05	2.1738e-05	1.2839e-05	2.9659e-05
2	1.0000e-12	4.4296e-11	7.3737e-06	4.7017e-06	1.0534e-05	3.8963e-05	8.7014e-05
3	1.0000e-14	4.4296e-11	8.1744e-04	0.0080	0.0065	0.0016	0.0036
4	1.0000e-16	4.4296e-11	8.1744e-04	0.3035	0.2725	0.4525	5.7889

f. Finally, we can reconstruct the 5x5 and 10x10 Hilbert matrices using the filtered version of S to see how close these filtered versions can come to the original S's Hilbert matrix. As we can see from the printed comparisons between the Hilbert matrices, the matrices produced by the filtered versions are very close to the original matrices, which fits in with our understanding of these matrices, as most of the entries in the S matrix will be too large to be filtered, therefore meaning that many of the entries are the same between the filtered and unfiltered versions. Our norms tend to

become smaller as our filter becomes more precise, which also makes sense as more of the matrix being preserved means that the differences between the two are small, and therefore the norm becomes less and less, generally from around e-11 towards e-16 as the tolerance becomes smaller.

2. For the two images given,
    a. I found that the bunny image became very highly compressed around mode 50, which was the last point where I could personally read the little sign. For this image, we know that the original size is equal to 3 * (768 * 1024), which comes out to 2,359,296 bytes, or approximately 2.3 MB. With our SVD decomposition, we can calculate our compressed size as 3 * 50 * (768 + 1024), which is equal to 268,800 bytes, or approximately 0.27 MB or 268.8 KB. As we can see, this compression reduces the size of the image by about a factor of 9, which is a huge improvement! Furthermore, this means that we saved about 2.03 MB of storage on this

image, which is a huge amount of the original image.

b. For me personally, I thought that it was important to save the subtle color contrasts and smaller stars present in the image (we want the viewer to be awed by the majesty of space, after all!). I found that the lowest mode I was personally comfortable with in regards to this aim was mode 150. We can calculate our original image's size as 3 * (630 * 567), which is equal to 1,071,630 bytes, which is equal to 1.07 MB. Our compressed image's size is 3 * 150 * (630 + 567), which comes out to 538,650 bytes, 538.6 KB, or 0.54 MB. This means that our compression saves a total of 0.53 MB, just

under half of the original image size.