Dan Ruley

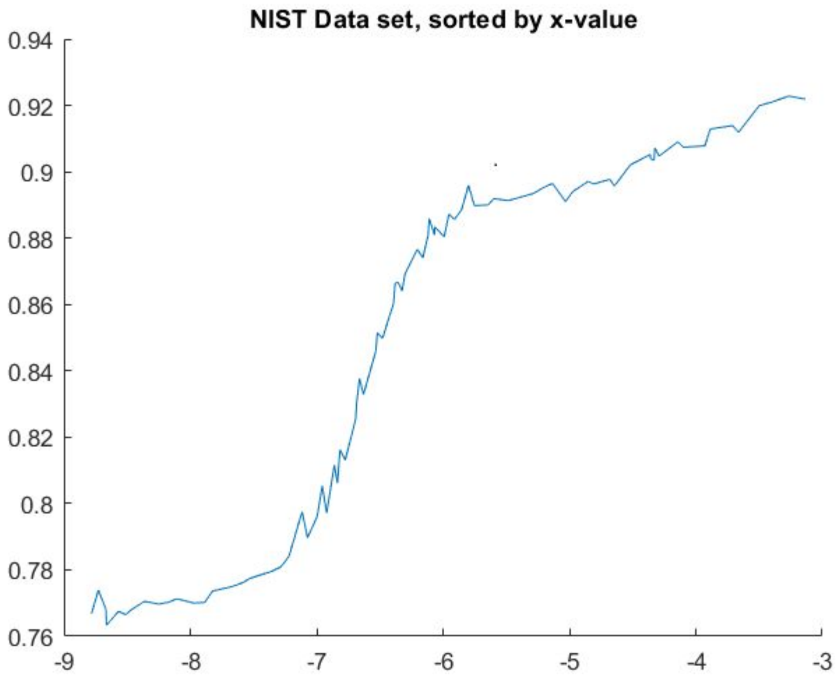
CS 3200

Assignment #6

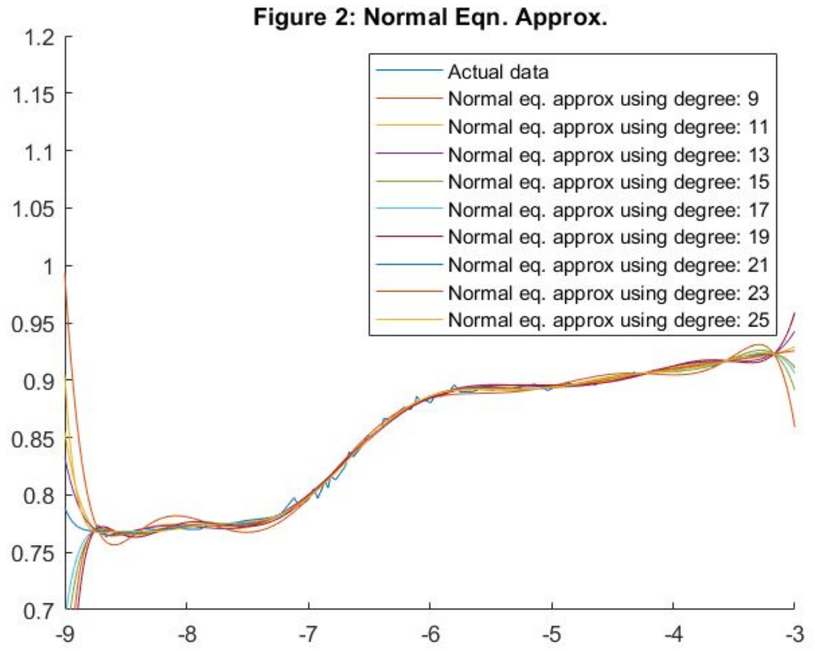
May, 2020

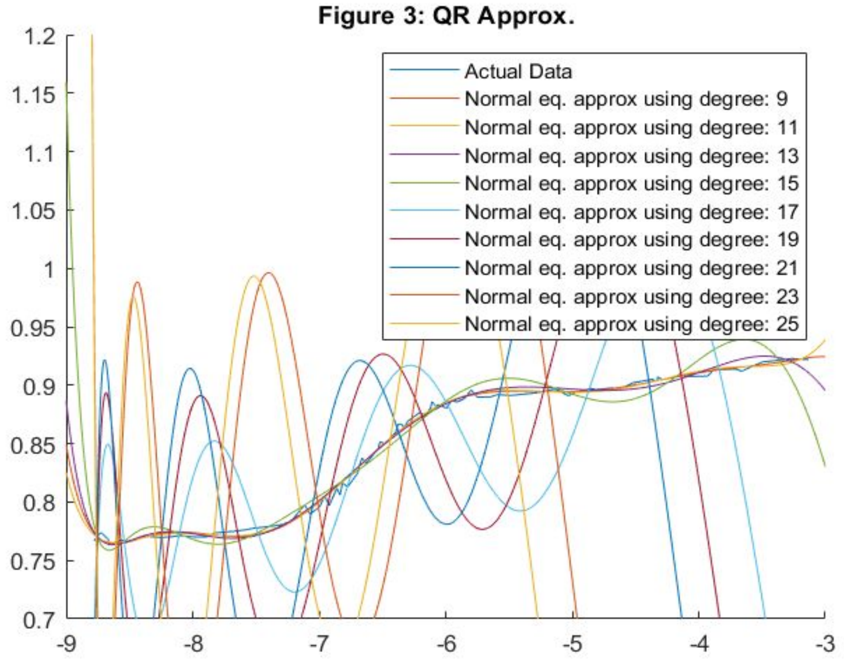
Problem 1:

a. I made a small bit of Matlab code to read in the data set, add it to a matrix, and sort it by the x-values. Then, I plotted the data:



b. I modified the program as suggested to use the Vandermond matrix and either the Normal Equations or QR method to approximate the NIST dataset. I used n = 82, polynomials of degree {9,11,13,15,17,19,21,23,25}, and 1001 points at which to evaluate the Vandermonde polynomials. Please see Figures 2 and 3 for these results:





We can see that as Dr. Berzins suggested, the QR approximations begin to get quite out of hand as the degree increases. Beyond degree 15 or so, the results seem to be extremely inaccurate.

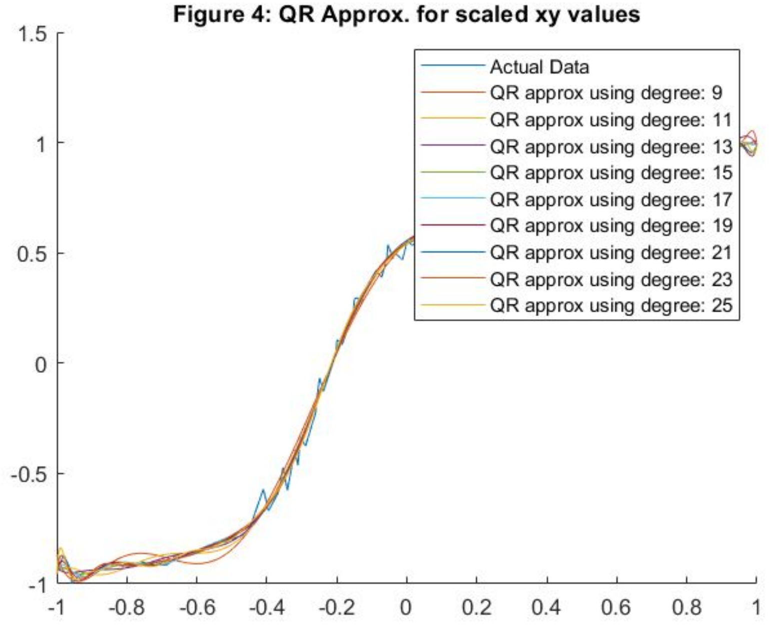
c. I calculated the least squares errors using the suggested formula. The results are exactly what one would expect from glancing at the graphs; theLSE for the Normal Equations and lower degree QR methods are fairly low, while the LSE for the higher degree QR approximations are much higher.

Table 1: Least Squares Error for Normal Equation / QR method

NORMAL METHOD ERRORS:	QR ERROR VALUES:
degree: 9 LSQError: 0.002649	degree: 9 LSQError: 0.001022
degree: 11 LSQError: 0.001083	degree: 11 LSQError: 0.001049
degree: 13 LSQError: 0.001115	degree: 13 LSQError: 0.001882
degree: 15 LSQError: 0.000782	degree: 15 LSQError: 0.009426
degree: 17 LSQError: 0.000725	degree: 17 LSQError: 0.733021
degree: 19 LSQError: 0.001273	degree: 19 LSQError: 2.149043
degree: 21 LSQError: 0.000681	degree: 21 LSQError: 4.055274
degree: 23 LSQError: 0.000809	degree: 23 LSQError: 11.678091
degree: 25 LSQError: 0.000671	degree: 25 LSQError: 14.649531

d. I scaled the x and y values as suggested and reran the computations from part b. As Dr. Berzins said, I also found that scaling the data results in a much more accurate QR approximation for higher degree polynomials. Please see Figure 4 and compare it to Figure 3 - just as a glance it is obvious that the scaled QR values are much more accurate than before, especially for higher degree polynomials. The large oscillations are completely eliminated and the QR approximation matches the data quite closely.

Figure 4: QR Approx. for scaled xy values



e. It seems like using the scaled data that the degree 25 polynomial with the QR method obtains the best results. The least squares error for the deg. 25 QR method is 0.0774 - lower than any other polynomial.

Problem 2:

a. I emulated the provided Gatlin.m code to create the rank r = {2, 4, 8, 16, 32, 64, 128} approximations for the mandrill and durer images. It was quite interesting to see how the image changes with the rank and also to see how low the rank can get while still creating a reasonable-quality image. See Figures 5 and 6 below for these results:

Figure 5: Mandrill SVD Approximation

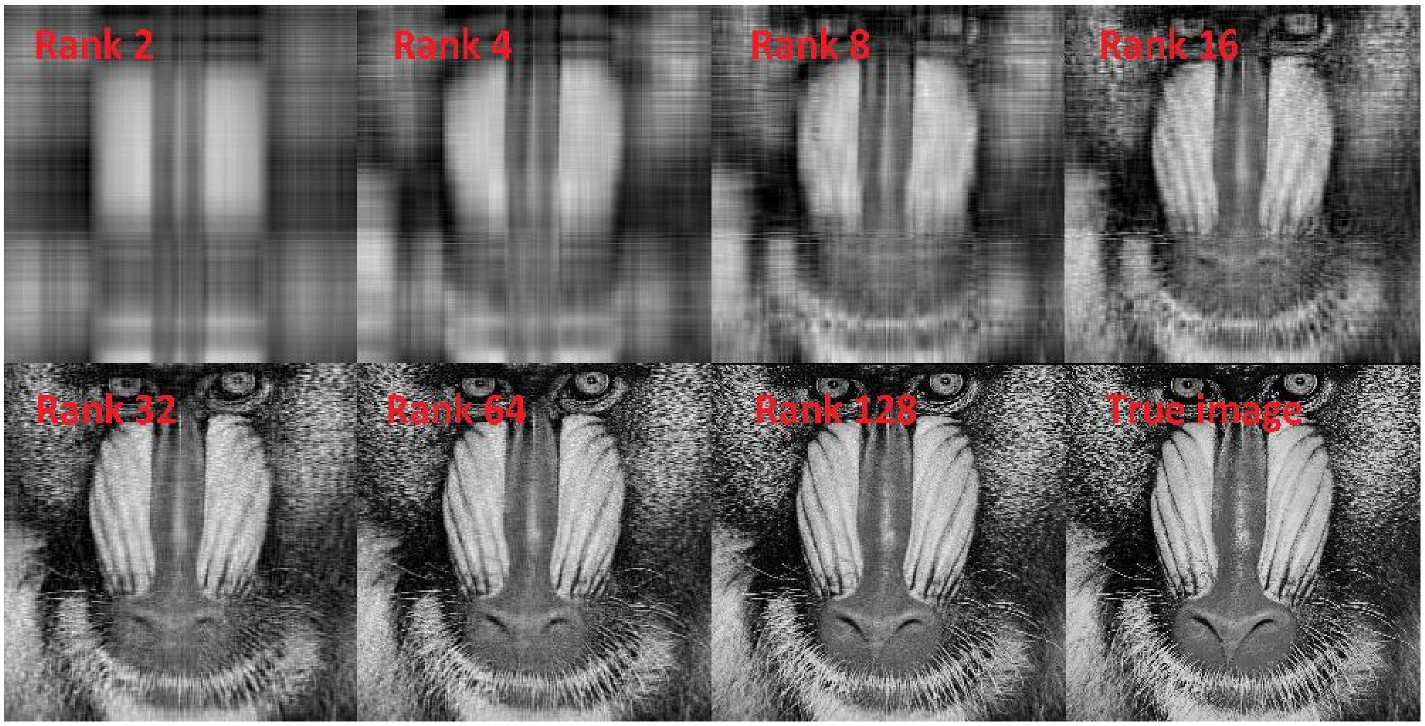
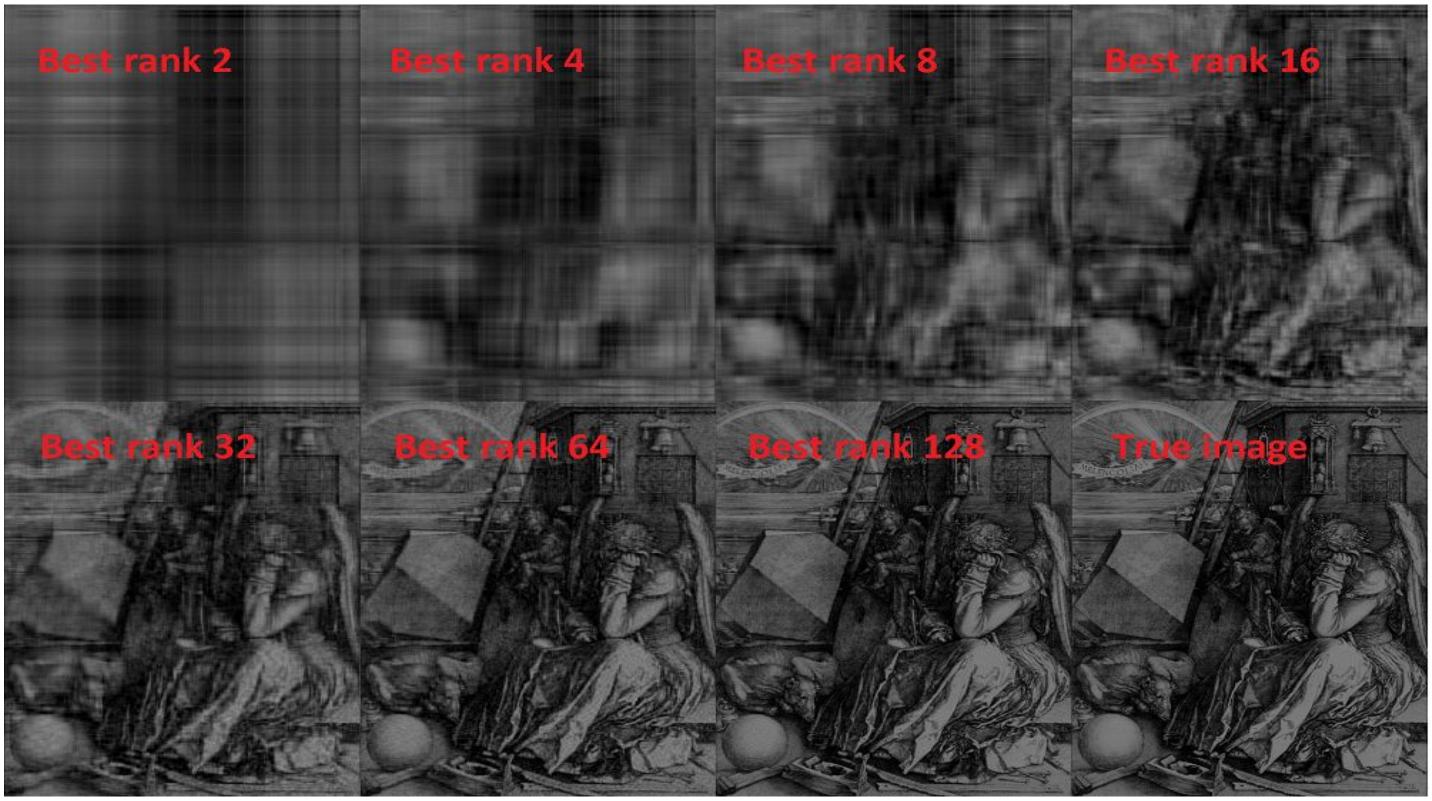


Figure 6: Durer SVD Approximation



b. The SVD truncation definitely reduces the file size, although not quite as much as I expected. I imagine there is some overhead in the image format that takes up a certain amount of size regardless. Nevertheless, one can see how SVD provides an effective compression algorithm for images. It depends on the context of how the image is used, but for certain cases I can see how you would be satisfied with a rank-n SVD approximation where n is much lower than the original value. In our example, we can see that the rank 32 result, while being over an order of magnitude less than the original rank 480 image, still results in decent image quality while saving significantly on file space. See table 2 for the file size results.

Table 2: File Sizes of SVD Rank Approximations

Rank	Durer jpg size	Mandrill jpg size
2	23 KB	24 KB
4	27 KB	28 KB
8	32 KB	35 KB
16	37 KB	44 KB
32	44 KB	55 KB
64	52 KB	65 KB
128	59 KB	75 KB
480 (original)	65 KB	84 KB