Brandon Walters

CS 3200

Prof. Berzins

Homework 5 Report

1. For the given population dataset given from the years 1900 to 2000,
    a. We can see from the data printed by the method that our gradient descent is not working on this particular set of data and the matrices constructed by it. An easy failure indicator is the iteration number being 1 for each successive Vandermonde matrix.
    b. If we look at the condition numbers of the Vandermonde matrices that we generate, we can immediately see that they are extremely high, starting at 2.32e+15 and climbing all the way up to 3.04e+44 on the full matrix. Obviously this is an extremely high condition number, meaning that the matrix is ill-conditioned, and therefore our gradient descent method cannot converge. The reason for this can be found by looking at our years() data: it is used to create the Vandermonde matrix, and we can see that it starts up at a whopping 1900 for the beginning index and goes all the way up to 2000. Using this kind of dataset to create our Vandermonde matrix leads to a very quick explosion of values as the powers are applied for successive iterations, meaning that the matrix quickly becomes ill-conditioned and preventing the Gradient Descent method from converging on this particular Vandermonde matrix.

c. With the newly scaled year set creating our Vandermonde matrix, we can see that our condition numbers for the different Vandermonde matrices have fallen by orders of magnitude, now only going from 7.00e+0 to 1.40e+4, which is a drastic improvement. We can also see that our iterations are now going up properly as they should, from 279 all the way to 17,326,835. Overall, we can see that now our Gradient Descent method is converging properly, with both the value and residual norms holding steady around the e-5 range of values, which is our selected tolerance.
d. When varying between polynomials of degree 2 to 5 with a tolerance of 1e-5, we find that our least square residuals start at around 100 for the polynomial of degree 2, and begin to go down, with the lowest being generated by the polynomial of degree 5 (of the form ax^5 + bx^4 + cx^3 + dx^2 + ex + f, meaning that we generate 6 coefficients), which is 58.4114. Each successive growth of the polynomial gives a reduction of the least squares residual. As we can see, the least square residuals change slightly between the different sizes, with larger polynomials being smaller.
e. Of the polynomials, I found that the best fit to the actual populations in 2010 and 2019 was the polynomial of degree 4, which were calculated to be 302.2344 and 317.2513 respectively. Although both of these numbers are slightly under the actual data of 308.745 and 328.239, they were the closest generated to the actual numbers. Coupled with the LSR data from Part D, where we saw that the higher polynomials had lower LSRs, with the degree 4 polynomial having a LSR of 81.6216, which is a good

improvement over the second-best guess generated by the degree 2 polynomial (whose LSR is 102.8159), my conclusion is that the polynomial of best fit is the polynomial of degree 4.

f. When tweaking the alpha value in our Gradient Descent function, I found that a multiplication factor of 0.85 gave the best results, lowering the iterations needed for the degree 4 polynomial from 1339 to 132 iterations, and lowing the residual norm from 6.47e-05 to 2.44e-05. This alpha adjustment also works really well with the rest of the polynomials, with iterations brought down from the thousands to just a few hundred per polynomial.

2. For the PowerMethod function work on the given matrices,
    a. We can find that the largest eigenvalue of our given matrix A is 16.1168 when used with a starting vector guess of all 1s, which is equivalent to the value generated by the eig() method given to us by Matlab. We also find that our largest eigenvector is [0.2833, 0.6417, 1], also consistent with the results generated by Matlab's eig() function.
    b. As we can see when the method is run with the new B matrix and initial vector, our Power Method fails to converge, leading to a NaN result. Obviously this is a failure, which in this particular case is caused by a problem with our eigenvalues, something we can see if we examine the eigenvalues generated by the eig() method. The eigenvalues for this particular matrix are 3, 1, and -3. The Power Method can only converge if there is a dominant eigenvalue; that is, an eigenvalue with a higher

absolute value than any other. However, as we can see, there are two eigenvalues with an absolute value of 3. This means that there is no dominant eigenvalue, and as such the Power Method cannot converge.

c. With the new starting vector, we can see that the Power Method now converges in only one iteration, but converges to the wrong eigenvalue. This is because of the fact that our starting vector is actually equal to one of the matrix's eigenvectors, something we can see from the PowerMethod printout of the generated x from the method, it's the exact same as our starting vector! As such, our Power Method ends up immediately converging onto that particular eigenvector and associated eigenvalue, which is why this particular version gives 1 instead of NaN, even though 1 is not the actual correct answer of 3.
d. Going back to our A matrix from Part A, we can make a single character change in our Power Method, changing the multiplication of A and x_prev into a \ to get our smallest eigenvalue of A, which is 0. This is supported by the eig() function's output, where the smallest value is -9.7592e-16, a value very close to 0 (remember, the size of an eigenvalue is measured in absolute value, so 0 is the smallest, not -1.1168!).
e. For the biology example with the given birth and death rates,
    i. We can find that the largest eigenvalue of the resulting matrix as constructed in the slides is 0.9106, which can be confirmed using the eig() function.

ii. Since the largest eigenvalue of the matrix is less than 1, we should expect that the population of the species will dwindle and die out over time.

iii. With the given starting populations, at year 1000 we can find that all of our population numbers are in the e-81 range, which means they have all diminished to 0, a conclusion which supports our initial eigenvalue analysis from Part II.

iv. With the new death rate for the P4 population, we can find our new largest eigenvalue is 1.0797, which is over 1, thereby meaning that our population should grow forever. At year 1000, we can see that all of the population segments are around the e+48 mark, which shows that our population has grown unbounded over the 1000 years, a conclusion which supports our initial eigenvalue observation. We can conclude that changing the d4 rate from 0.9 to 0.01 has changed our population dynamic from one of decline and extinction into one of unbounded growth.