Dan Ruley
CS 3200
Assignment #3
March 19, 2020

1.)

Given $B = \begin{bmatrix} 4 & 1 & -2 \\ 4 & 4 & -3 \\ 8 & 4 & 2 \end{bmatrix}$ and $C = \begin{bmatrix} 2 & 1 & -2 \\ 4 & 4 & -3 \\ 8 & 4 & 4 \end{bmatrix}$

We can see the two given matrices look similar; they only differ by a small amount.

Here are the LU factorizations of B and C:

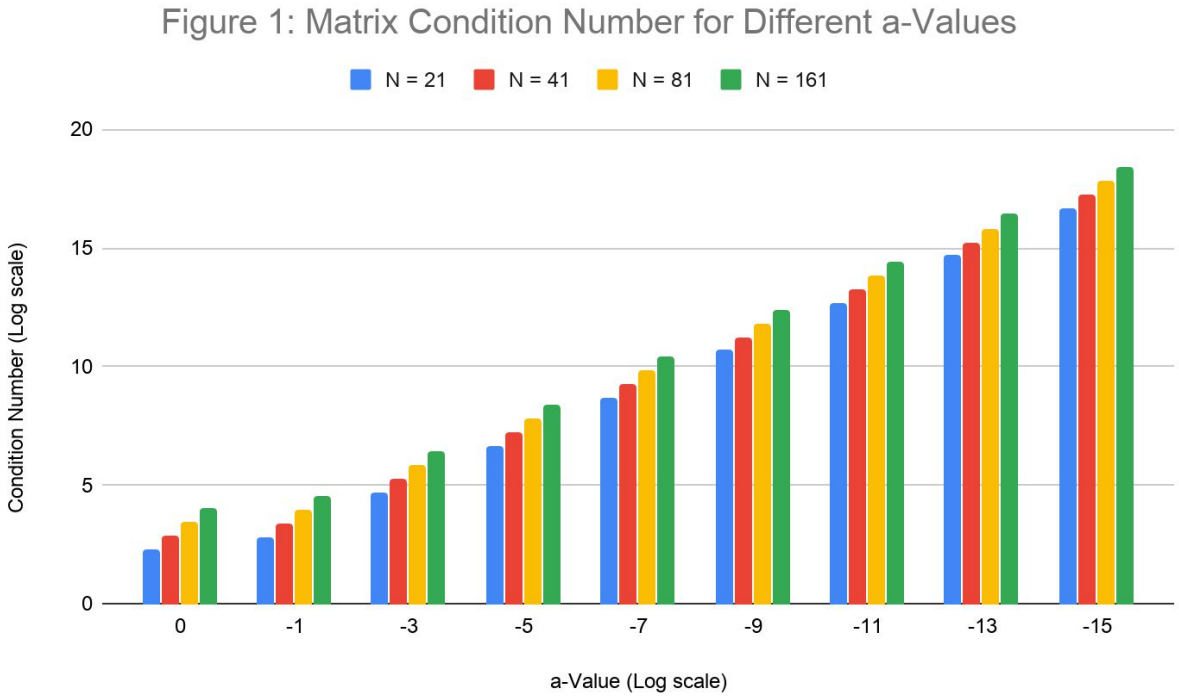
$$BL = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & -0.5 & 1 \end{bmatrix} \quad BU = \begin{bmatrix} 8 & 4 & 2 \\ 0 & 2 & -4 \\ 0 & 0 & -5 \end{bmatrix} \quad CL = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.25 & 0 & 1 \end{bmatrix} \quad CU = \begin{bmatrix} 8 & 4 & 4 \\ 0 & 2 & -5 \\ 0 & 0 & -3 \end{bmatrix}$$

These also seem superficially similar, the numbers in the L factors differ by 0.5 at the most, while the numbers in the U factors differ by 2 at the most.

Looking at the condition numbers of the LU matrices for B and C, we can see that they are close in terms of order of magnitude; however, the CU condition is about twice that of the BU, while the BL condition is twice the CL. In this sense they do not seem quite as similar because they are likely to perform quite differently when used to solve actual systems of equations.

BL-inf cond.	BU-inf cond.	CL-inf cond.	CU-inf cond.
4.5	12.6	2.25	21.33333333

2.)
I built up the matrix as described (using the assumption that the … on the upper half of the matrix should be (1, -2, 1) and the … on the bottom half should be (a, -2a, a)). Then I calculated the condition numbers for each value of N and a. See Figure 1 on the next page for a graph of these results.



It seems that the condition number increases proportionally to the decrease in the a-values. Essentially, each time a decreases by two orders of magnitude the condition number increases by two orders of magnitude. Note that both the a-values and the condition numbers are plotted using a log scale so this effect can be illustrated better. Additionally, as N doubles we can see that the condition number also jumps by an order of magnitude. So larger matrices of this form are more poorly conditioned than smaller matrices.

I solved the system of equations using $H_l = 8$, $H_r = 4$, and N = 161 for the a-values 1, 1e-5, and 1e-15. I found that even though the matrices are ill conditioned, the solutions are accurate in each case. I confirmed this by multiplying the matrix with the solution vector x* and noticing that the result is nearly identical to the B vector of $(-H_l,..,-aH_r)$. I used iterative refinement to observe its effect on the solution but observed that this effect is minimal. This is because the first residuals calculated are already at the level of roundoff error, i.e. ~1.0e-16. After one round of iterative refinement, the residuals do decrease slightly but since they are right around machine roundoff error, the effect is minimal.

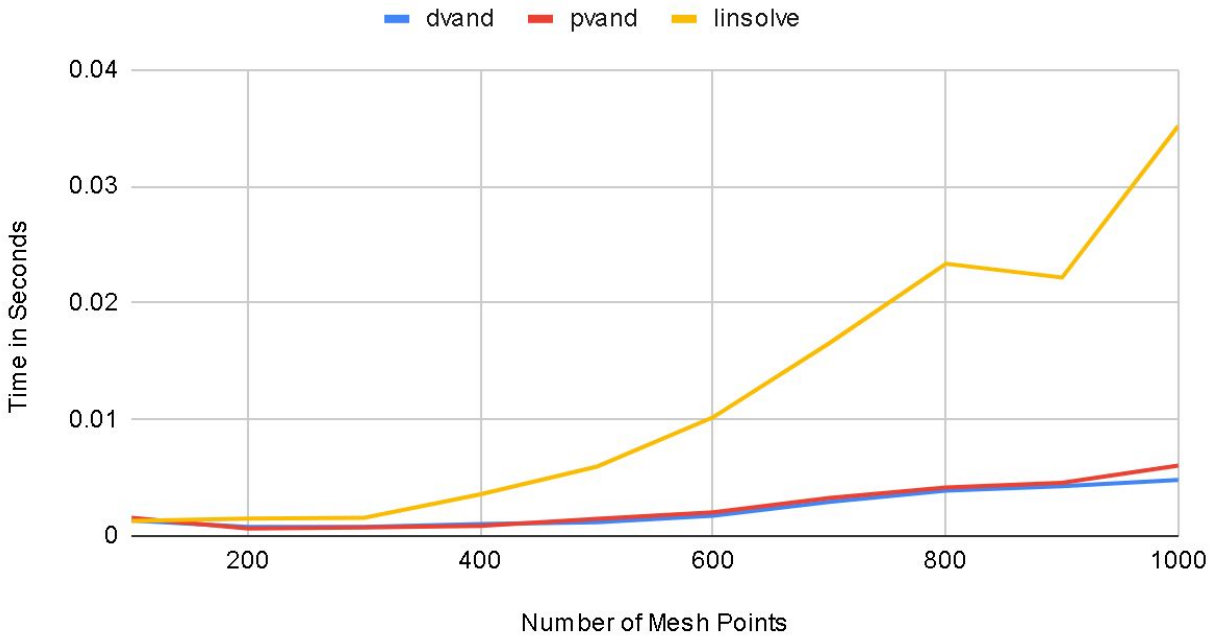
3.)
I set up vectors of mesh points in the interval of [0,1] using spacings of 100 to 1000, increasing by increments of 100. I then evaluated the mesh points using the exponential function to find a b vector. Also, I built the Vandermonde matrix using Matlab's vander(x) function and recorded condition numbers, which were massive for every N value. I then computed the coefficient vector of the polynomial approximation by using the linsolve function on the Vandermonde matrix and the b vector and timed this process. See Table 2 for these results:

Table 2: Timing and Condition Numbers for Matlab Vandermonde Solve

Mesh Points in [0,1]	Matrix Cond. #	Linsolve Time
100	2.15E+20	0.0012969
200	3.60E+20	0.0015019
300	4.22E+20	0.0015498
400	1.74E+22	0.0035731
500	7.33E+20	0.0059533
600	4.23E+21	0.0101843
700	2.76E+21	0.0165479
800	5.62E+21	0.0233606
900	1.80E+22	0.0221739
1000	1.56E+22	0.0351917

Next, I retrieved the dvand.m and pvand.m codes from the provided website. I solved the system using the same alpha vector of mesh points and b vector. I recorded the times for these solves. See Figure 2 on the next page for a graph comparing the solve times for dvand and pvand to Matlab linsolve.

Figure 2: Comparing Solve Times for dvand, pvand, linsolve



Clearly, both dvand and pvand significantly outperform linsolve. While they are close for lower mesh spacing values, by the time the mesh gets larger the linsolve times are about an order of magnitude higher than those of the pvand and dvand times. One would expect this trend to continue as linsolve using the matlab Vandermonde matrix grows in $O(N^3)$ time.

In all three cases, however, the solution was very inaccurate. This is likely due to the extremely high condition numbers of the Vandermonde matrices. For mesh spacing of only 100, the 100x100 Vandermonde matrix already has a condition number over 10^{20}. Therefore it does not produce a reasonable approximation for the exponential function.

I ran a smaller sample with n=10 and found that the solution is very accurate for dvand and linsolve. Surprisingly, it was inaccurate for pvand, for reasons that are unclear to me. See figure 3 on the next page for these results.

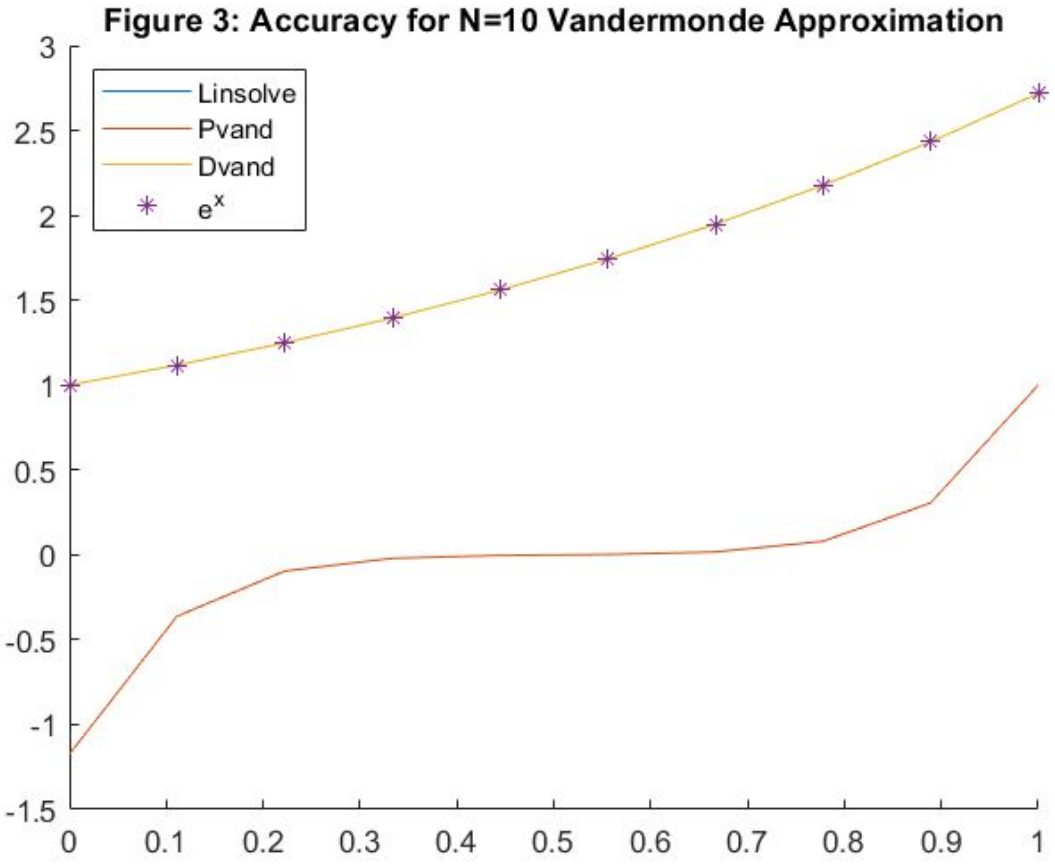


Figure 3: Accuracy for N=10 Vandermonde Approximation
Linsolve
Pvand
Dvand
e^x
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1