Brandon Walters

CS 3200

Prof. Berzins

Homework 3 Report

1. We know that the given matrices B and C are very superficially similar, with the only difference being that the values in the top-left and bottom-right spots in the matrix are swapped. We can then compute their LU decompositions by hand, both with and without pivoting, to see if the decomposition matrices are also superficially similar. As we can see from the final forms in the picture below, without pivoting, our lower and upper matrices for each matrix are significantly more different: multiple positions are completely different between each matrix, without any real correlation to their changes between the different decompositions. As we can see, the two similar matrices B and C become much more different when using LU decomposition without pivoting. With pivoting, however, the permutation, lower, and upper matrices are more similar, with the permutation matrices in particular being the exact same between the two. The upper and lower matrices with pivoting also show a much closer similarity, with entries either the same or only different by about 1. As we can see, adding pivoting into our LU decomposition makes our resulting matrices much more

similar to the originals than without pivoting.

Without pivoting:

Matrix B: $L: \begin{Bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 2/3 & 1 \end{Bmatrix}$ $U: \begin{Bmatrix} 4 & 1 & -2 \\ 0 & 3 & -1 \\ 0 & 0 & 20/3 \end{Bmatrix}$

Matrix C: $L: \begin{Bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 4 & 0 & 1 \end{Bmatrix}$ $U: \begin{Bmatrix} 2 & 1 & -2 \\ 0 & 2 & 1 \\ 0 & 0 & 12 \end{Bmatrix}$

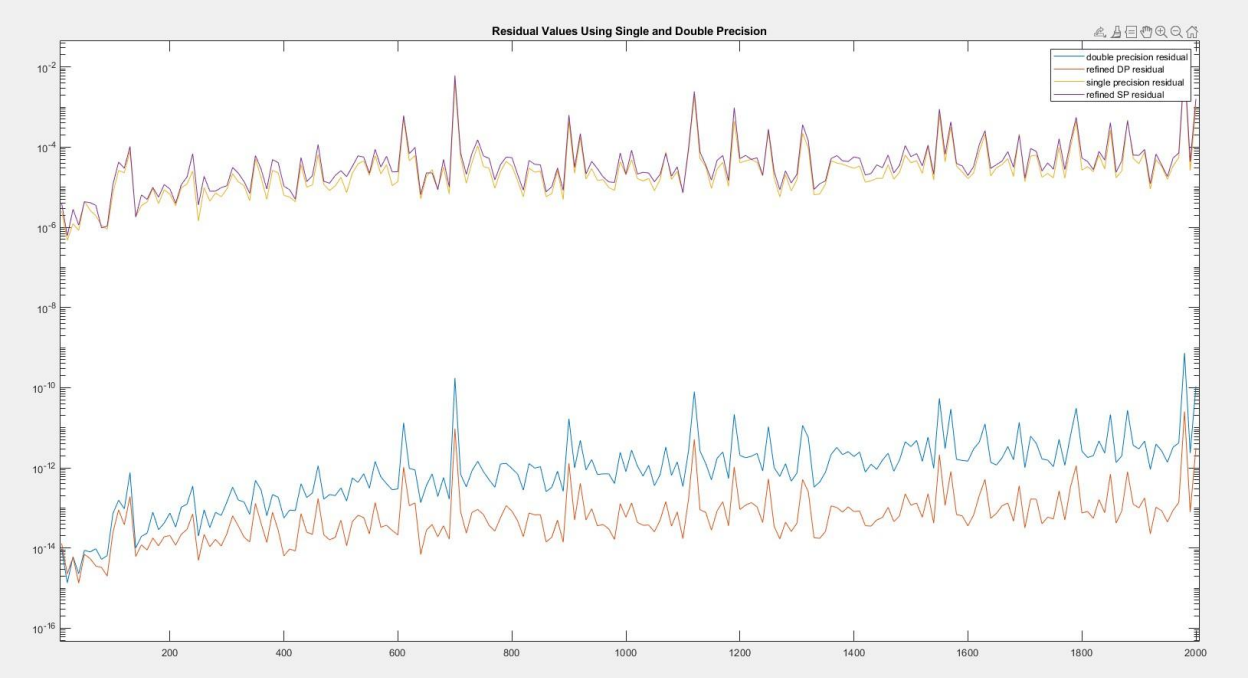
With pivoting:

Matrix B: $P: \begin{Bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{Bmatrix}$ $L: \begin{Bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/2 & -1/2 & 1 \end{Bmatrix}$ $U: \begin{Bmatrix} 8 & 4 & 2 \\ 0 & 2 & -4 \\ 0 & 0 & -5 \end{Bmatrix}$

Matrix C: $P: \begin{Bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{Bmatrix}$ $L: \begin{Bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/4 & 0 & 1 \end{Bmatrix}$ $U: \begin{Bmatrix} 8 & 4 & 4 \\ 0 & 2 & -5 \\ 0 & 0 & -3 \end{Bmatrix}$

2. As we can see from the graph of our infinity-norm residual values provided below, the difference between double-precision and single-precision is quite significant, with residual values dropping from around 10^-5 for single-precision to around 10^-15 for double-precision values. Clearly, double-precision's added accuracy leads to much more accurate calculations, and therefore much lower residuals. We can also see the effects of two steps of iterative refinement on our residual values. We can notice that for double-precision values, our iteratively refined values are consistently lower than our non-refined values by about a power or two, which shows that our iterative refinements are having a positive impact on the accuracy of our calculations. However, our single-precision calculations do not seem to be affected in any way by our iterative refinement methods, with the refined values actually returning residuals that are just slightly higher than our non-refined single-precision residuals. This shows that the loss of accuracy from our single-precision values in our matrices and arrays leads to a corresponding loss in effectiveness on our iterative refinement strategy, at least for the two iterations that we perform for our particular implementation of this

iterative refinement strategy.



3. Our table, provided below, gives all of the data needed to analyze our results for the calculations of our ill-conditioned matrices. As we can see from the table, our condition numbers start relatively high and immediately climb much higher as our alpha values get smaller and the size of the matrix gets larger, which is exactly what we expect for an ill-conditioned matrix, as the higher condition numbers reflect that the matrix is much more sensitive to changes in the input data and roundoff issues, which is a problem as our alpha value becomes smaller and more precise. Correspondingly, we can see that our constant number that we multiply with our alpha values to find each condition number slowly increases as the alpha values decrease. These constant values do decrease when viewed against the trend of matrix size, ranging from between 0.02 to 0.0003 as the size of the matrix increases. This trend in our constant values further proves the correlation between our condition numbers and our alpha values: as our

condition numbers increase across our samples, then to compensate our constant must decrease such that our equation 1/(const * a) yields a larger number. Finally, we can look at our residual values to see how our iterative refinement strategy affects our numbers. As we can see in our Original Residual column, we get a constant residual value of 8 for all iterations, which means that our matrix is close to singular, which is consistent with the fact that our matrices are ill-conditioned, meaning that they are very close to or are singular and therefore have very low residuals. However, we can notice that our first iteration for our iterative refinement strategy produces a universal residual value of 0 for all of our different matrices, which means that our matrices are now singular. Since we know that a lower residual value is better and means that our estimation is closer to the actual values of our operation, we can tell that this first iteration step has succeeded in reducing the norm of the residual. However, we can also see in the final column that our second iteration also produces a residual value of 0, which means that our second iteration step does not provide any improvement in our residual calculation compared to the residuals created from our first iterative refinement. Therefore, we can conclude that our residuals are improved by one iteration of our iterative refinement strategy, but our second

refinement does not create any improvements.

	Condition Number	Constant for Cond Calc	Original Residual	First Iteration for Refinement	Second Iteration for Refinement
Matrix Size = 21, a = 1.0e-1	617.7476	0.0162	8	0	0
Matrix Size = 21, a = 1.0e-3	4.8554e+04	0.0206	8	0	0
Matrix Size = 21, a = 1.0e-5	4.8460e+06	0.0206	8	0	0
Matrix Size = 21, a = 1.0e-7	4.8459e+08	0.0206	8	0	0
Matrix Size = 21, a = 1.0e-9	4.8459e+10	0.0206	8	0	0
Matrix Size = 21, a = 1.0e-11	4.8459e+12	0.0206	8	0	0
Matrix Size = 21, a = 1.0e-13	4.8459e+14	0.0206	8	0	0
Matrix Size = 21, a = 1.0e-15	4.8459e+16	0.0206	8	0	0
Matrix Size = 41, a = 1.0e-1	2.2765e+03	0.0044	8	0	0
Matrix Size = 41, a = 1.0e-3	1.7847e+05	0.0056	8	0	0
Matrix Size = 41, a = 1.0e-5	1.7811e+07	0.0056	8	0	0
Matrix Size = 41, a = 1.0e-7	1.7811e+09	0.0056	8	0	0
Matrix Size = 41, a = 1.0e-9	1.7811e+11	0.0056	8	0	0
Matrix Size = 41, a = 1.0e-11	1.7811e+13	0.0056	8	0	0
Matrix Size = 41, a = 1.0e-13	1.7811e+15	0.0056	8	0	0
Matrix Size = 41, a = 1.0e-15	1.7811e+17	0.0056	8	0	0
Matrix Size = 81, a = 1.0e-1	8.7061e+03	0.0011	8	0	0
Matrix Size = 81, a = 1.0e-3	6.8201e+05	0.0015	8	0	0
Matrix Size = 81, a = 1.0e-5	6.8065e+07	0.0015	8	0	0
Matrix Size = 81, a = 1.0e-7	6.8064e+09	0.0015	8	0	0
Matrix Size = 81, a = 1.0e-9	6.8064e+11	0.0015	8	0	0
Matrix Size = 81, a = 1.0e-11	6.8064e+13	0.0015	8	0	0
Matrix Size = 81, a = 1.0e-13	6.8064e+15	0.0015	8	0	0
Matrix Size = 81, a = 1.0e-15	6.8064e+17	0.0015	8	0	0
Matrix Size = 161, a = 1.0e-1	3.4011e+04	2.9402e-04	8	0	0
Matrix Size = 161, a = 1.0e-3	2.6638e+06	3.7541e-04	8	0	0
Matrix Size = 161, a = 1.0e-5	2.6585e+08	3.7616e-04	8	0	0
Matrix Size = 161, a = 1.0e-7	2.6584e+10	3.7616e-04	8	0	0
Matrix Size = 161, a = 1.0e-9	2.6584e+12	3.7616e-04	8	0	0
Matrix Size = 161, a = 1.0e-11	2.6584e+14	3.7616e-04	8	0	0
Matrix Size = 161, a = 1.0e-13	2.6584e+16	3.7616e-04	8	0	0
Matrix Size = 161, a = 1.0e-15	2.6584e+18	3.7616e-04	8	0	0