Problem Set 3
Math 146 Spring 2023

1. (30 points)

   (a) Write a Matlab function for solve $A\mathbf{x} = \mathbf{b}$ by using Gaussian Elimination without pivoting, and backward/forward substitution. Your function should take as inputs: square matrix A and vector $\mathbf{b}$. Its output should be the solution $\mathbf{x}$. Hint: Within this function, you can call other functions saved in your folder. Use the functions you made for Problem Seet 2, problems 5-7.

   See the function directlinsolve_nopivoting.m.

   (b) Validate your code by forming a 100×100 matrix for A made up of random numbers between 0 and 50 and a similarly formed vector for b. Solve for $\mathbf{x}$ using both your function and the Matlab backslash and find the relative difference between the solutions, relative to the max norm. Comment on your results, using the comments from Problem Set 2, Number 5 to help guide your analysis.

   See the file PS3N1.m. An example maximum relative difference between the solutions for a randomly generated A and $\mathbf{b}$ is $4.3(10)^{-13}$. First of all, this is small enough to validate my code, but we should investigate further to see if it makes sense. For this same matrix, the condition number with respect to the max norm is $3.6(10)^3$. This tells me that my relative error should be about on the order of 10^{-13} or 10^{-12}. (In other words, I would expect to lose about 3 to 4 digits of accuracy). We do not know what the relative error for this problem is, since we do not have an exact solution to which to compare. However, the Matlab solution is also expected to have a relative error of about the same magnitude (since a condition number characterizes the problem, not the algorithm used to solve it). Since we are using two different algorithms (theirs uses pivoting; ours does not), our error and the Matlab error are not likely to be the same. Therefore, the relative difference between our solution and the Matlab solution is about the same size as the true relative error (which, again, is unknown). Therefore, this relative difference is exactly what we expect to see. This was all discussed a bit more abstractly in PS 2, Number 5 solutions. So here we can see that in fact matches what would be expected. (In PS 2, Number 5, on the other hand, the algorithms were the same, so their errors were about the same, so we did not witness this same phenomenon.)

2.

3. (a) (40 points) Write a Matlab function to use Gaussian Elimination with partial pivoting to perform LU decomposition for a matrix A. Your function should take as input A, an $n \times n$ matrix. Its output should be the lower triangular and upper triangular matrices, L and U, and the permutation matrix, P. Validate your code. Explain how you validated it and give any relevant output.

See the function myLUdecomp_partialpivoting.m for the function and PS3N3.m for the script. I validated my code by forming a random 100×100 matrix of numbers from 0 to 20. Then, I calculated P, L, and U with my function, and then found $PA - LU$ to confirm these matrices. I then found the component of the difference matrix of maximum absolute, which I found to be $3.9(10)^{-14}$, confirming my code.

(b) (25 points) Write a Matlab function to solve $A\mathbf{x} = \mathbf{b}$ by using Gaussian Elimination with partial pivoting, and backward/forward substitution. Your function should take as inputs: square matrix A and vector $\mathbf{b}$. Its output should be the solution $\mathbf{x}$. Validate your code. Explain how you validated it and give any relevant output.

See the function directlinsolve_partialpivoting.m for the function and PS3N3.m for the script. I validated my code by forming a random 100×100 matrix of numbers from 0 to 20, as well as a similarly formed vector, $\mathbf{b}$. Then, I calculated my solution and the solution from Matlab's backslash and found the relative difference, with respect to the max norm. For one trial, I got a relative difference of $2.7(10)^{-13}$, validating my code. Additionally, the condition number of this matrix, with respect to the max norm, is about 10^5, indicating that the true relative error should be about 10^{-11}. Since the Matlab algorithm and my algorithm are likely similar at this point, it is not surprising that the relative difference is a bit smaller than this.

(c) (30 points) Use your code from part (b) to solve the linear system $A\mathbf{x} = \mathbf{b}$ for `A= [20:2:36; 100:2:116; -20*ones(1,9); linspace(0.001, 0.005,9);21:2:37; 23:2:39; 24:2:40 ; linspace(-0.6, 0,9); linspace(-17,20,9) ]` and `b=[1380; 4980; -900; 0.165; 1425; 1515; 1560; -9; 345]`. The exact solution to this is $x_{exact} = [1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7 \;\; 8 \;\; 9]^\intercal$. All with respect to the 2-norm, find the relative error for your solution, the relative error for the Matlab solution found by using backslash, and the relative difference between your solution and the Matlab solution. Comment on your results.

See the script PS3N3.m. My relative error with respect to the 2-norm was 188.26. Matlab's relative error was 44.36, and our relative difference was 217.26 (which we can again see is of about the same magnitude as the true relative errors). These relative errors look absolutely terrible, but by checking the condition number of A, with respect to the 2-norm, we see that this is actually expected. The condition number is about 10^{21}. This tells us that we expect to see a relative error on the order of $10^{-16}(10^{21}) = 10^5$, which

is even worse than the relative error that we do see. (Remember that the condition number tells you the "worst-case scenario", but the true relative error still depends on $\mathbf{b}$). Therefore, it is completely reasonable that we see these bad errors. This means we cannot trust any digits of our solutions. (Look at your solution versus the real solution, and you will see they are completely unrelated.) This shows you that even though A was invertible, because of floating point arithmetic and the finite number of bits we have to store our numbers, this matrix cannot be accurately numerically inverted. In fact, Matlab prints out an error warning you of this that says "Warning: Matrix is close to singular or badly scaled. Results may be inaccurate." Try doing `inv(A)*A`, and again, you will see you do not get the identity. Therefore, this demonstrates that the round-off errors of floating point arithmetic can sometimes play a major role in the accuracy of your solution.

(d) (40 points) Perform a computational experiment in which in a single trial, you generate a random square matrix of a given size. Generate a random vector $\mathbf{z}$ and compute $\mathbf{b} = A\mathbf{z}$. Then use your code in part (b) to solve the linear system $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$. Compute the relative error, $||\mathbf{x}-\mathbf{z}||/||\mathbf{z}||$, with respect to the 2 norm. Also compute the condition number of the matrix. Do this for matrices of various sizes from $n = 5$ to 500. Make a scatter plot of the relative error vs. the condition number on a log-log plot. (Matlab command `loglog`). Look at the scaling between the condition number and the relative error. Does the scaling depend on the matrix size? Do the results match your expectation? Discuss.

See the script PS3N3D.m. See the below log-log scatter plot of the relative error and relative condition numbers. Firstly, we see that the relative error increases as the condition number of the matrix increases. This of course makes sense, as we have learned that a larger condition number means a matrix has worse conditioning, which according to our rule of thumb, that the relative error is $\mathcal{O}(\kappa\epsilon_{mach})$, means that the relative error would increase in size (i.e. our solution is less accurate).

Next, we notice that the relationship of the logs of these quantities is about linear. Again looking at the rule of thumb we had from class, we could say say that, in the limit that $n \longrightarrow \infty$,

rel error $\approx C_1\kappa\epsilon_{mach}$ for some constant C_1

$\implies \log_{10}(\text{ rel error }) \approx C_2 - 16 + \log_{10}(\kappa)$,

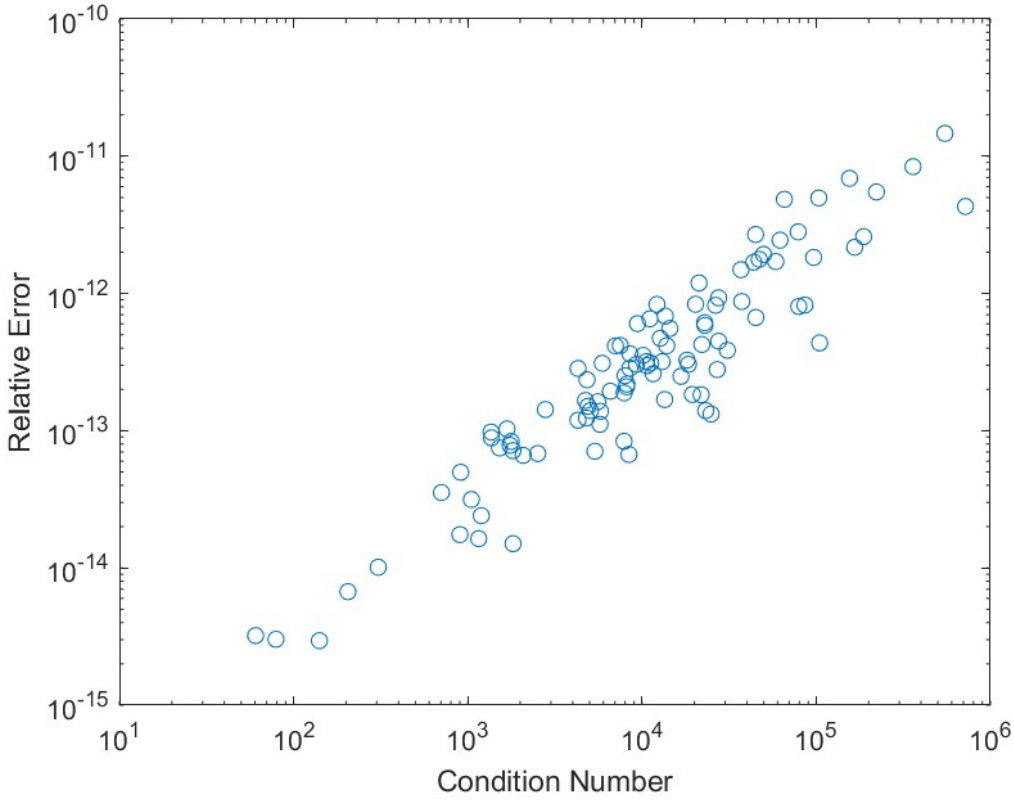
where $C_2 = \log_{10} C_1$

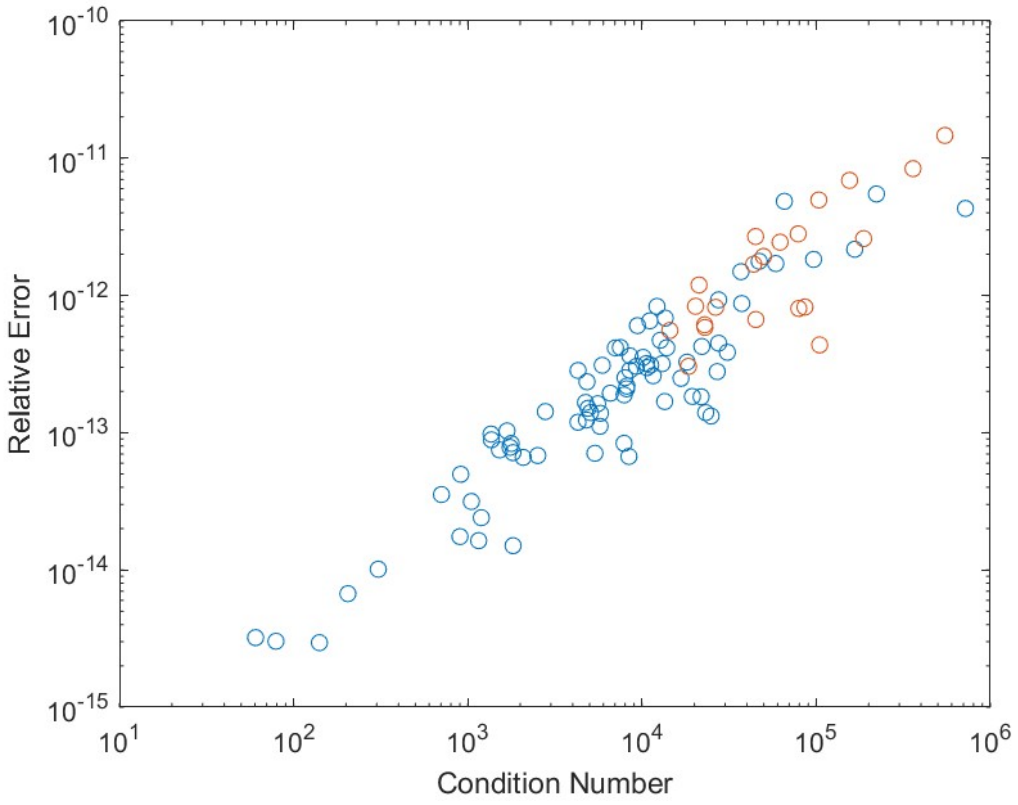
Therefore, the slope of the line we see in the scatter plot would be about 1.

To see this better, simply let $x = \log_{10}(\kappa)$ and $y = \log_{10}(\text{ rel error })$. Then the above formula is the equation for a line with slope 1. (Keep in mind that if we used natural log, as we do in the plot, we would get the same formula, just with a different constant term.) We can see by eyeballing it that the slope of this line is in fact near to 1 just as we expect. Therefore, the scaling between the condition number and relative error is that if we increase the condition number by a factor of 10 (or, say, any positive constant

α), we increase our relative error by a factor of 10 (or α). Again, this all matches the rule of thumb mentioned above.

Lastly, using the rule of thumb and the analysis above, since there is no n dependence, we do not expect the scaling to depend on matrix size. To explore things a little further, I also plotted the same plot but with the largest 20 matrices plotted in red. We see that the larger randomly generated matrices tend to have larger condition numbers. But since the slope of this line does not change as the condition number grows, the scaling between relative error and condition number does in fact not depend on the matrix size.




Relative Error
Condition Number
10^{-10}
10^{-11}
10^{-12}
10^{-13}
10^{-14}
10^{-15}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}