Dan Ruley
CS 3200
Assignment #4
March 31, 2020

Problem 1:

I wrote a function to build the specified matrix for any input size N. Using N = 8 and tol = 1e-5, I built the matrix and then used the Jacobi and GaussSiedel iterative solvers to find a solution to the equation Ax = b, where b is column vector containing the values 1 / N^4 for each element. Here is the output of my program:

Jacobi results for N = 8 and tol = 1e-5:
Error: NaN, Convergence value: NaN, Iterations: 1897

GaussSiedel results for N = 8 and tol = 1e-5:
Error: 0.043845, Convergence value: 0.973489, Iterations: 101

We can see clearly that the Jacobi method fails completely, the x solution vector does not converge and the numbers blow up to Inf or NaN. Accordingly, the calculated c indicates a lack of convergence. Even worse, it takes 1897 iterations to deliver these lackluster results.

On the other hand, GaussSiedel does much better. The relative error indicates that the solution vector is accurate to about 2 decimal places, and the c value indicates that there is convergence. It only takes 101 iterations which is much less than the Jacobi and it actually delivers reasonable results.

Problem 2:

I extended my program to do more experiments with the GaussSiedel method, testing GaussSiedel with the given matrix for values of N = 16, 32, 64 and tol = 1e-6, 1e-8, and 1e-10. See below for tables containing the results from these experiments.

Table 1.1: GaussSiedel results for tol=1e-6

tol = 1e-6	Iterations	Error	Convergence
N=16	1922	0.04960049718	0.9985544559
N=32	4909	0.5574679782	0.9999171764
N=64	1	1.036602063	Inf

Table 1.2: GaussSiedel results for tol=1e-8

tol=1e-8	Iterations	Error	Convergence
N=16	5105	0.00049633964	0.998554456
N=32	60485	0.00557458629	0.999917142
N=64	526082	0.06295359132	0.9999950514

Table 1.3: GaussSiedel results for tol=1e-10

tol=1e-10	Iterations	Error	Convergence
N=16	8289	4.96E-06	9.99E-01
N=32	116062	5.57E-05	1.00E+00
N=64	1456797	0.00062953775	0.9999950459

Clearly, the tolerance is a large factor in determining the behavior of the GaussSiedel method. As Professor Berzins mentioned in the assignment prompt, if tol is too low, then the function may incorrectly converge for larger values of N. Specifically, when N = 64 and tol = 1e-6, GaussSiedel "converges" after 1 iteration; however, the result is very inaccurate and the calculated c-value does not indicate convergence.

Lower tolerances make GuassSiedel more accurate, especially for smaller N sizes. The smallest error values were from N=16 and N=32 with tol=1e-10. One drawback of GaussSiedel seems to be the number of iterations, which increase dramatically as the size of the input matrix increases. For lower tolerances and N=64, we can see it takes a very large amount of iterations to converge below the tolerance value. For N=64 and tol=1e-10, it takes almost 1.5 million iterations but results in a higher error.

Problem 3:

I made a third program to run calculations with GaussSiedel and experimented with under-relaxation using different ω-values. I set tol = 1e-10 since this gave the best results for larger N. I also used N=16, 32, and 64 as in the previous problem. See the tables below for these results.

Table 2.1: GaussSiedel results using ω = 0.45, tol = 1e-10.

omega = 0.45	Iterations	Error	Convergence
N=16	17198	1.10E-05	0.9993494999
N=32	236503	0.00012389013	0.999962747
N=64	2878706	0.00139897831	0.9999977523

Table 2.2: GaussSiedel results using ω = 0.5, tol = 1e-10

omega = 0.50	Iterations	Error	Convergence
N=16	15624	9.93E-06	0.9992772404
N=32	215396	0.00011149814	0.999958535
N=64	2633423	0.00125907845	0.9999975688

Table 2.2: GaussSiedel results using ω = 0.55, tol = 1e-10

omega = 0.55	Iterations	Error	Convergence
N=16	14323	9.03E-06	0.9992049324
N=32	197906	0.00010135968	0.999954485
N=64	2429043	0.0011446174	0.999978293

Clearly, the under-relaxation does not seem to reduce the number of iterations for the GaussSiedel solver. It seems to reduce error slightly when N=16, but increases the error for N=32 and N=64. The most accurate results using under-relaxation were for ω = 0.5. In every case, convergence was still achieved in terms of the c-value calculated during the solve.

It is unclear to me why the under-relaxation failed to reduce the iteration count, but I suspect it is because the system already converges. From what I read when doing some googling it seems under-relaxation is typically used for systems that do not converge, while over-relaxation is used for systems that converge but you want to decrease the iteration count (i.e. speed up convergence). Perhaps we would achieve better (or at least faster) results in this case by using over-relaxation instead.