Homework #9
CS 575: Numerical Linear Algebra
Spring 2023

John Tran

April 27, 2023

Important Notes

- `Python` 3.11 was used to run the notebook (i.e., to use the `match` statement)
- the partially completed notebook was used to complete the coding assignment, so many of the things asked (e.g., verification and testing) was done for us and they were modified for the mat-mat portion
- the `README` was done in `Markdown` but the raw text can still be viewed

Problem 1

We are given the SVD of $A \in \mathbb{R}^{m \times n}$ (added constraint where $m > n$ for simplicity, but it can be generalized) as

$$A = U\Sigma V^T$$

and we are asked to find the range and null-space of A^T in terms of the columns of matrices U and V.

The SVD of A^T is merely the tranpose of the above equation with reversed roles of the left and right singular vectors

$$\begin{aligned} A^T &= (U\Sigma V^T)^T \\ &= V\Sigma^T U^T \end{aligned}$$

Since we went over this in the previous homework, a brief summary will be adapted from the previous solution:

For finding the range, we can use the theorem to find r where $\sigma_i > 0$ for $i \leq r$. The range is given as the columns of the ***left*** singular vectors (columns of V for A^T) for the set $\{v_1, v_2, \dots v_r\}$ as the first n columns of V.

Then for finding the null-space of A^T, we would find the rightmost $n - r$ ***right*** singular vectors – from U^T for A^T.

In summary, we have

$$\text{range of } A^T\text{: } \{v_1, v_2, \ldots, v_r\}$$
$$\text{null-space of } A^T\text{: } \{u_{r+1}, u_{r+2}, \ldots, u_n\}$$

Now for a general matrix A (and thus A^T), the null-space of A^T would go up to $u_{\min\{m,n\}}$ instead of u_n

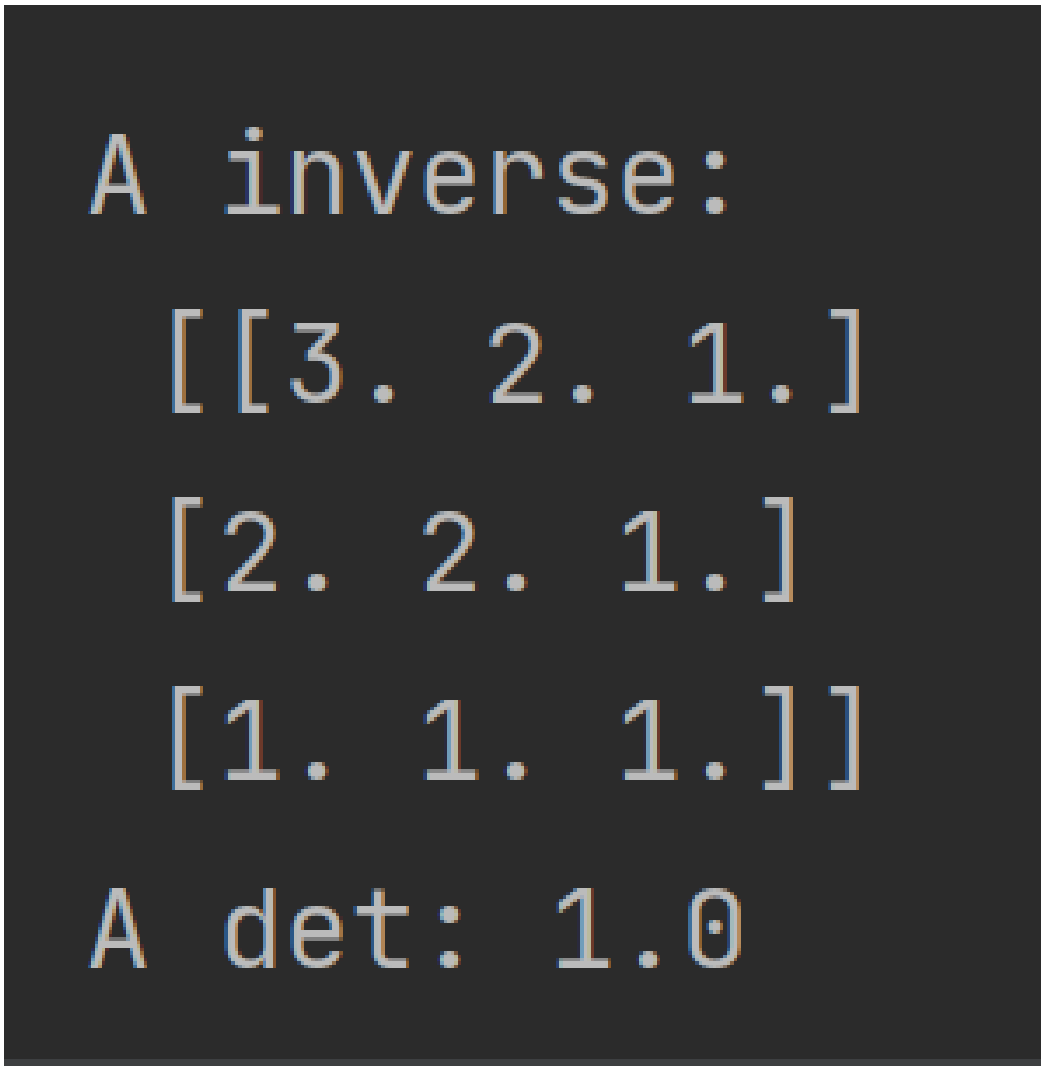
Problem 2

Note: the solution to this problem was found at the back of the textbook so it will be adapted for the provided solution below
Another note: we can use code to complete parts (a) - (c) so they will be coded up in `Python` and a summary of the results will be given below (please see the code for implementation details)

We are given

$$A = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 1 & -1 & 0 \end{bmatrix}$$

(a) We are asked if A is invertible. An easy way to check is to find the determinant of A since a nonzero determinant would show that A is invertible. However, the inverse itself could also be found since the `numpy.linalg.inv` function will throw an error if the input matrix A is singular. Since we found the determinant of A to be 1 (and no error was produced), A is indeed invertible.



```
A inverse:
 [[3. 2. 1.]
 [2. 2. 1.]
 [1. 1. 1.]]
A det: 1.0
```

Figure 1: Inverse and determinant of A

(b) To find the spectral radius $\rho(M)$ for M, we first need to find M. For the Gauss-Seidel iteration, the equation for the iteration matrix M is

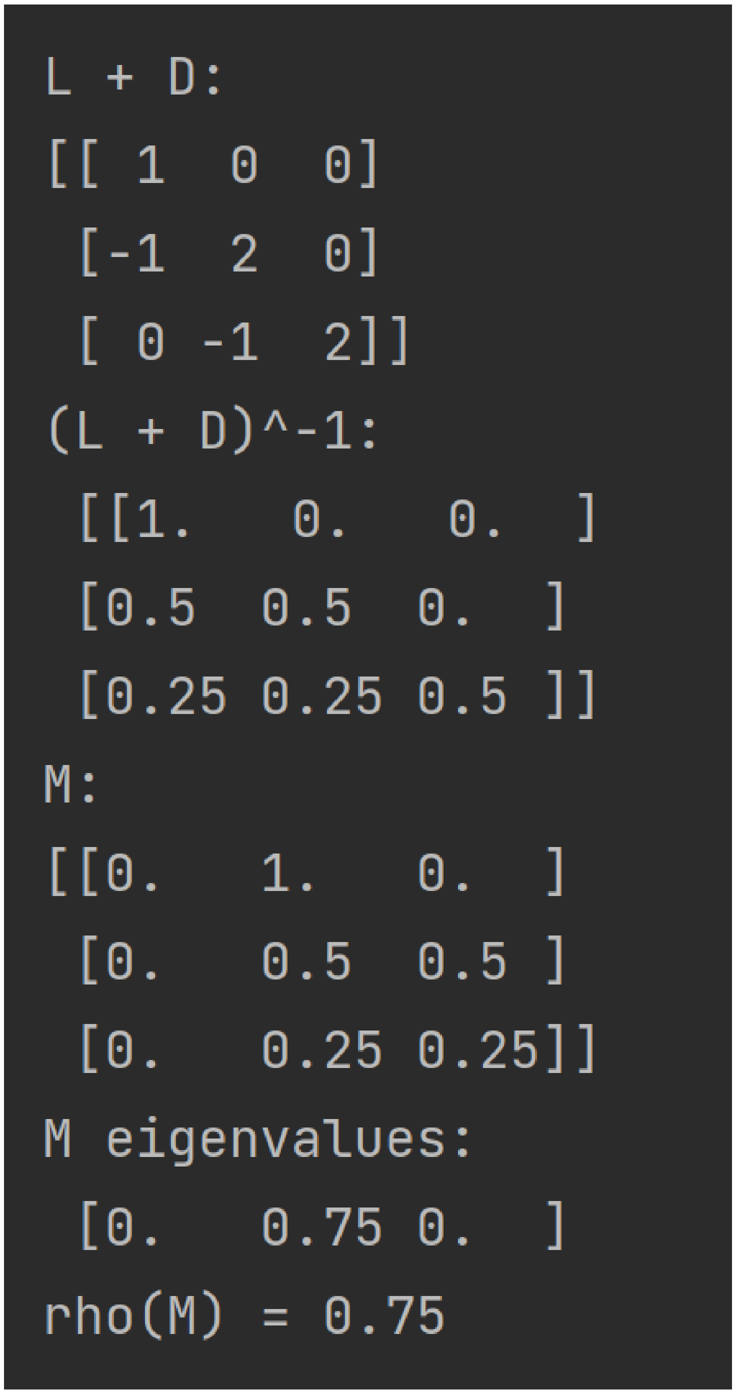
$$M = I - (L + D)^{-1}A$$

where L and D are the lower triangular (not including the diagonal entries) and diagonal matrices, respectively

From here, the spectral radius is

$$\rho(M) = \max\{|\lambda| : \lambda \in \mathbb{C} \text{ is an eigenvalue of } M\}$$

So, we would need to find the max absolute eigenvalue of M to find $\rho(M)$. We used *numpy.linalg.eig* to find the eigenvalues of M and then found the max absolute eigenvalue to be 0.75.

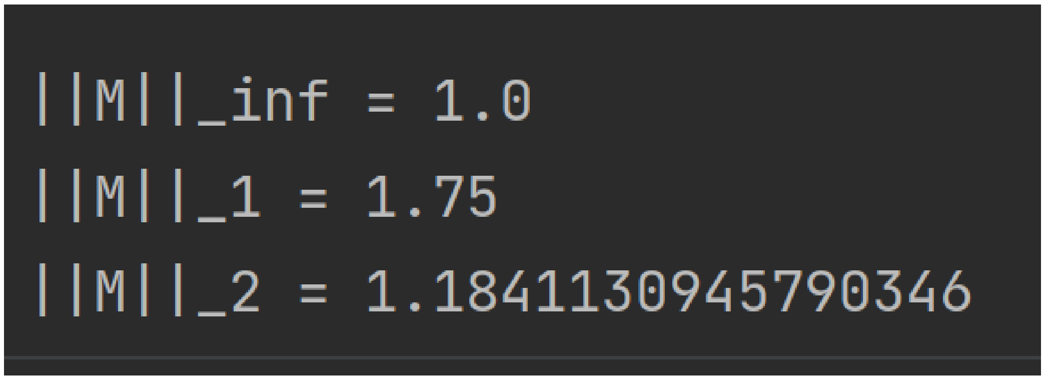


```
L + D:
[[ 1  0  0]
 [-1  2  0]
 [ 0 -1  2]]
(L + D)^-1:
 [[1.   0.   0.  ]
 [0.5  0.5  0.  ]
 [0.25 0.25 0.5 ]]
M:
[[0.   1.   0.  ]
 [0.   0.5  0.5 ]
 [0.   0.25 0.25]]
M eigenvalues:
 [0.   0.75 0.  ]
rho(M) = 0.75
```

Figure 2: Iteration matrix and spectral radius of A

(c) We are asked to find $\|M\|_\infty$, $\|M\|_1$, $\|M\|_2$ and they are calculated easy with `numpy.linalg.norm` and specifying which norm we want to find.

So, we got $\|M\|_\infty = 1$, $\|M\|_1 = 1.75$, $\|M\|_2 \approx 1.18$



```
||M||_inf = 1.0
||M||_1 = 1.75
||M||_2 = 1.1841130945790346
```

Figure 3: Different matrix norms of A

(d) Based on our answers for the above parts, the Gauss-Seidel iteration will converge. For part (c) we found that all three tested norms resulted in a norm ≥ 1 which does not satisfy the premise to the condition that if there exists a norm < 1, then it converges. However, we have to be careful about the implication. Even though the premise is not satisfied, this statement does not say anything about for norms ≥ 1 (i.e., if it does not converge).

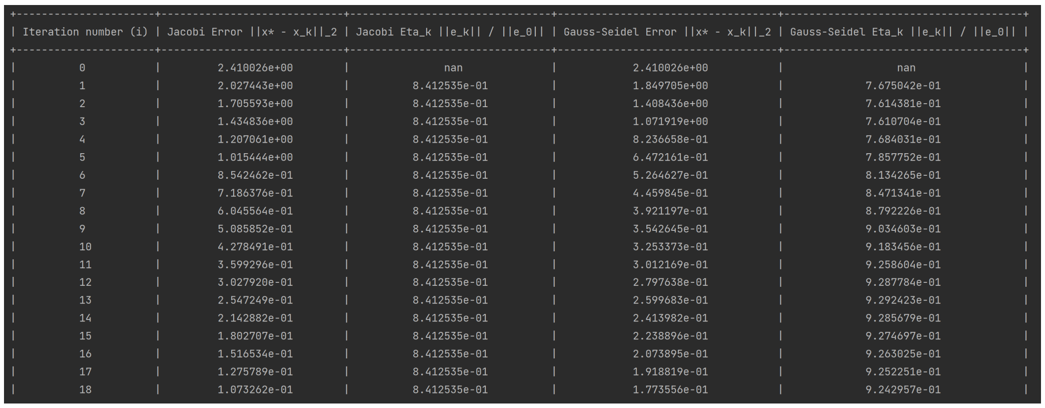
This is evident in another theorem we covered where a linear iterative method (including Gauss-Seidel) will converge if and only if it has a spectral radius < 1. So from our result for part (b), we do have a spectral radius of 0.75 which is indeed less than 1. So, this method will converge even though we have norms ≥ 1.

Problem 3

For this problem, parts (a) - (d) were assumed to be implementation problems and a summary of the results along with responses to parts (d) and (e) are given below.

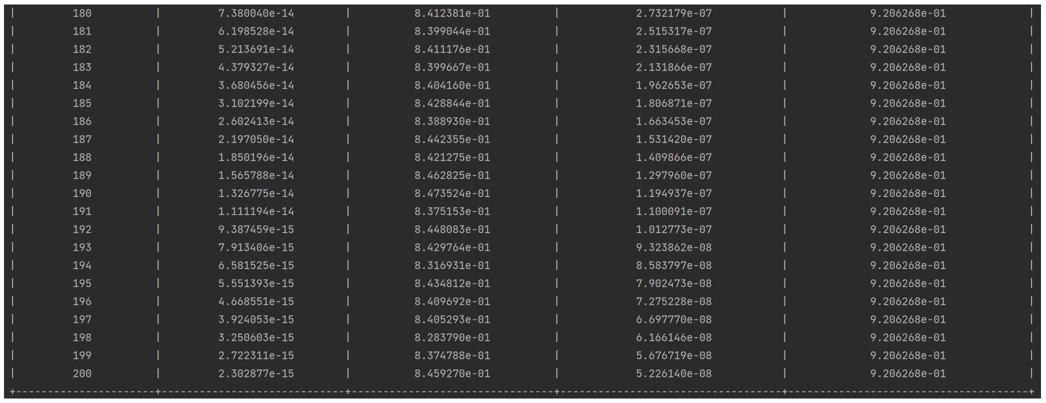
For part (c), we are asked to experiment with different array sizes $N = 10, 20, 40, 80, 160$ for both the Jacobi and Gauss-Seidel methods to solve the linear system as shown in the Poisson 1D code given to us – a tridiagonal matrix with 2s on the main diagonal and -1s on the sub and super diagonals (0s elsewhere). So we stored the matrix as a sparse matrix using the `scipy.sparse` packages in `Python`. Again, see the code for more details.

After testing out the various array sizes, we found that the 10×10 size produced relatively good convergence rates while all other sizes were markedly slower (i.e., final errors around $1e-4$ to $1e-1$ or even higher for the larger array sizes). So, the table detailing the iteration number and the corresponding errors and relative error change in between iterations (i.e., $\eta_{k+1} = \|e_{k+1}\|/\|e_k\|$ where a typo was given in the provided equation for η_{k+1} showing the relative error change with respect to the initial error instead of the previous iteration error) with array size 10×10 for Jacobi and Gauss-Seidel methods are shown below



```
+----------------------+-----------------------------+--------------------------------+-----------------------------------+--------------------------------------+
| Iteration number (i) | Jacobi Error ||x* - x_k||_2 | Jacobi Eta_k ||e_k|| / ||e_0|| | Gauss-Seidel Error ||x* - x_k||_2 | Gauss-Seidel Eta_k ||e_k|| / ||e_0|| |
+----------------------+-----------------------------+--------------------------------+-----------------------------------+--------------------------------------+
|          0           |        2.410026e+00         |              nan               |           2.410026e+00            |                 nan                  |
|          1           |        2.027443e+00         |          8.412535e-01          |           1.849705e+00            |             7.675042e-01             |
|          2           |        1.705593e+00         |          8.412535e-01          |           1.408436e+00            |             7.614381e-01             |
|          3           |        1.434836e+00         |          8.412535e-01          |           1.071919e+00            |             7.610704e-01             |
|          4           |        1.207061e+00         |          8.412535e-01          |           8.236658e-01            |             7.684031e-01             |
|          5           |        1.015444e+00         |          8.412535e-01          |           6.472161e-01            |             7.857752e-01             |
|          6           |        8.542462e-01         |          8.412535e-01          |           5.264627e-01            |             8.134265e-01             |
|          7           |        7.186376e-01         |          8.412535e-01          |           4.459845e-01            |             8.471341e-01             |
|          8           |        6.045564e-01         |          8.412535e-01          |           3.921197e-01            |             8.792226e-01             |
|          9           |        5.085852e-01         |          8.412535e-01          |           3.542645e-01            |             9.034603e-01             |
|          10          |        4.278491e-01         |          8.412535e-01          |           3.253373e-01            |             9.183456e-01             |
|          11          |        3.599296e-01         |          8.412535e-01          |           3.012169e-01            |             9.258604e-01             |
|          12          |        3.027920e-01         |          8.412535e-01          |           2.797638e-01            |             9.287784e-01             |
|          13          |        2.547249e-01         |          8.412535e-01          |           2.599683e-01            |             9.292423e-01             |
|          14          |        2.142882e-01         |          8.412535e-01          |           2.413982e-01            |             9.285679e-01             |
|          15          |        1.802707e-01         |          8.412535e-01          |           2.238896e-01            |             9.274697e-01             |
|          16          |        1.516534e-01         |          8.412535e-01          |           2.073895e-01            |             9.263025e-01             |
|          17          |        1.275789e-01         |          8.412535e-01          |           1.918819e-01            |             9.252251e-01             |
|          18          |        1.073262e-01         |          8.412535e-01          |           1.773556e-01            |             9.242957e-01             |
```

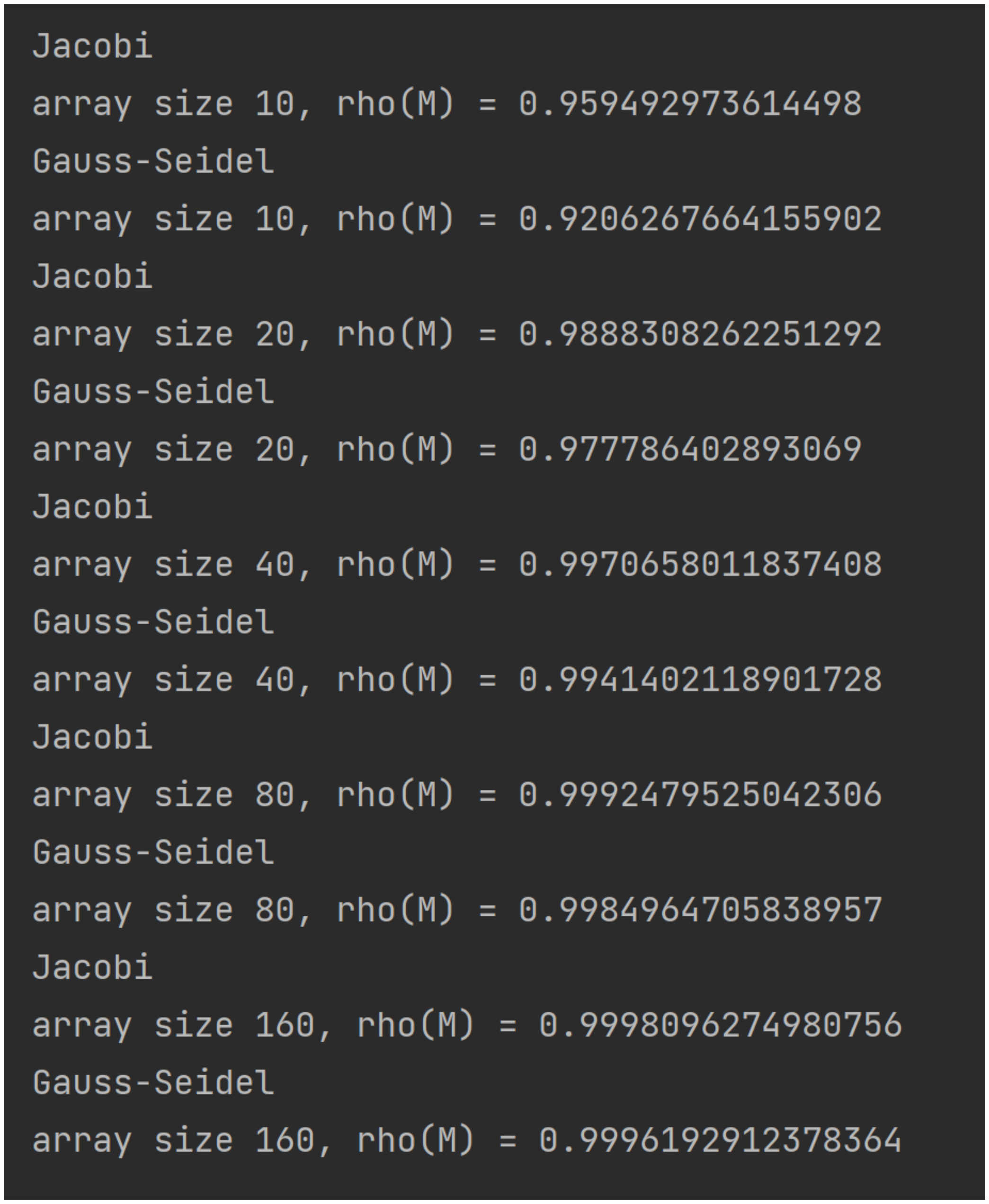
Figure 4: First few iterations with results for Jacobi and Gauss-Seidel methods



```
|        180        |        7.380040e-14        |        8.412381e-01        |        2.732179e-07        |        9.206268e-01        |
|        181        |        6.198528e-14        |        8.399044e-01        |        2.515317e-07        |        9.206268e-01        |
|        182        |        5.213691e-14        |        8.411176e-01        |        2.315668e-07        |        9.206268e-01        |
|        183        |        4.379327e-14        |        8.399667e-01        |        2.131866e-07        |        9.206268e-01        |
|        184        |        3.680456e-14        |        8.404160e-01        |        1.962653e-07        |        9.206268e-01        |
|        185        |        3.102199e-14        |        8.428844e-01        |        1.806871e-07        |        9.206268e-01        |
|        186        |        2.602413e-14        |        8.388930e-01        |        1.663453e-07        |        9.206268e-01        |
|        187        |        2.197050e-14        |        8.442355e-01        |        1.531420e-07        |        9.206268e-01        |
|        188        |        1.850196e-14        |        8.421275e-01        |        1.409866e-07        |        9.206268e-01        |
|        189        |        1.565788e-14        |        8.462825e-01        |        1.297960e-07        |        9.206268e-01        |
|        190        |        1.326775e-14        |        8.473524e-01        |        1.194937e-07        |        9.206268e-01        |
|        191        |        1.111194e-14        |        8.375153e-01        |        1.100091e-07        |        9.206268e-01        |
|        192        |        9.387459e-15        |        8.448083e-01        |        1.012773e-07        |        9.206268e-01        |
|        193        |        7.913406e-15        |        8.429764e-01        |        9.323862e-08        |        9.206268e-01        |
|        194        |        6.581525e-15        |        8.316931e-01        |        8.583797e-08        |        9.206268e-01        |
|        195        |        5.551393e-15        |        8.434812e-01        |        7.902473e-08        |        9.206268e-01        |
|        196        |        4.668551e-15        |        8.409692e-01        |        7.275228e-08        |        9.206268e-01        |
|        197        |        3.924053e-15        |        8.405293e-01        |        6.697770e-08        |        9.206268e-01        |
|        198        |        3.250603e-15        |        8.283790e-01        |        6.166146e-08        |        9.206268e-01        |
|        199        |        2.722311e-15        |        8.374788e-01        |        5.676719e-08        |        9.206268e-01        |
|        200        |        2.302877e-15        |        8.459270e-01        |        5.226140e-08        |        9.206268e-01        |
+-------------------+----------------------------+----------------------------+----------------------------+----------------------------+
```

Figure 5: Last few iterations (up to 200) with results for Jacobi and Gauss-Seidel methods

The greatly reduced convergence rate may be attributed to what we spectral radii we found for each of the array sizes. Before we go over that, it is important to note the relative error change for the 10×10 above. The relative error factor for Jacobi was lower which meant that each iteration reduced the error more than Gauss-Seidel, which is what we observe with the final errors. However, the corresponding spectral radius for Jacobi was calculated to be higher than what was observed in the relative error change (which was observed for the other array sizes as well, not only 10×10). Overall, Jacobi outperforms Gauss-Seidel in this particular case of A but Gauss-Seidel has a slight edge as the array size increases (though, the rate of convergence is reduced significantly). The spectral radii for various array sizes for Jacobi and Gauss-Seidel are shown below



```
Jacobi
array size 10, rho(M) = 0.959492973614498
Gauss-Seidel
array size 10, rho(M) = 0.9206267664155902
Jacobi
array size 20, rho(M) = 0.9888308262251292
Gauss-Seidel
array size 20, rho(M) = 0.977786402893069
Jacobi
array size 40, rho(M) = 0.9970658011837408
Gauss-Seidel
array size 40, rho(M) = 0.9941402118901728
Jacobi
array size 80, rho(M) = 0.9992479525042306
Gauss-Seidel
array size 80, rho(M) = 0.9984964705838957
Jacobi
array size 160, rho(M) = 0.9998096274980756
Gauss-Seidel
array size 160, rho(M) = 0.9996192912378364
```

Figure 6: Spectral radii for various array sizes for Jacobi and Gauss-Seidel methods

As we can see, the spectral radii for all of the array sizes are indeed less than 1 to guarantee convergence, but as the array size increases, the spectral radii gets closer to 1 and significantly reduces limit on the converge rate. We can clearly see for Gauss-Seidel above for the 10×10 case that the relative error change approaches the corresponding spectral radius rather quickly, unlike Jacobi as mentioned above.