Task 3 Analysis

Moritz Jasper Techen, Paul Kang, Bailiang Li, Max Spanning

December 5, 2024

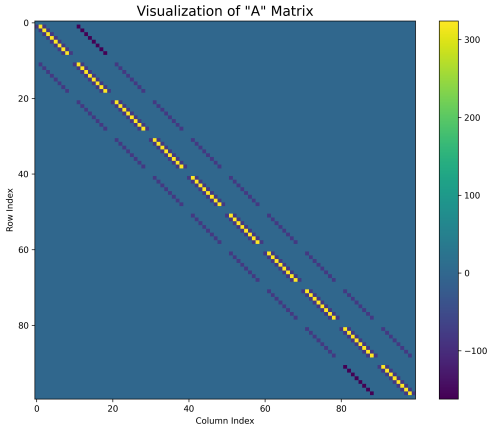

Figure 1: Sample Visualization of Diagonally Dominant Generated Matrix by Solver Generated by 'matrix.py' Using a Resolution of 10 ($10^2 \times 10^2$ matrix)

Parameter Name	*Value*
Resolution	128
Number of Iterations	10, 100, 1000, 10000, 50000
Values for the Dirichlet Boundary Condition on left and right side	0,0
Values for the Neumann Boundary Condition	0
(x,y) center coordinates for the Gaussian Source	(0.5,0.5)
Deviation of the Gaussian Source	0.1

Table 1: Parameters used for Solving the Poisson Equation ($-\Delta\vec{u} = \vec{b}, \vec{b} =$ source function) via Finite Discretization with the Jacobi Matrix Method

1 Discussion of Converged Solution Plot

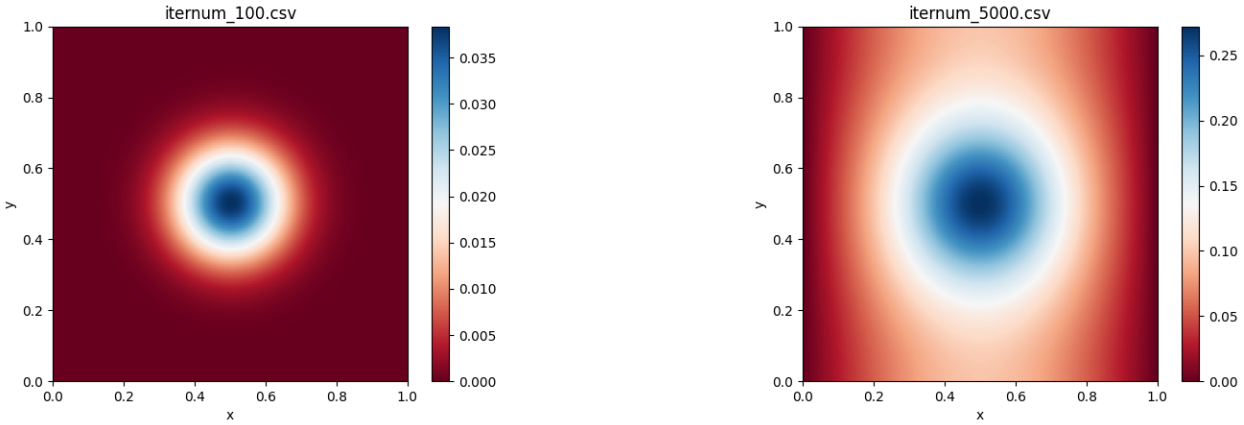

Figure 2: Converged Solution Heatmap after 100 and 5000 Iterations Generated by 'plotty.py'

A couple of key characteristics can be observed from Figure 2 that show signs of correct behavior. The circular behavior of the figure above can be immediately noticed and is to be expected because the form of the Poisson equation is reminiscent of the simple 2D formula of a circle: $x^2 + y^2 = r^2$. Other immediate physical characteristics is that the solution is centered at $(0.5, 0.5)$, which is the center that was set for the Gaussian source function used for this task. This qualitative characteristic is capture by both low and high iterations of the solver code. Examining the edges of the heatmap reveal that the boundary conditions have also been fulfilled. Along $x = 0, 1$, the values are the Dirichlet values assigned. Because this was "forced" within the code, both low and high iterations are expected to capture this characteristic. However, where the two plots begin to deviate is when considering the implemented Neumann boundary conditions. The plot for the lower number of iterations shows no flux going through **all** edges but larger iterations show that the boundary only restricts flux normal to the y-axis but correctly allows flux normal to the x-axis. As a result, the behavior of the true solution can be observed from the solver at higher iterations.

2 Discussion of Residual Norm Plots

$$\text{residual} = |A\vec{u} - \vec{b}| \tag{1}$$

Figure 3: Formula for Calculation of Residuals

$$u_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{i \neq j} a_{ij} u_j^{(k)}) \tag{2}$$

Figure 4: Element Based Formula for Jacobi Matrix Method (https://en.wikipedia.org/wiki/Jacobi_method)

In order to examine the error in the converged solution by number of iterations, the 2-norm and infinite norm (inf-norm) produced by the residuals were examined. The residual values were obtained by obtained using the equation provided in Figure 3. In this case, the A matrix consisted of constants formed a 5-point stencil that approximated the laplacian operator acting on $\vec{u}$, which is the solution vector of interest. 2 norms were used on the residual vector to examine behavior of the overall error (2-norm) and its maximum possible error (inf-norm) of the finite discretization method over increasing N iterations (refer to Table 1 for specific parameters used). Because the impact of varying iteration number was examined for this task, convergence depends soley on the diagonalization by the Jacobi method used within the solver. Both plots show no erratic behavior and the negative linear slopes indicate that the solver shows good logarithmic convergence as the

number of iterations increase. This is further reinforced by the same extrapolated slopes, which show agreement that both residuals converge as iterations increase.

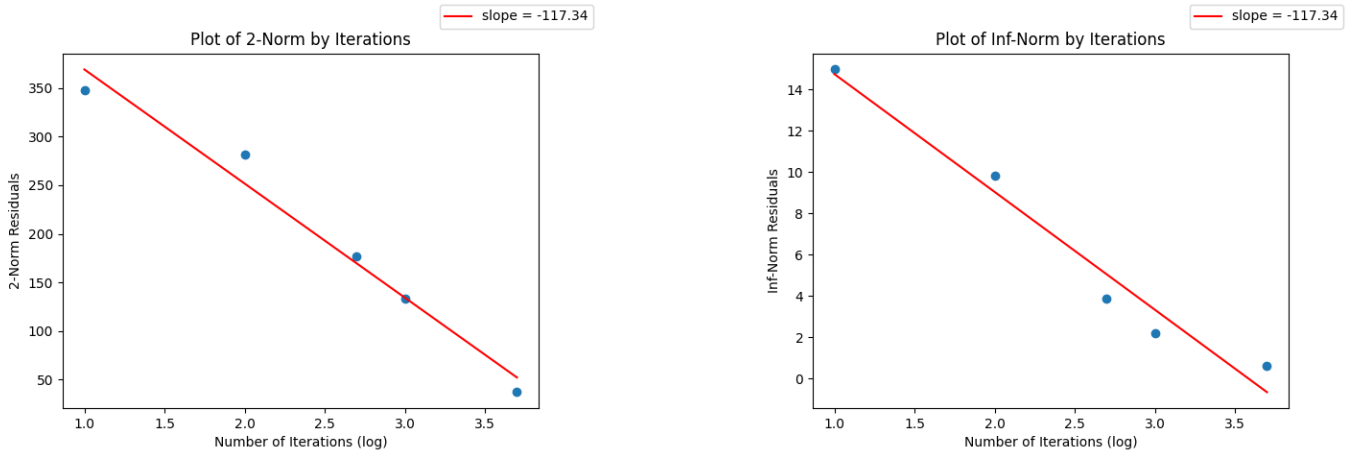

Figure 5: Plots of the Residual Norms by Number of Iterations (Left: 2-norm, Right: inf-norm) Generated by 'plotty.py'