# Computation PDEs: Project 2 Laplace's Equation in an Annulus

Yadu *Bhageria*

The problem of interest is

$$0 = \nabla^2 u = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} \quad \text{in} \quad 1 < r < b, \quad 0 < \theta < 2\pi \tag{1}$$

subject to three boundary conditions

a. The Dirichlet boundary condition that the value at $r = 1$ is fixed.

$$u(1,\theta) = f(\theta) \tag{2}$$

b. The Neumann boundary condition that the derivative of $u$ at the $r = b$ boundary is 0.

$$\frac{\partial u}{\partial r}(b,\theta) = 0 \tag{3}$$

c. The periodic boundary condition as the annulus is circular. i.e. $\theta = \theta + 2\pi$

The 2D grid is set up in the $r - \theta$ plane with distances in terms of $(\delta r, \delta\theta)$ where $M\delta r = b - 1$ and $N\delta\theta = 2\pi$. Thus the solution is stored on a $(M+1) \times (N)$ grid. So the Dirichlet boundary a. is enforced by not changing the initial conditions at $u_{1,:} = f(\theta)$. Then the Neumann boundary b. is enforced by introducing another row of points at $M + 2$ that mirrors the values at $M$, i.e. $u_{M+2,:} = u_{M,:}$. Finally, the periodicity boundary c. is enforced by introducing a column of point at $u_{:,N+1}$ making sure $u(r,0) = u(r,2\pi)$ which is enforced by $u_{:,N+1} = u_{:,1}$.

The centred finite difference approximation of the problem is

$$\frac{u_{m+1,n} + u_{m-1,n} - 2u_{m,n}}{\delta r^2} + \frac{1}{r}\frac{u_{m+1,n} - u_{m-1,n}}{2\delta r} + \frac{1}{r^2}\frac{u_{m,n+1} + u_{m,n-1} - 2u_{m,n}}{\delta\theta^2} = 0 \tag{4}$$

1. For my trial function proportional to $\sin(p\theta)$ I have chosen

$$u(r,\theta) = \frac{(r-b)^2}{r}\sin(p\theta) \tag{5}$$

with a chosen value of 5 for b. So for this function

$$
\begin{aligned}
u(1,\theta) &= (1-b^2)sin(p\theta) \\
\frac{\partial u}{\partial r} &= (1-\frac{b^2}{r^2})\sin(p\theta) \implies \frac{\partial u}{\partial r}(b,\theta) = 0 \\
\frac{\partial^2 u}{\partial r^2} &= \frac{2b^2}{r^3}\sin(p\theta) \\
\frac{\partial^2 u}{\partial \theta^2} &= -\frac{(r-b)^2}{r}p^2\sin(p\theta)
\end{aligned}
$$

which gives the source term

$$
S(r,\theta) = -\frac{\sin(p\theta)}{r}[(\frac{b^2}{r^2}+1)(1-p^2)+\frac{2bp^2}{r}]
$$

I now try and solve $\nabla^2 u = S(r,\theta)$ iteratively using the Jacobi-iteration scheme. The problem has been set up in `q1.m` and solved using `Jacobi.m`. Convergence of the result is assessed by looking at the residual of the iterative scheme. If it is under a prescribed tolerance level then I stop iterating. To assess the accuracy of the results, the average error has been recorded, i.e. the average difference between the true solution and the computed solution over all the solution space $M+1\times N$.

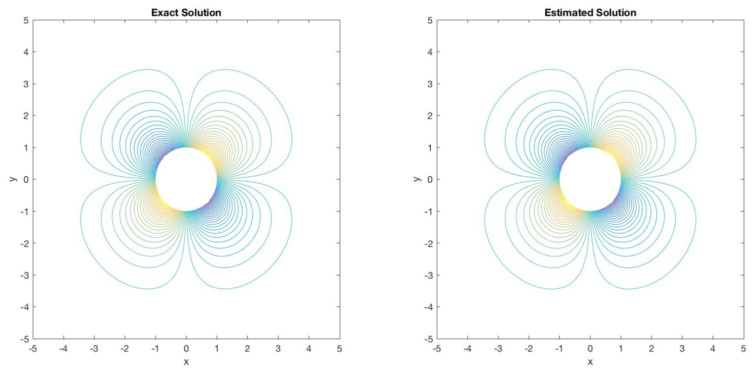

Figure 1: Contours of the exactly solution and estimated solution using the Jacobi iterative method. $p=2$ and $b=5$

The accuracy results for Jacobi iteration show that increasing $M$ or $N$ decreases the error. This makes sense because as we increase the number of points on our grid, we would expect the average error to decrease. But what is of more interest is the fact that increasing the side of $N$ relative to $M$ decreases the average Error much faster than the other way around. This suggests that a smaller ratio of $M/N$ is ideal for a grid with the same number of points.

| M \ N | 16 | 32 | 64 | 128 |
|---|---|---|---|---|
| 16 | 0.0791 | 0.0340 | 0.0223 | 0.0194 |
| 32 | 0.0687 | 0.0212 | 0.0089 | 0.0058 |
| 64 | 0.0663 | 0.0179 | 0.0054 | 0.0023 |
| 128 | 0.0658 | 0.0171 | 0.0045 | 0.0014 |

Table 1: Average Error between the True and Estimated solution using Jacobi Iteration for various grid sizes

Figure 2 shows the average error for various ratios of $M/N$. It is clear that there is a minimum near $M/N = 0.45$. Reasonably, the error does not continue to decrease for ever smaller ratios of $M/N$ as after a certain point the effect of having higher resolution in $\theta$ is outweighed by the coarseness in resolution in $r$.

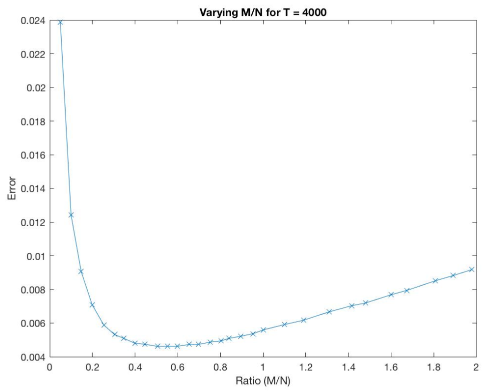

Figure 2: A graph for various ratios of $M/N$ when the total number of points, $T$, remains at roughly 4000. $p = 2$ and $b = 5$. A minimum at $M/N \approx 0.45$

Remembering that we are working in the $r - \theta$ plane, we note larger values of $N$ mean that the grid we are working on has finer divisions in $\theta$ and thus locally has less curvature which agrees with our results of small ratio of $M/N$ working well.

Another point of interest is the impact of the choice of $p$ in the test function on the ideal ratio of $M/N$. Looking at Figure 2 and 3, it seems like the ideal ratio of $M/N$ increases as $p$ decreases. Once again, thinking about the curvature of the space and how the periodicity of the test function sheds light on the matter. For larger values of $p$ the test function repeats itself more frequently in the $\theta$ direction. This means that more resolution is needed to get an accurate result using a centred difference scheme.

2. In this section Poisson's equation is solved using the Successive Over

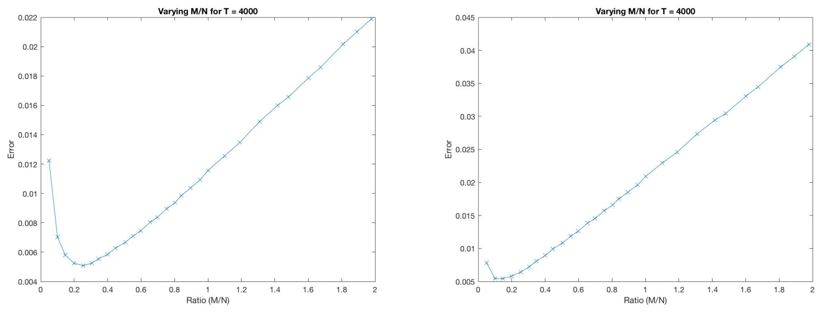

Figure 3: A graph for various ratios of $M/N$ when the total number of points, $T$, remains at roughly 4000 for $p = 3$ (left) and $p = 4$ (right) with $b = 5$.

Relaxation (SOR) iterative scheme. This can be written, for the annulus as

$$\begin{aligned} u_{m,n}^{j+1} = (1-\omega)u_{m,n}^{j} + \frac{\omega}{\frac{2}{\delta r^2} + \frac{1}{r^2}\frac{2}{\delta \theta^2}}[(&\frac{1}{\delta r^2} + \frac{1}{r}\frac{1}{2\delta r})u_{m+1,n}^{j} + \\ (&\frac{1}{\delta r^2} - \frac{1}{r}\frac{1}{2\delta r})u_{m-1,n}^{j+1} + \\ (&\frac{1}{r^2}\frac{1}{\delta \theta^2})(u_{m,n+1}^{j} + u_{m,n-1}^{j+1})] \end{aligned} \tag{6}$$

Looking at Table 2, it can be seen that for the same tolerance condition for convergence and the same test function as in the previous section, SOR gives exactly the same results as the Jacobi iterative scheme. But it is much faster. Before doing a comparison of the iterations required for convergence in the Jacobi and SOR schemes, we must first find out the optimal value of $\omega$ parameter for SOR.

| M \ N | 16 | 32 | 64 | 128 |
|---|---|---|---|---|
| 16 | 0.0791 | 0.0340 | 0.0223 | 0.0194 |
| 32 | 0.0687 | 0.0212 | 0.0089 | 0.0058 |
| 64 | 0.0663 | 0.0179 | 0.0054 | 0.0023 |
| 128 | 0.0658 | 0.0171 | 0.0045 | 0.0014 |

Table 2: Average Error between the True and Estimated solution using SOR Iteration for various grid sizes

The optimal value of $\omega$ is near $\omega \approx 2$ but definitely not $\omega > 2$ or else the scheme becomes unstable. This can be seen in Figure 4 where the number of iterations needed are much larger for smaller values of $\omega$. Doing a finer linear search between $1.9 and 2.0$ gives the results for table 3. It can be seen that there is some variation in the optimal value of $\omega$ for various values of $M$ and $N$.

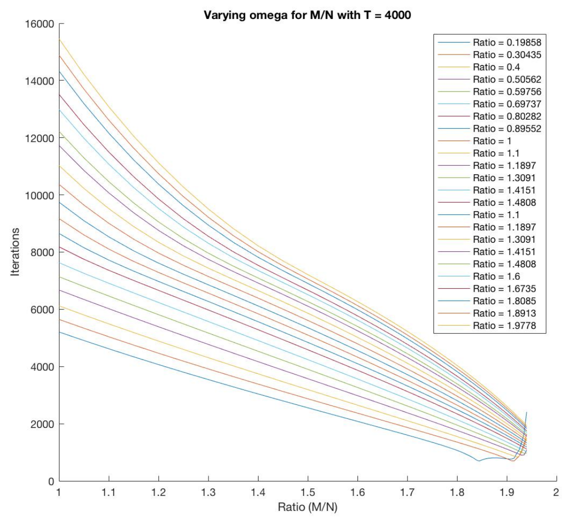

Figure 4: Number of iterations vs $\omega$ when the total number of points, $T$, remains at roughly 4000 with lines representing various ratios of $M/N$

3. The Multigrid solver for Poisson's Equation in the annulus is contained in various values. `FullMG.m` takes a boundary condition vector for the Dirichlet condition and matrix $f$ that would solve $\nabla^2 u = f$. It then interpolates from the coarsest grid to the finest grid to find an optimal initial guess. It then performs 1 complete V-cycle to smooth out the errors by calling `MultiGridV.m`. In each step of the algorithm 10 iterations of Gauss-Seidel are applies for error smoothing. `interpolate.m` and `restrict.m` help transform the residuals between grid sizes using a equal weighting nearest neighbour routine. `FMGV.m` performs a given number of MultiGrid V cycles.

   I have implementation the MultiGrid algorithm to solve the same problem as in the previous parts to check it is accuracy and indeed it is exactly the same as Jacobi and SOR. This can be seen in Table BLAH.

   Looking at the Table BLAH it is clear that the Jacobi method is the slowest out of all 3 methods for all situations. Just this method should never be utilized unless some form of paralellization is utilized for speed-up. Note that SOR cannot be parallelized and since MultiGrid uses Gauss-Seidel it too cannot be parallelized. In our case however, MultiGrid performs significantly better than SOR (and of course Jacobi) for square grid sizes but

| M \ N | 16 | 32 | 64 | 128 |
|---|---|---|---|---|
| 16 | 0.0791 | 0.0340 | 0.0223 | 0.0194 |
| 32 | 0.0687 | 0.0212 | 0.0089 | 0.0058 |
| 64 | 0.0663 | 0.0179 | 0.0054 | 0.0023 |
| 128 | 0.0658 | 0.0171 | 0.0045 | 0.0014 |

Table 3: Average Error between the True and Estimated solution using the MultiGrid Method for various grid sizes

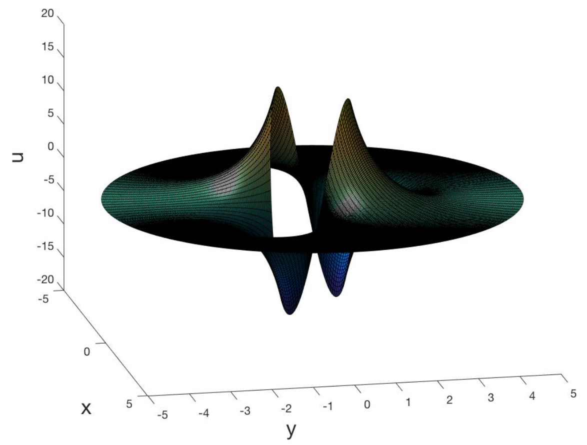

Figure 5: A plot of the solution of the Test Problem in 3D space with the X-Y axis and the value of $u(r, \theta)$ on the Z axis

not performs worse for large $M$ and small $N$ and on par with Gauss-Seidel for large $N$ and small $M$. This can be attributed to two issues. Firstly for non-square grid sizes, MultiGrid cannot go to the coarsest possible level in both dimension. Secondly, the way our interpolation and restriction function is set up, i.e. with a equal weighting nearest neighbour routine, favours grids that don't have large differences between the distances between grid points in each dimension.

4. Given that all 3 algorithm are equally accurate based on the previous tests, I have used a MultiGrid routine to compute the solution for a grid of size $256 \times 256s$ and $b = 5$. Complications might arise when dealing with the second problem (b) because the Dirichlet boundary condition is not $2\pi$ periodic as it is of the form $\sin(p\theta)$ but now $p$ is not an integer.

   Produced in `q4a.m` and `q4b.m`

   Looking at the contour and 3D graphs it can be seen that our code does well to model the first scenario. But in the second case where the bound-

| M \ N | 16 | 32 | 64 | 128 |
|---|---|---|---|---|
| 16 | 0.0212 | 0.0243 | 0.0556 | 0.2194 |
| 32 | 0.0825 | 0.1543 | 0.3196 | 0.8266 |
| 64 | 0.5605 | 1.0496 | 2.0788 | 4.3562 |
| 128 | 3.8748 | 7.0303 | 13.6241 | 27.8593 |

| M \ N | 16 | 32 | 64 | 128 |
|---|---|---|---|---|
| 16 | 0.0240 | 0.0466 | 0.0691 | 0.0717 |
| 32 | 0.0166 | 0.0323 | 0.0675 | 0.1217 |
| 64 | 0.0812 | 0.1641 | 0.3249 | 0.6028 |
| 128 | 0.3540 | 0.7159 | 1.4308 | 2.9065 |

| M \ N | 16 | 32 | 64 | 128 |
|---|---|---|---|---|
| 16 | 0.0100 | 0.0133 | 0.0225 | 0.0775 |
| 32 | 0.0438 | 0.0386 | 0.0748 | 0.1310 |
| 64 | 0.2786 | 0.1816 | 0.2171 | 0.4235 |
| 128 | 1.9909 | 1.2513 | 0.8228 | 1.3400 |

Table 4: Time using the Jacobi, SOR and then the Full MultiGrid Method for various grid sizes. Clearly Jacobi is the slowest and Full MultiGrid the fastest.

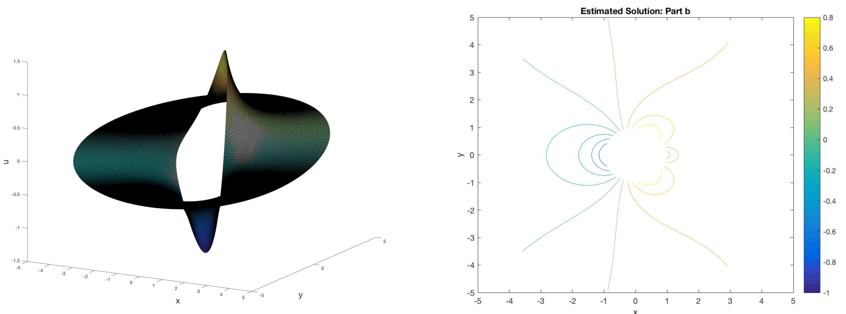

Figure 6: Solution for case (a)

ary condition is now $2\pi$ periodic a sharp point occurs at $u(1,0)$ which is not smooth. Looking at the boundary condition it can be seen that it is easy to visualize that why this effect has occurred. Considering only the values between $[0, 2\pi)$ the values on both sides of of $f(0)$ at positive. Ultimately, this is not necessarily a problem with the solution produced but rather with the given boundary function in that it has a non smooth point and thus is unlikely to simulate any kind of boundary to be present in a real world annulus.

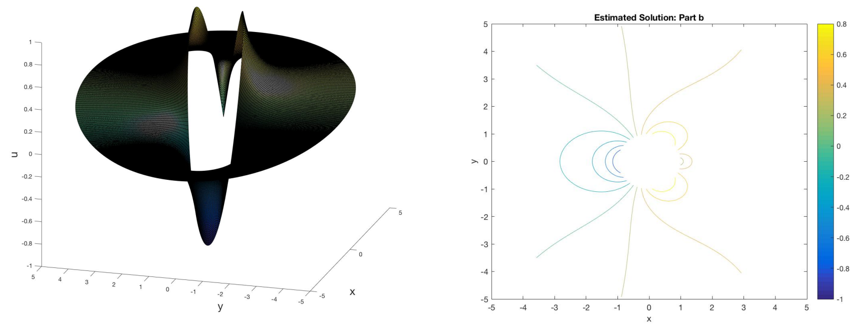

Figure 7: Solution for case (b)

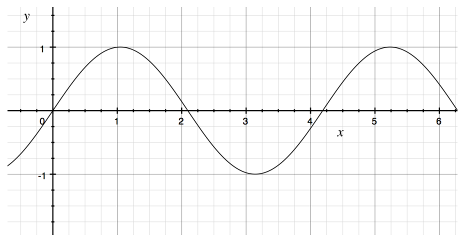

Figure 8: Boundary function for case (b)