MATH60025 Computational PDEs Coursework 2

CID: 02033585

1 Part A

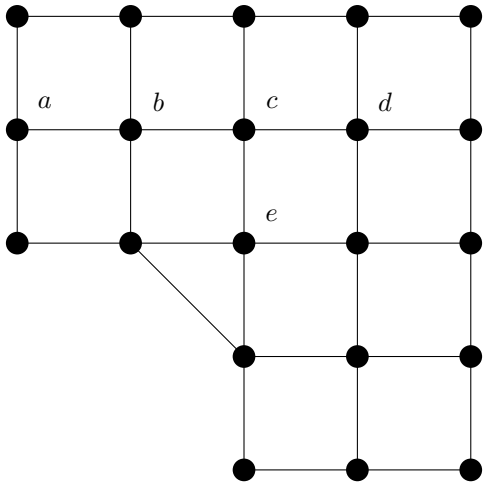


Figure 1: Coarse grid, with five unknown values.

Everywhere except for a, we use the second-order central differences δ_x^2 and δ_y^2 to approximate $u_{xx} + u_{yy}$.

$$\frac{1}{h^2}\left(4U_{i,j} - U_{i+1,j} - U_{i-1,j} - U_{i,j+1} - U_{i,j-1}\right) = 4 \tag{1.1}$$

At point a we enforce the boundary condition with the $O(h^2)$ accurate central difference (with i pointing in the positive x direction)

$$\frac{\partial u_{i,j}}{\partial x} = \frac{u_{i+1,j} - u_{i-1,j}}{2h} \tag{1.2}$$

This introduces a ficticious point to the left of a, which must have value equal to b by the above. Combined with the finite difference approximation for $u_{xx} + u_{yy} + 4$, we get the equation

$$4a - b - b - 0 - 1 = 4h^2 \tag{1.3}$$

We therefore get the system of equations for the five variables, where we used the symmetry of the grid (for example, the point to the right of e has the same value as c):

$$4a - b - b - 0 - 1 = 4h^2 \tag{1.4}$$
$$4b - a - c - 1 = 4h^2 \tag{1.5}$$
$$4c - b - d - e = 4h^2 \tag{1.6}$$
$$4d - 2c = 4h^2 \tag{1.7}$$
$$4e - 2c - 2 = 4h^2 \tag{1.8}$$

Equivalently in matrix form:

$$\begin{pmatrix} 4 & -2 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & -1 \\ 0 & 0 & -2 & 4 & 0 \\ 0 & 0 & -2 & 0 & 4 \end{pmatrix} U = \begin{pmatrix} 4h^2+1 \\ 4h^2+1 \\ 4h^2 \\ 4h^2 \\ 4h^2+2 \end{pmatrix} \tag{1.9}$$

1.1 Direct Method

Solving this system with $h = \frac{1}{2}$ yields the solution with direct method `direct_method_A` as

a	b	c	d	e
1.0	1.0	1.0	0.75	1.25

1.2 Jacobi Method

We iterate starting with $a_0, b_0, c_0, d_0, e_0 = 0$ and equations

$$4a_1 = 2 + b_0 \tag{1.10}$$
$$4b_1 = 2 + a_0 + c_0 \tag{1.11}$$
$$4c_1 = 1 + b_0 + d_0 + e_0 \tag{1.12}$$
$$4d_1 = 1 + 2c_0 \tag{1.13}$$
$$4e_1 = 3 + 2c_0 \tag{1.14}$$

Once all values are calculated, we replace a_0 with a_1, b_0 with b_1 etc. simultaneously, each iteration. If we let D be the diagonal part of A and C be the off-diagonal entries, we can rewrite this as the matrix system

$$U^{j+1} = D^{-1}(b - CU^j) \tag{1.15}$$

The function is located at `jacobi_method_A`.

1.3 Gauss-Seidel Method

We again iterate starting with $a, b, c, d, e = 0$ and equations, where we use new values as soon as possible.

$$4a = 2 + b \tag{1.16}$$
$$4b = 2 + a + c \tag{1.17}$$
$$4c = 1 + b + d + e \tag{1.18}$$
$$4d = 1 + 2c \tag{1.19}$$
$$4e = 3 + 2c \tag{1.20}$$

We define further matrices L and V as the lower and upper triangular parts of C. Using these matrices, the scheme can be written in matrix form as

$$U^{j+1} = (D + L)^{-1}(b - VU^j) \tag{1.21}$$

The function is written at `gauss_seidel_method_A`.

1.4 Successive over-relaxation method

The successive over-relaxation scheme is given by iterates

$$U^{j+1} = U^j + w(D + L)^{-1}r^j \tag{1.22}$$

where w is a parameter to be chosen (the case of Gauss-Seidel is given by $w = 1$) and r^j is the residual $b - AU^j$. The function is written at `SOR_method_A`.

1.5 Convergence of the three methods

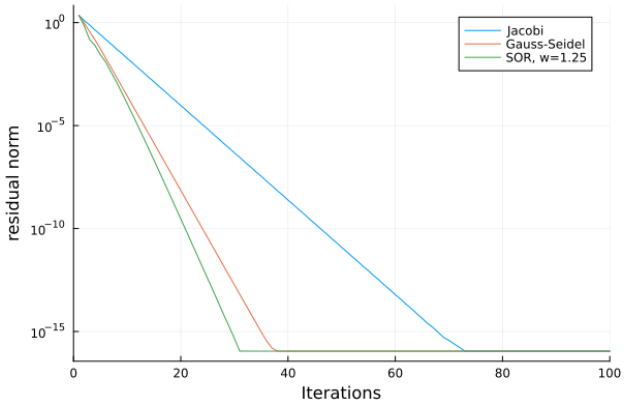


Figure 2: Plot of residual norm over iterations, for the three different iterative methods.

In Figure 2 we see that Jacobi reduces the residual norm by a factor of 10^{-15} in about 65 iterations, Gauss-Seidel in 40 and SOR, $w = 1.25$ in about 25. The following section will show how these values are a consequence of the different spectral radii of the iteration matrices.

For Jacobi, the iteration matrix is $D^{-1}(L+V)$, which has spectral radius $\rho_J \approx 0.5896$ (calculated using `spectral_radius`) in this case. Similarly, ρ_{GS} is calculated with $(D+L)^{-1}V$ and is approximately 0.3476.

To derive the iteration matrix for SOR, we start with the scheme and rearrange into a form such that $A = B + C$ with $BU^j = b - CU^j$.

$$U^{j+1} = U^j + w(D+L)^{-1}r^j \tag{1.23}$$

$$U^{j+1} = U^j + w(D+L)^{-1}b - w(D+L)^{-1}VU^j - wU^j \tag{1.24}$$

$$\frac{1}{w}(D+L)U^{j+1} = b - \left(\frac{w-1}{w}(D+L) + V\right)U^j \tag{1.25}$$

The iteration matrix can be read off as

$$(w-1)I + w(D+L)^{-1}V \tag{1.26}$$

For $w = 1.25$, the spectral radius is ≈ 0.25.

With these spectral radii, we can estimate the number of iterations needed to reduce the residual by a factor of 10^{-15}, since the magnitude of the error exponentially decreases. The formula is then

$$\text{iterations} \approx -\frac{15}{\log_{10}(\rho)} \tag{1.27}$$

Plugging in the three spectral radii, we get approximately 65 iterations for Jacobi, 33 iterations for Gauss-Seidel and 25 iterations for SOR, $w = 1.25$, which is very close to what we observe in Figure 2.

1.6 Optimal parameter for SOR

We can use the previous formula in Equation (1.26) to find the optimal parameter by minimising

$$\rho((w-1)I + w(D+L)^{-1}V) \tag{1.28}$$

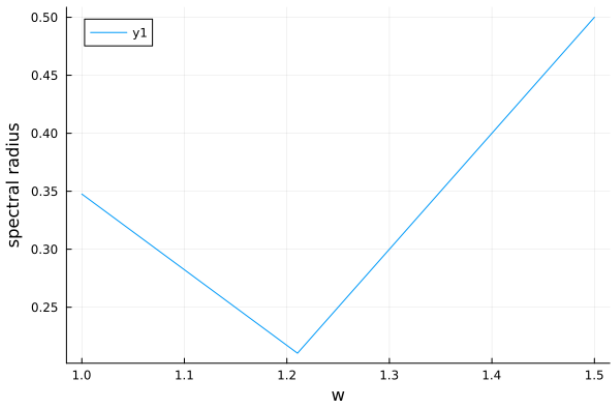


Figure 3: Plot of spectral radius of iteration matrix of SOR, with w varying.

Figure 3 shows the spectral radius as w varies. There is a clear minimum at around 1.2103, and we find that the spectral radius at this point is ≈ 0.2104. We plot below the convergence rate at this point, compared to neighbouring values of w. As expected w minimises the iteration count within a margin of error (Here, iteration count was measured by checking if the residual was below a tolerance 10^{-14}).

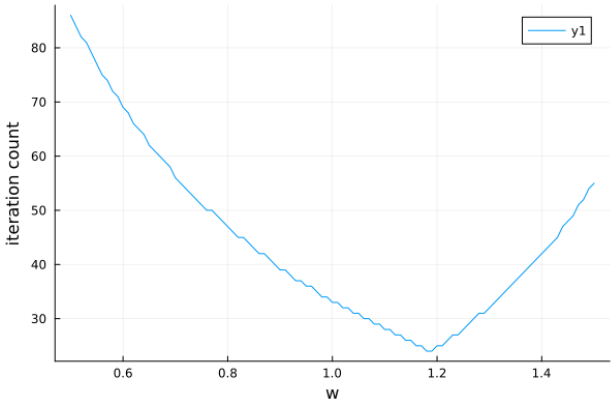


Figure 4: Plot of iteration count of SOR, with w varying.

2 Part B

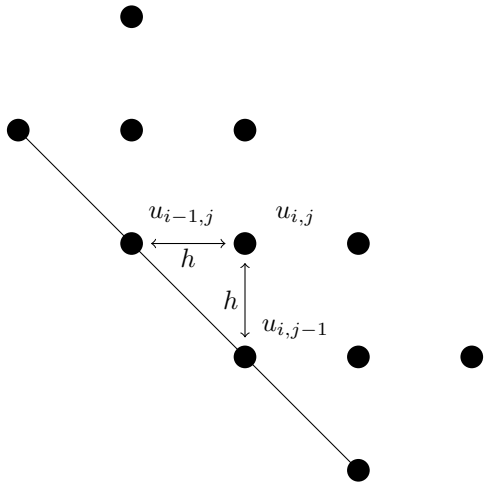


Figure 5: $q-v$ boundary discretisation in the case $\Delta x = \Delta y$. The values $u_{i-1,j}$ and $u_{i,j-1}$ are known exactly, as they lie directly on the boundary.

Throughout this part, we assume that Δx and Δy both divide into 0.5 an integer number of times, for simplicity. In addition, we also assume they are equal, to make use of the symmetry condition. The discretisation near the boundary therefore looks like Figure 5. In this case, for any point $u_{i,j}$ close to the boundary, $u_{i,j-1}$ and $u_{i-1,j}$ lie **exactly** on the boundary, and thus have known values. As such, we can apply the usual finite difference formula without worrying about irregular boundaries and adaptive stencils.

2.1 Gauss-Seidel (Single Grid)

Function written at `gauss_seidel_method_B` (as a special case of SOR), with parameter N detailing how many points in both x and y directions.

We solve for only the values on half of the domain, reflecting the values along the line of symmetry. Convergence is monitored using the pseudo-residual $\|u_{new} - u_{old}\|$, and we determine convergence when this residual is lower than 10^{-12}. A solution calculated using this method is contoured at Figure 10.

2.2 Successive Over-relaxation

Function written at `SOR_method_B`, with the same grid and convergence criterion as described before. We will determine the optimal w parameter in two ways.

2.2.1 Testing with a loose tolerance.

First, we estimate by testing different values w and recording the number of iterations it takes for the residual to drop below 10^{-3} (function at `optimal_SOR_parameter_test`). An example plot is given for different N values (Figure 6).

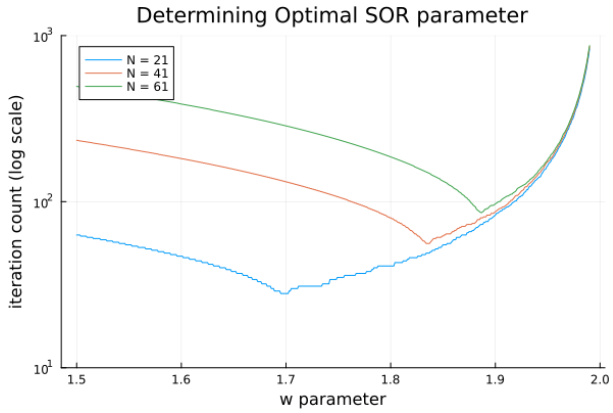


Figure 6: Iteration count as the SOR parameter w is varied, for different N.

The table below shows the optimal parameters obtained via this method (testing every value of w in the interval $[1.5, 1.99]$ with 0.001 spacing).

N	21	41	61	81	101
w_{opt}	1.695	1.834	1.886	1.913	1.929

2.2.2 Asymptotic spectral radius

Secondly, we estimate the optimal parameter by using the asymptotic formula for the spectral radius of the Gauss-Seidel iteration matrix, that is:

$$\rho_{GS} \approx 1 - \frac{\pi^2}{N^2} \tag{2.1}$$

$$w_{opt} \approx \frac{2}{1 + \sqrt{1 - \rho_{GS}^2}} \tag{2.2}$$

The above formulas were taken from the lecture notes (Function calculating this is written at `optimal_SOR_parameter_asym`). The second formula is only exact for real symmetric block tridiagonal matrices, which typically occur when the domain the PDE is being solved on is square. However, we will see later on that nevertheless, the formula is a good guess for the optimal parameter with a fraction of the computational cost.

2.2.3 Combination and comparison between methods

The first method comes with the disadvantage that it becomes expensive to compute for large N, and so it is desirable to reduce the size of the interval search space to speed up computation time. On the other hand, the asymptotic formula is easy to compute, but is only a heuristic.

This suggests a combination of the above two methods: use asymptotics to get a small interval in which to search for the optimal parameter.

Using this method, we can see how accurate the asymptotic formula is by plotting w_{opt} for larger N (Figure 7). We see that the asymptotic formula obtains a decent guess with barely any computational cost. Another thing we observe is that the optimal parameter seems to be approaching 2 with higher and higher N. This is a consequence of the spectral radius of the iteration matrix approaching 1 as $N \to \infty$ (the same reason the iterative method converges slower for large N).

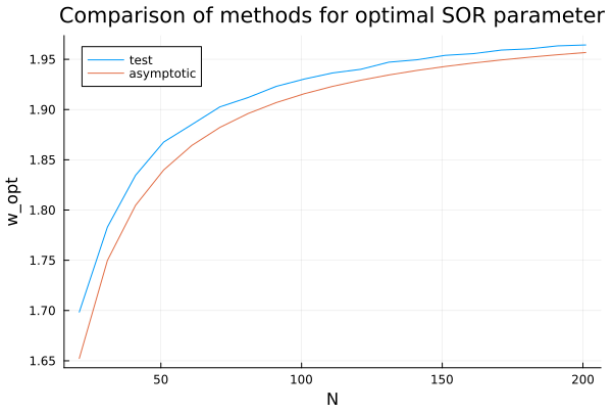
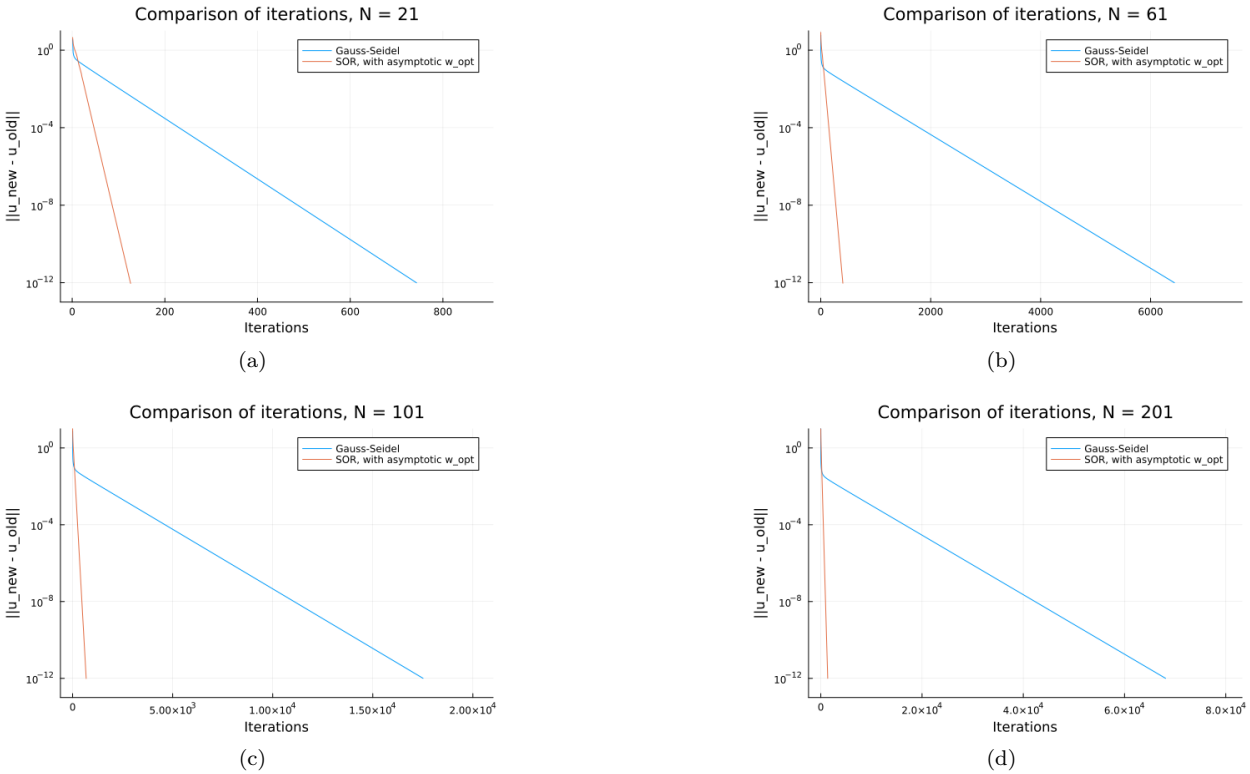


Figure 7: Comparison between the asymptotic value of w_{opt} with the value obtained from testing the convergence rate.

A solution calculated using this method is contoured at Figure 11.

2.3 Convergence rate of Gauss-Seidel and SOR as N varies

From the previous section, we know that the asymptotic formula for the spectral radius of the iteration matrix yields a good approximation of the optimal SOR parameter with little computational cost. As such, we will use it in our analysis of the convergence rate for SOR.



(a) (b) (c) (d)

Figure 8: Pseudo-residual plotted against iterations taken, for both Gauss-Seidel and SOR with our asymptotic formula for w_{opt}.

The table below details the iteration counts for both Gauss-Seidel and SOR with the asymptotic optimal

parameter for w. We note that SOR bring a very big improvement to the number of iterations required, especially for large N. The mathematical theory says that Gauss-Seidel requires $O(N^2)$ iterations, whilst SOR requires $O(N)$ only with the optimal parameter, and one can test this by seeing that $\frac{GS}{SOR \times N}$ is a fixed constant, which is ≈ 0.26 from the table below.

N	Gauss-Seidel	SOR	$\frac{GS}{SOR \times N}$
21	743	126	0.2808
61	6433	402	0.2623
101	17512	677	0.2561
201	68101	1365	0.2482

2.4 Asserting grid-independence

To determine grid independence, we calculate a solution for grid sizes $N_k = 2^{k+1} + 1$ *i.e.* halving the grid spacing each time. All these grids share the same initial 25 grid points in N_1. Thus, we can compare the 2-norm of the difference between the computed solution to u_{N_k} and $u_{N_{k+1}}$ on the shared grid points. To assert grid-independence, we want this norm to be small. The analysis becomes easier if we plot in terms of the corresponding grid spacing $h_k = \frac{2}{N_k - 1}$.

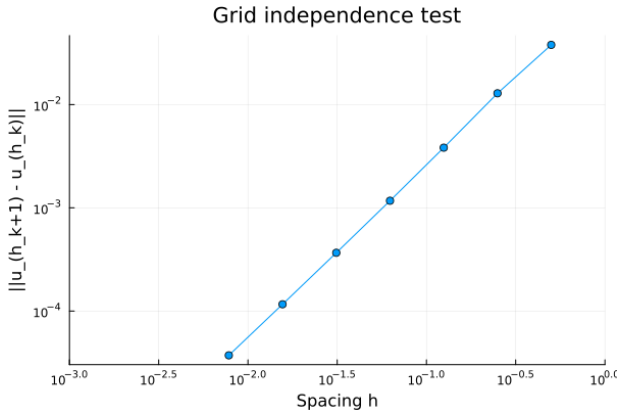


Figure 9: Log-Log plot of $\|u_{h_{k+1}} - u_{h_k}\|$ plotted against the grid spacing h_k.

From the plotted line we can see that the measured norm difference can be modelled as Kr^k where $K \approx 0.121$ and $r \approx 0.315$. Under the assumption that u_{h_k} converge to the true solution as $k \to \infty$ and that for any k, $\|u_{h_{k+1}} - u_{h_k}\| = Kr^k$, we can apply the triangle inequality and geometric series to see that

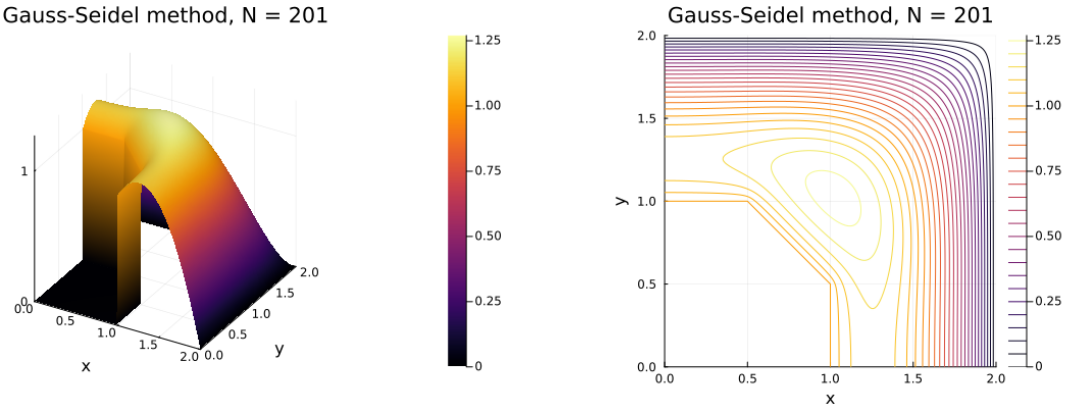
$$\|u_{h_k} - u\| \leq \sum_{j=k}^{\infty} \|u_{h_{j+1}} - u_{h_k}\| = \sum_{j=k}^{\infty} Kr^j = \frac{Kr^k}{1-r} \tag{2.3}$$

Thus, in order for our solution to satisfy $\|u_h - u\| < t$ at the 25 sampled grid points, we need h to be smaller than

$$h < 10^{\frac{\log(0.5)\log\left(\frac{t(1-r)}{K}\right)}{\log(r)}} \tag{2.4}$$

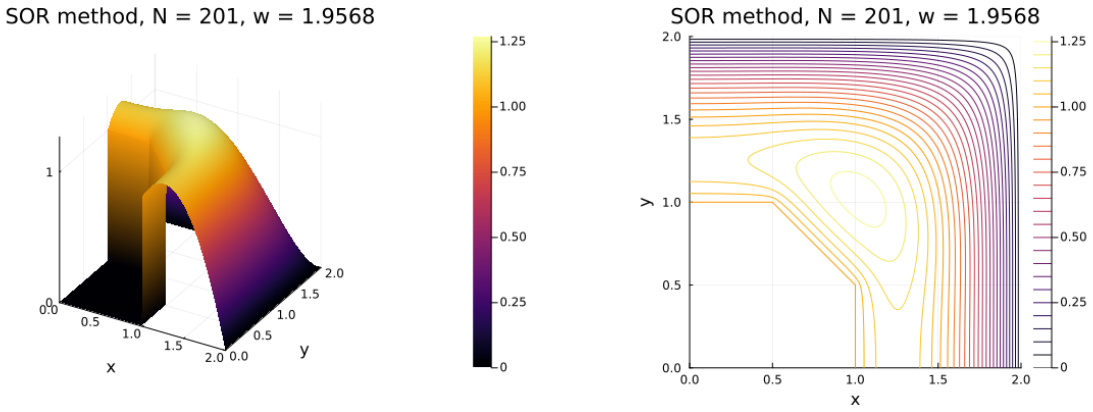
Thus, suppose we wished to have a solution accurate within 1% of the true solution (at the 25 grid points). We calculate that $h < 0.01887$, so we need at least 106 grid points. We will choose 201 grid points to achieve this and have a reasonably grid-independent solution.

2.5 Contour plots of u



(a) (b)

Figure 10: Contoured solution computed with Gauss-Seidel, using $N = 201, h = 0.005$.



(a) (b)

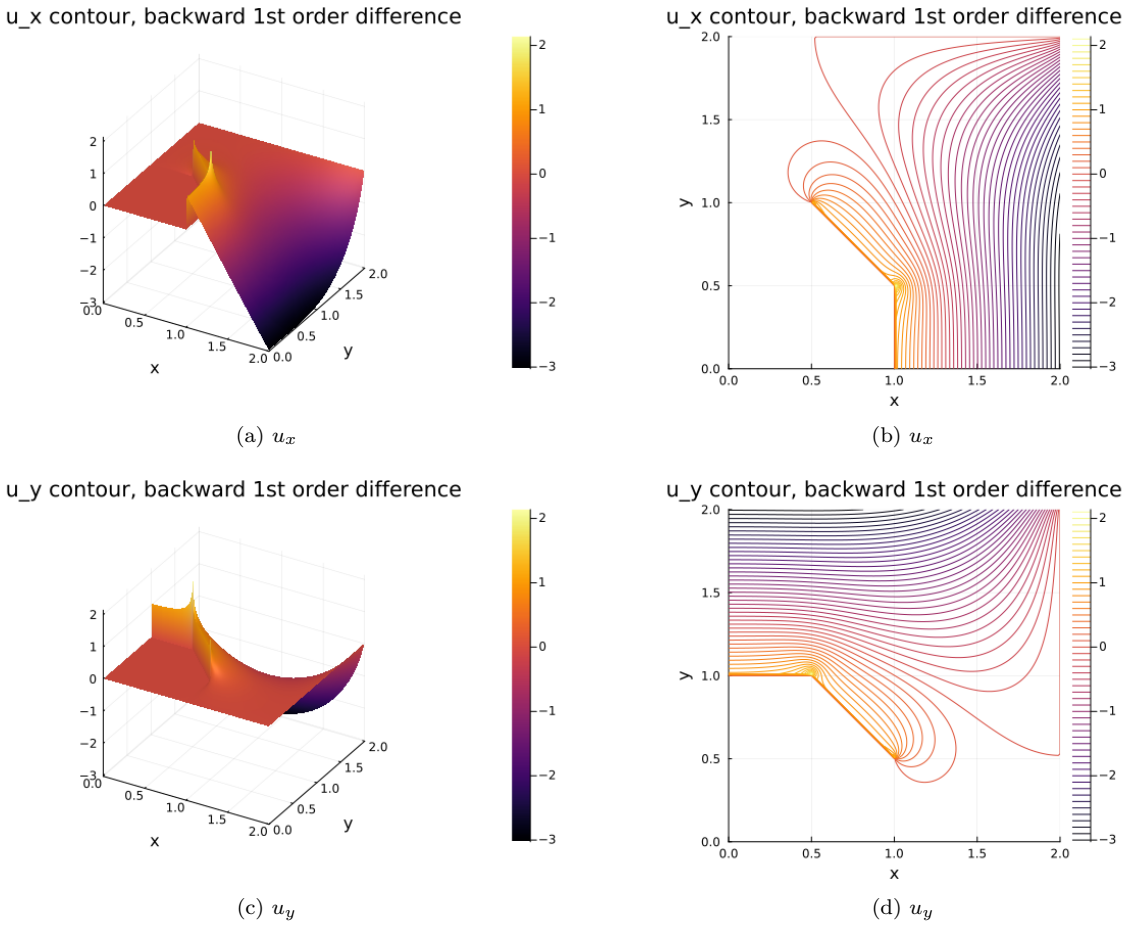
Figure 11: Contoured solution computed with SOR, $w = 1.9568$ using $N = 201, h = 0.005$.

2.6 Material Stresses

We can plot the individual components of u_x and u_y by using a backward 1st order difference (where i and j increase in the direction of positive x and positive y respectively).

$$u_x \approx \frac{U_{i,j} - U_{i-1,j}}{h} \qquad u_y \approx \frac{U_{i,j} - U_{i,j-1}}{h} \tag{2.5}$$

The individual components are plotted below in Figure 12. We notice the small spikes near the two 135° corners, and the large components near the edge.



(a) u_x (b) u_x

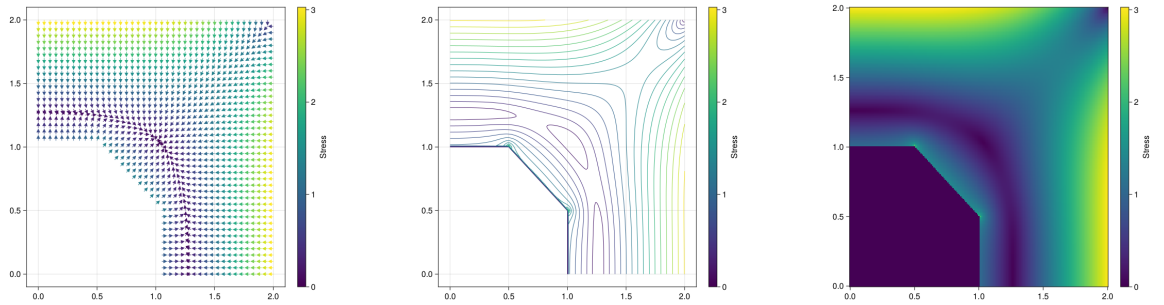
(c) u_y (d) u_y

Figure 12: Horizontal and vertical components of stress vectors.

For this grid, the maximum and minimum values are summarised in the table below. Since the problem is symmetric, we expect that the maximum and minimum values for u_x and u_y are the same.

$\max(u_x)$	$\min(u_x)$	$\max(u_y)$	$\min(u_x)$
2.1386238454222495	-3.0268664561932646	2.1386238454222495	-3.0268664561932646

We can also plot as a heatmap the magnitude and direction of the stress vectors, as seen in Figure 13. From this we see that the areas of greatest stress are the two long edges, particularly near the corners $(0, 2)$ and $(2, 0)$. In addition, the two corners located at $(0.5, 1)$ and $(1, 0.5)$ experience stress as well in a very concentrated region.



(a) Vector field (b) Contour map of vector magnitudes (c) Heatmap of vector magnitudes

Figure 13: Material stress calculated using backward 1st order finite differences.

2.7 Material stress in different grid sizes

In order to investigate grid-independence for material stress, we repeat these calculations for different N and calculate the corresponding maximum and minimum u_x and u_y. In the table below, we see that the minimum values of σ_x and σ_y are invariant under different grid sizes. However, the maximum is not: it is increasing as the grid becomes finer, without converging.

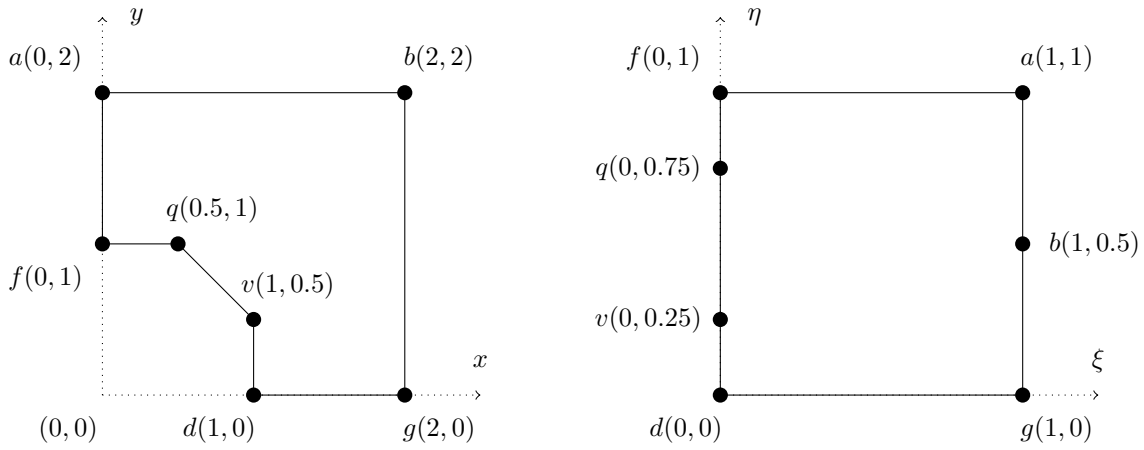
Intuitively, this makes sense since the negative values of σ_x and σ_y are achieved near the top-left and bottom-right corners and are generally spread out over a large area. In Figure 12, these two regions appear to behave nicely *i.e.* they do not change rapidly.

However, the positive values are concentrated in a small region, and when contoured in Figure 12 and Figure 13, the values of σ_x and σ_y have a sudden spike in the corners and appear to blow up. This requires a finer grid in the neighbourhood the points to resolve the material stresses accurately, which could explain why these values are heavily dependent on the grid.

N	$\max(u_x)$	$\min(u_x)$	$\max(u_y)$	$\min(u_x)$
101	1.8334958585643601	-3.0067386661442326	1.8334958585643601	-3.0067386661442326
201	2.1386238454222495	-3.0268664561932646	2.1386238454222495	-3.0268664561932646
301	2.33045628089954	-3.033566569231908	2.33045628089954	-3.033566569231908
401	2.474243586283187	-3.0369122087117493	2.474243586283187	-3.0369122087117493
501	2.590726251260278	-3.0389180506760622	2.590726251260278	-3.0389180506760622
801	2.851829158146124	-3.041925611353496	2.851829158146124	-3.041925611353496

3 Part C

3.1 Grid transform diagram



Physical plane Computational plane

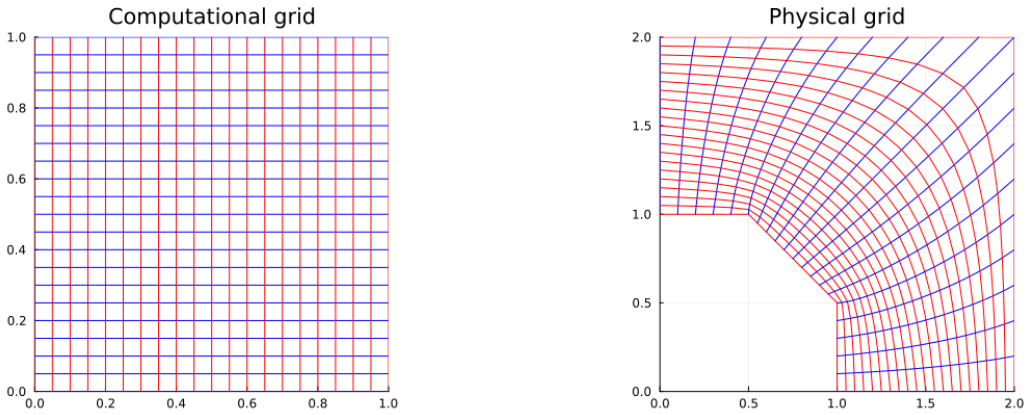
The above diagram shows the map from the physical plane to the computational plane. We choose ξ and η such that

- ξ and η are harmonic *i.e.*

$$\xi_{xx} + \xi_{yy} = 0 \qquad \eta_{xx} + \eta_{yy} = 0 \tag{3.1}$$

- $\xi = 0$ on $f - q - v - d$ and $\xi = 1$ on $g - b - a$. Similarly $\eta = 0$ on $g - d$ and $\eta = 1$ on $a - f$. All other boundary values in the physical plane are interpolated linearly *e.g.* ξ goes from 0 to 1 along the edge $a - f$ linearly.

In Figure 14 we plot the computational grid and show how it maps to the real physical grid we have.



(a) (b)

Figure 14: Computational grid to Physical grid mapping

The above plot was generated by solving the system of quasilinear elliptic PDEs

$$\begin{aligned} &\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = 0 \\ &\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = 0 \\ &\text{where} \\ &\alpha = x_\eta^2 + y_\eta^2 \\ &\beta = x_\xi x_\eta + y_\xi y_\eta \\ &\gamma = x_\xi^2 + y_\xi^2 \end{aligned} \tag{3.2}$$

The boundary conditions are just Dirichlet boundary conditions, which correspond to the x or y values we should expect along the edges of the computational plane. The above equations were solved using $O(h^2)$ order finite differences for all terms, with i pointing in the positive ξ direction and j in the positive η direction.

$$\frac{\partial x_{i,j}}{\partial \xi} \approx \frac{x_{i+1,j} - x_{i-1,j}}{2h} \tag{3.3}$$

$$\frac{\partial^2 x_{i,j}}{\partial \xi^2} \approx \frac{x_{i+1,j} - 2x_{i,j} + x_{i-1,j}}{h^2} \tag{3.4}$$

$$\frac{\partial^2 x_{i,j}}{\partial \xi \partial \eta} \approx \frac{x_{i+1,j+1} + x_{i-1,j-1} - x_{i-1,j+1} - x_{i+1,j-1}}{4h^2} \tag{3.5}$$

Using these finite differences, Gauss-Seidel iteration was applied (with the possibility for acceleration), solving for $x_{i,j}$ and $y_{i,j}$ each time (despite it not being a linear problem). A function for solving this is written at `gauss_seidel_xy_C`.

3.2 Computing the solution using the elliptic transform

The transformed PDE in terms of ξ, η is calculated by using the chain rule *e.g.* $u_x = u_\xi \xi_x + u_\eta \eta_x$ and the symmetry of second-order partial derivatives.

$$u_{\xi\xi}(\xi_x^2 + \xi_y^2) + 2u_{\xi\eta}(\eta_x \xi_x + \eta_y \xi_y) + u_{\eta\eta}(\eta_x^2 + \eta_y^2) + 4 = 0 \tag{3.6}$$

We can then use the inverse of the Jacobian matrix to rewrite the PDE using the derivatives of x and y, where the Jacobian determinant is given as $J = x_\xi y_\eta - x_\eta y_\xi$.

$$\begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix} = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix}^{-1} = \frac{1}{J} \begin{pmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{pmatrix} \tag{3.7}$$

resulting in the PDE

$$u_{\xi\xi}(y_\eta^2 + x_\eta^2) - 2u_{\xi\eta}(y_\xi y_\eta + x_\xi x_\eta) + u_{\eta\eta}(y_\xi^2 + x_\xi^2) + 4J^2 = 0 \tag{3.8}$$

The boundary conditions are given by

- $u(0, \eta) = 1$
- $u(1, \eta) = 0$
- $u_\xi y_\eta - u_\eta y_\xi = 0$ when $\eta = 1$
- $u_\eta x_\xi - u_\xi x_\eta = 0$ when $\eta = 0$

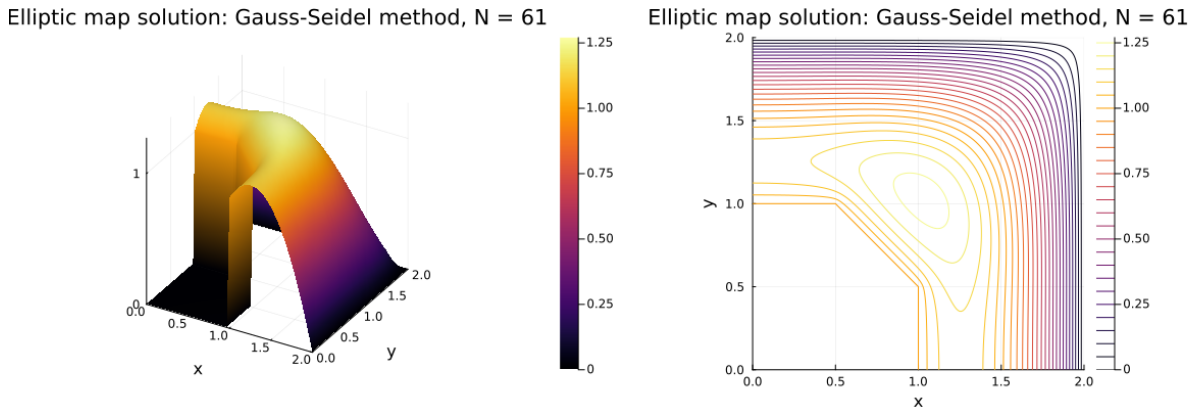
The last two conditions follow from the chain rule when applied to $u_x = 0$ and $u_y = 0$ respectively. We discretise all the derivatives as $O(h^2)$ accurate (the same used in finding x and y). At the top and bottom

boundary, we have to use $O(h^2)$ accurate forward and backward differences for the derivatives x_η and y_η, namely

$$f'(x) = \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h} + O(h^2) \tag{3.9}$$
$$f'(x) = \frac{3f(x) - 4f(x-h) + f(x-2h)}{2h} + O(h^2)$$

Once we have approximated all the derivatives of x and y, we can use Gauss-Seidel iteration (Function written at `gauss_seidel_method_C`) applied to the linear PDE An alternative approach would be to calculate the values of x and y at "ghost" points just above and below the lines $\eta = 1$ and $\eta = 0$ and use central differences. These values have to be calculated using iteration, since the system to be solved is also non-linear.

We first calculate a solution using $N = 61$ and show it in Figure 15.

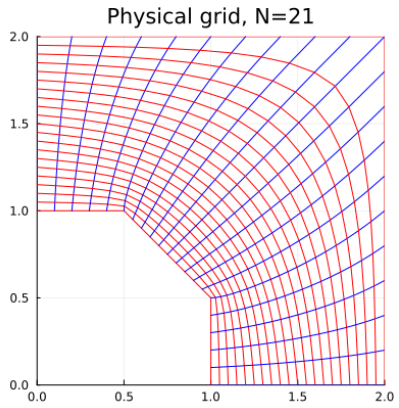


(a) (b)

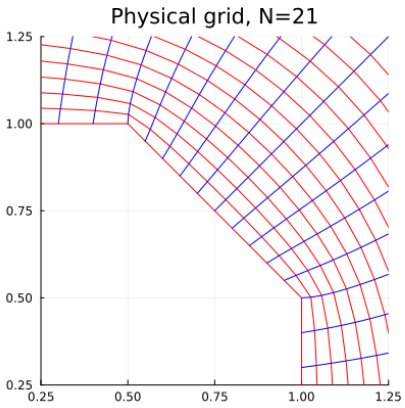
Figure 15: Solution u calculated using Elliptic maps to a square grid.

3.3 Clustering of grid points

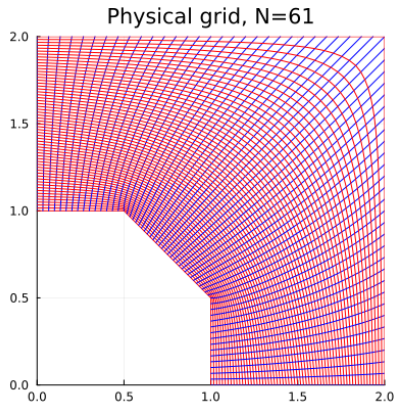
We demonstrate how the physical grid tends to cluster points around the two 135° corners in Figure 16.



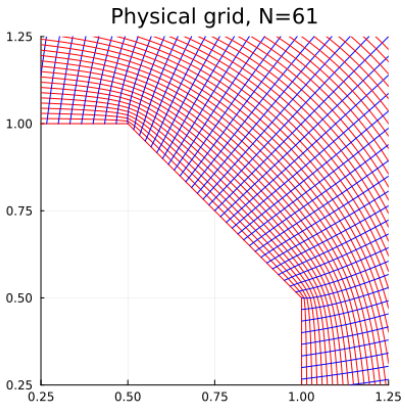
(a) $N = 21$



(b) Close up



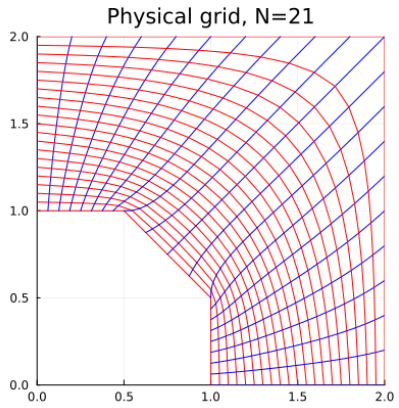
(c) $N = 61$



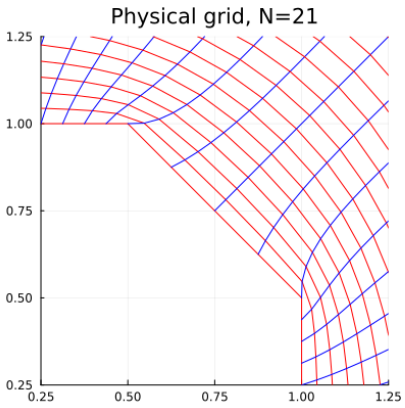
(d) Close up

Figure 16: Physical grid for $N = 21$ and $N = 61$, focused on the two inner corners.

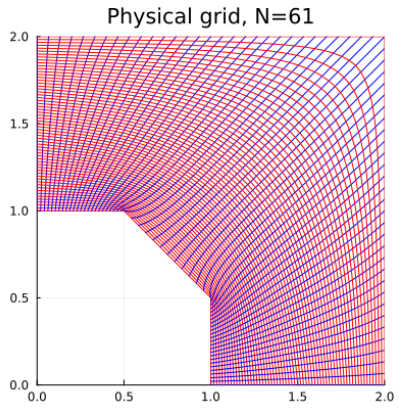
Even if we change the boundary conditions in solving PDE (3.2) (to lower the number of points along the $q - v$ edge from $N/2$ to only $N/5$, for example), the points still cluster near q and v, with some warping of the squares near the edge.



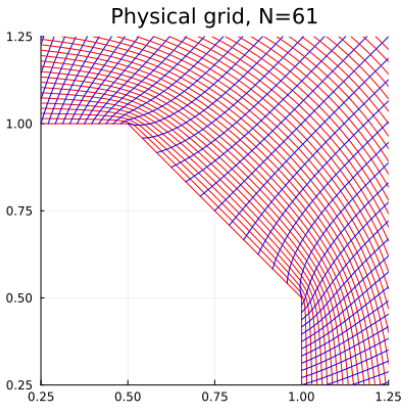
(a) $N = 21$



(b) Close up



(c) $N = 61$



(d) Close up

Figure 17: Physical grid for $N = 21$ and $N = 61$, focused on the two inner corners, with less points on the $q - v$ edge.

3.4 Asserting grid-independence: Elliptic mapping

Below (Figure 18) are plots of the pseudo residual on the computational grid as the number of points N is increased. A coarse 5×5 grid was refined by halving the spacing h each time, and the norm of $u_{new} - u_{old}$ was measured on the shared 25 grid points, though this time we cannot calculate with as high values of N due to the expensive computation of computing x and y from the quasilinear PDE (See Equation (3.2)). Although the accuracy is not as good as in Figure 9 for each individual spacing h, we see that the residual is decreasing by a fixed factor (roughly $r = 0.25$) for each halving of the grid spacing much like before. A similar calculation to before tells us that around $N = 220$ is needed to be within 1% of the true solution. Due to computational constraints, we will use $N = 101$ grid points from now on.

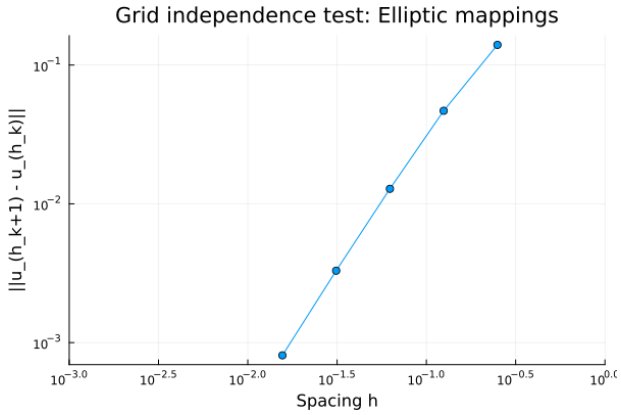


Figure 18: Log-Log plot of $\|u_{h_{k+1}} - u_{h_k}\|$ plotted against the grid spacing h_k, for the elliptic mapping solution.

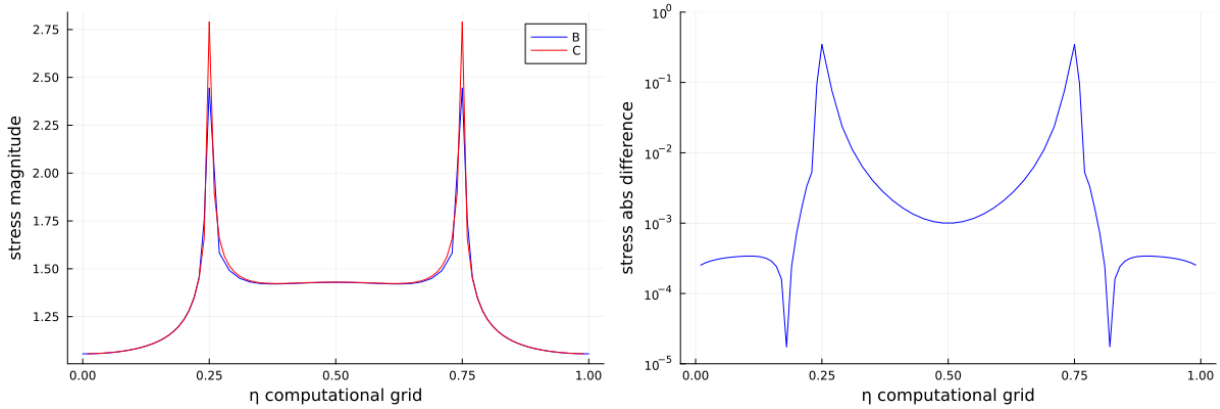
3.5 Comparison with Part B solution: Stress along $f - q - v - d$

Comparison of the solutions is hard to do within the interior of the domain, since the grid points are not shared and interpolation errors would dominate. However, we can measure the stress along the edge since these points are shared to compare.

In order to compare u_x and u_y, we need to use the chain rule for the solution using elliptic mappings, which results in

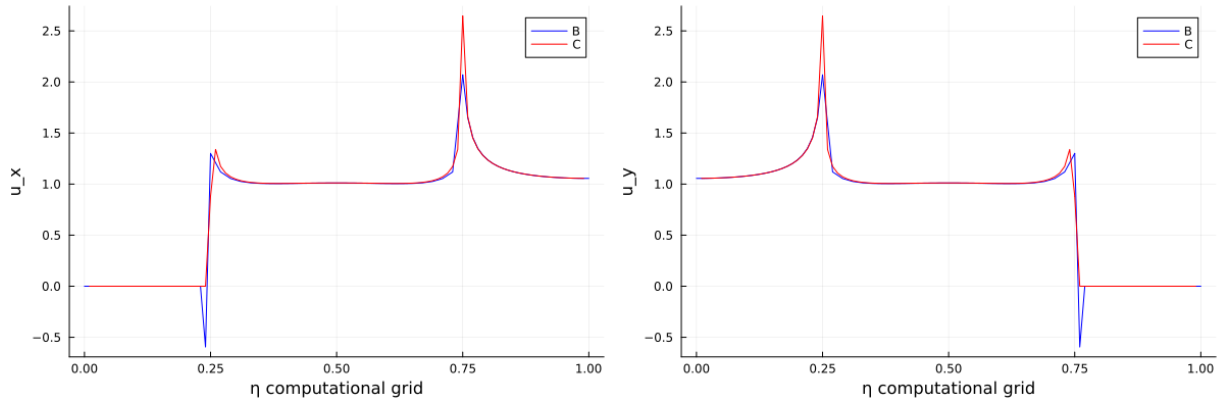
$$u_x = \frac{u_\xi y_\eta - u_\eta y_\xi}{J} \qquad u_y = \frac{u_\eta x_\xi - u_\xi x_\eta}{J} \tag{3.10}$$

We use $O(h^2)$ accurate forward differences (See Equation (3.9)) to approximate all the derivatives. Figure 19 shows both the individual stress components σ_x and σ_y along with the magnitude of the stress vector. We see that the solutions agree away from the two points q and v (Within $O(h^2)$ error), but closer to the two corners the solutions vary more. This is what we expect, as previously we determined that the maximum values of σ_x and σ_y are achieved at these two points and are not grid independent. As such, some variation is to be expected in the stress values near q and v.



(a) $\sigma = \sqrt{\sigma_x^2 + \sigma_y^2}$

(b) Absolute difference of σ from part B and C



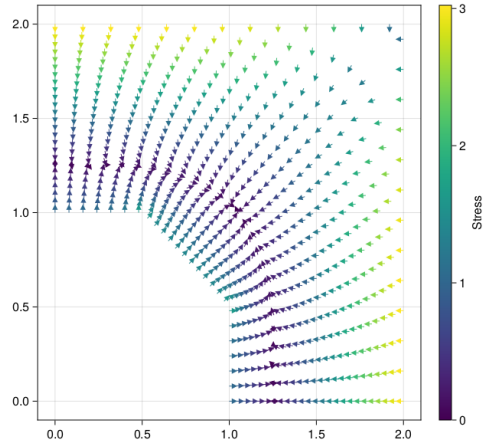
(c) σ_x

(d) σ_y

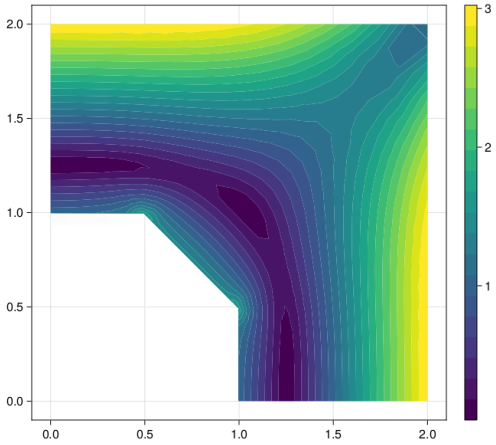
Figure 19: Comparison of stress along $f - q - v - d$ boundary.

3.6 Contour plots of stress field: Elliptic mapping

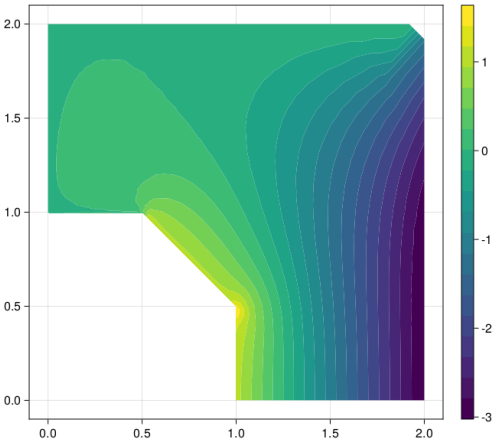
Due to the irregular placement of data, some plots can't be created (because Julia doesn't have a `tricontour` function...). Nevertheless, we plot the vector field of the stress and the contour plots of the magnitude of the stress as well as the individual x and y components. Much like before, we see that the greatest stress occurs near the top-left and bottom-right corners in a widespread area. Similarly we see spikes in the stress in the two corners at q and v.



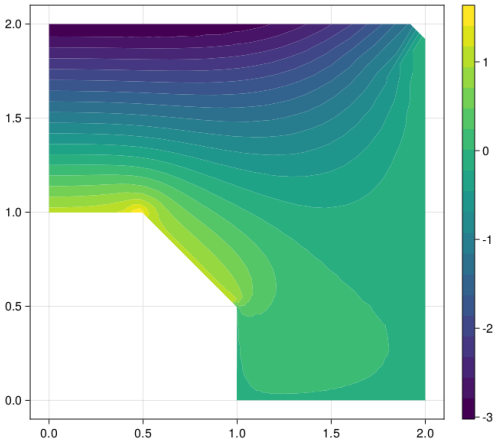
(a) Vector field



(b) $\sigma = \sqrt{\sigma_x^2 + \sigma_y^2}$



(c) σ_x



(d) σ_y

Figure 20: Plots of stress field σ