CO25: Laplace's Equation

Calum Holker
Somerville College

1 Abstract

An over-relaxation method is used to numerically solve Laplace's equation in a simple case. This is compared to the exact analytical solution and the values are found to all converge to exactly the analytical solution. The results further agree with the theory behind the over-relaxation parameter, agreeing on the optimum and maximum values. This suggests that this method of approximation is accurate and can be used for more complex problems that cannot be solve analytically.

2 Introduction

Laplace's equation appears frequently in a wide variety of areas of physics, such as electromagnetism, gravitational potentials, steady-state temperatures and of hydrodynamics. Laplaces equation states that the sum of the second order partial derivatives of an unknown function is equal to zero:

$$\nabla\psi^2 = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} = 0 \tag{1}$$

As it appears so often it is an incredibly useful equation to be able to solve for ψ, and has huge significance in many aspects of research. In this case we will use boundary conditions that lead to the equation being particularly simple to solve for ψ analytically. This will mean that we can test our numerical method against the analytical solution. The numerical method can also however be applied to much more complicated problems arising from more complicated boundary conditions, hence it's significance. Especially in practical physics research, often boundary conditions are too complex to solve analytically and so a numerical method must be deployed. We will solve within a rectangular domain with the following boundary conditions for our trial calculation [1]:

$$\begin{aligned}
\psi &= 0 && \textit{on } x = 0 \textit{ and } 0 \le y \le 1 \\
\psi &= 0 && \textit{on } y = 0 \textit{ and } 0 \le x \le 1 \\
\psi &= \sin 1 \sinh y && \textit{on } x = 1 \textit{ and } 0 \le y \le 1 \\
\psi &= \sin x \sinh 1 && \textit{on } y = 1 \textit{ and } 0 \le x \le 1
\end{aligned} \tag{2}$$

We can analytically solve this by taking the general solution to Laplace's equation and plugging in these boundary conditions [3]. This leads to a simple solution:

$$\psi = \sin x \sinh y \tag{3}$$

We use a successive over-relaxation method to solve this numerically [2], detailed in the following section. We can compare the numerical solution to the analytical solution above.

3 Method

Laplace's equation 1 can be approximated by:

$$\frac{(\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j})}{(\delta x)^2} + \frac{(\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1})}{(\delta y)^2} = 0 \tag{4}$$

Here the rectangular domain we are solving in is divided into grid points, with δx and δy being the distances between the grid points in the x and y directions, respectively. This approximation applies the equation to every point within the defined domain, leading to a large system of linear equations for $\psi_{i,j}$. This can be solved directly, however it is highly beneficial for time complexity to use an iterative process [2]. This arises especially when the system becomes more complicated that this example solving the huge system of linear equations would require an extreme amount of computing power and time. Equation 4 can be rewritten as below:

$$\psi_{i,j} = \frac{\frac{\psi_{i+1,j}+\psi_{i-1,j}}{\delta x^2} + \frac{\psi_{i,j+1}+\psi_{i,j-1}}{\delta y^2}}{2(\frac{1}{\delta x^2} + \frac{1}{\delta y^2})} \tag{5}$$

We then use a successive over-relaxation method [1], as this is much faster than using a Gauss-Seidel method, although both would work [2]. For this we use the following equation:

$$\psi_{i,j}^{m+1} = \alpha\overline{\psi}_{i,j} + (1-\alpha)\psi_{i,j}^{m} \tag{6}$$

Here $\overline{\psi}_{i,j}$ is calculated from 5 and m is the iteration number. In the case that we are looking at where our domain is defined as a uniform grid (i.e $\delta x = \delta y = \delta$), this can be rewritten as:

$$\psi_{i,j}^{m+1} = \psi_{i,j}^{m} + \frac{\alpha R_{i,j}}{4} \tag{7}$$

$$R_{i,j}^{m} = \psi_{i,j+1} + \psi_{i,j-1} + \psi_{i-1,j} + \psi_{i+1,j} - 4\psi_{i,j} \tag{8}$$

The method is to then sweep across the grid systematically row by row, calculating the values of $\psi_{i,j}$ and the residual $R_{i,j}^{m}$. The goal is to minimise the residual $R_{i,j}$ to zero, as at this point convergence has been reached. This method of over-relaxation systematically reduces the residuals. The parameter α is defined as the over relaxation parameter as it has a value greater than 1. It should produce convergence if the value is less than 2 [2]. The optimum value for a square with equal grid spaces in the x and y directions of $p = q = 7$ (where p and q are the number of grid spacings in the x and y directions) is given by [2][1]:

$$\alpha_{opt} = \frac{2}{1+\sin\frac{\pi}{p}} = 1.3948\ (5.s.f) \tag{9}$$

4 Results

For the boundary conditions defined in 2 and $\alpha = 1.4$ we used the successive over-relaxation method to produce the 3D contour plot in (Fig 1). The convergence of three different points in this plot were then examined closer in (Fig 2), and appeared to all converge to the analytical solutions within a few iterations. The locations with a greater amplitude converged faster than the locations with smaller amplitude.

We then examined how the value of α affected the convergence rate. (Fig 3) shows that the optimum value is in the range $1.35 < \alpha < 1.45$, and the number of iterations to converge increase hugely outside of this range. Looking at how the value of α affects the type of convergence, (Fig 4a) indicates that for α values less than the optimum, convergence is monotonic. (Fig 4b) indicates that for α values greater than the optimum, convergence is oscillatory. In both cases it is further clear that the further the value of α from the optimum, the slower the convergence.

Then we looked at how the value of α affected the convergence close to the maximum value for convergence ($\alpha = 2$). (Fig 5b) shows that the closer we get to $\alpha = 2$, the longer it takes to converge, with the wavefunction remaining constant at the maximum $\alpha = 2$. Then (Fig 5a) shows that as the value of α is increased past the maximum, the wavefunction rapidly converges. The figures in (Fig 6) show how for values close to $\alpha = 2$ the wavefunction oscillates within an envelope, which converges if α is less than 2 and diverges if greater than 2.

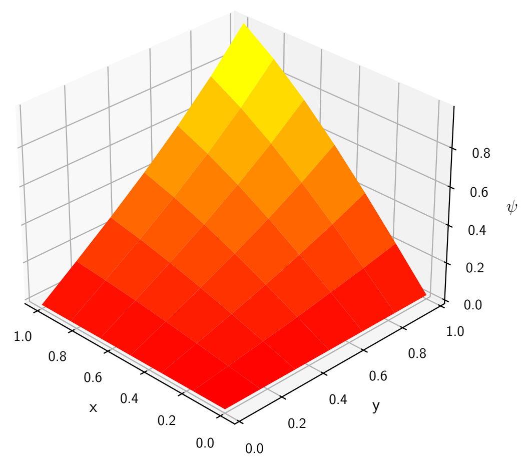


Figure 1: 3D Contour Plot of Function ψ

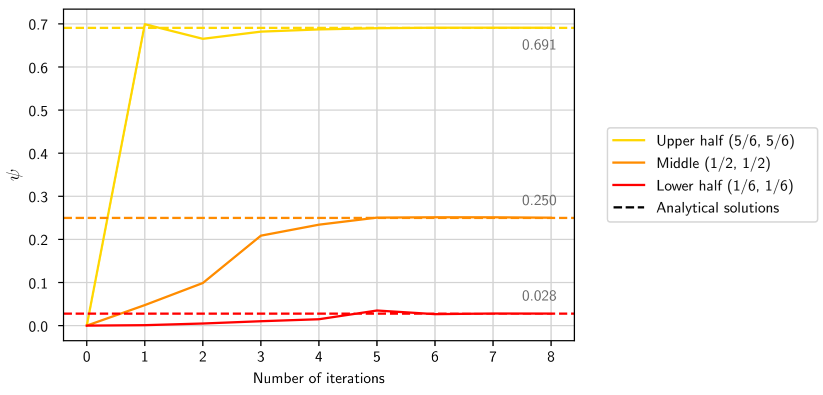


Figure 2: Convergence of numerical solution to analytical solution at three different points

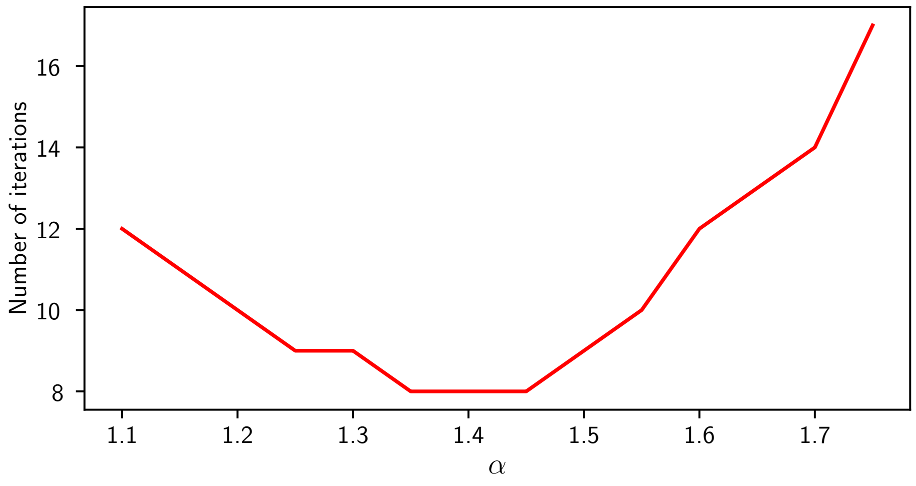
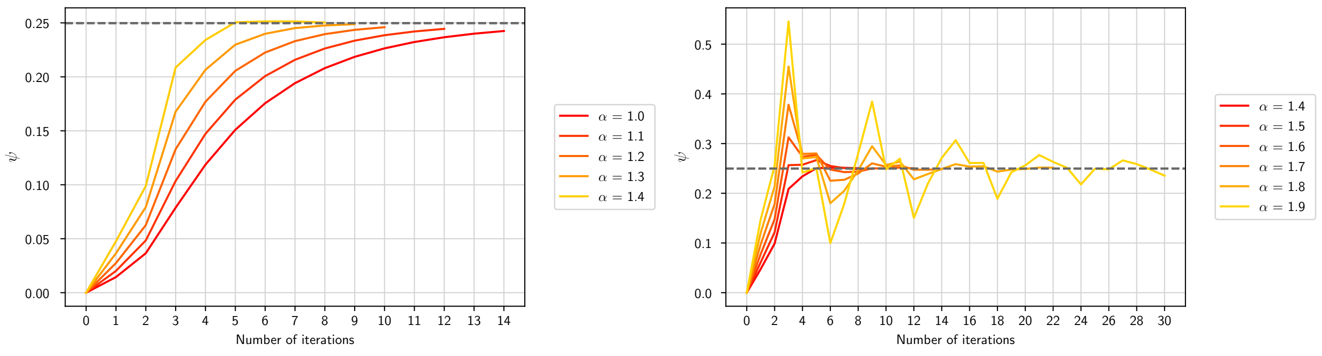


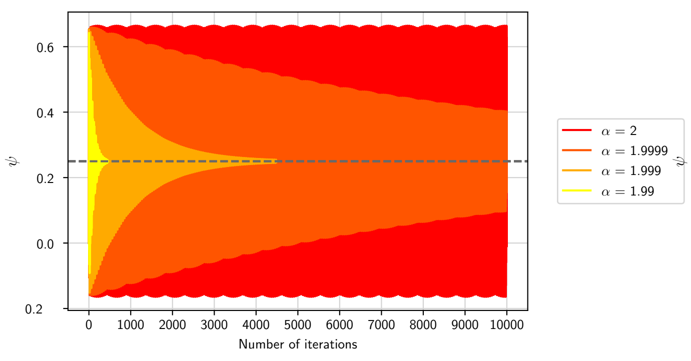
Figure 3: Number of iterations taken for values to converge as a function of α



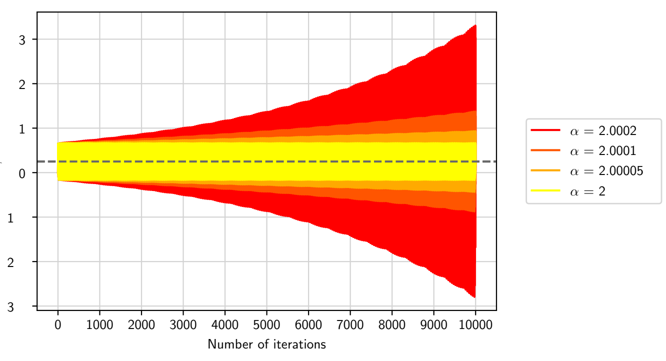
(a) α below the optimum value

(b) α above the optimum value

Figure 4: The convergence at (1/2, 1/2) for a range of α values

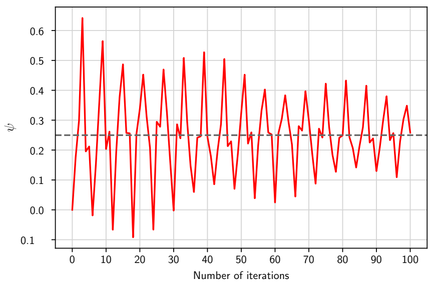


(a) α below the optimum value

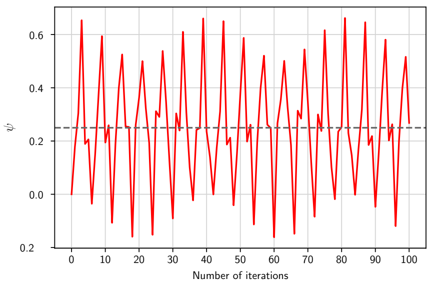


(b) α above the optimum value

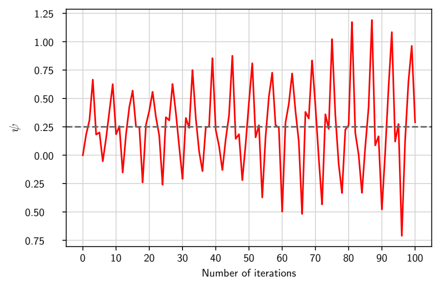
Figure 5: The convergence at (1/2, 1/2) for α values close to the optimum value



(a) $\alpha = 1.99$



(b) $\alpha = 2$



(c) $\alpha = 2.01$

Figure 6: The convergence at (1/2, 1/2) for α values close to the optimum value

5 Analysis

It is clear to see that the results imply that over-relaxation numerical method defined above works, converging each value to the exact analytical solution. The results also suggest that the theory behind the optimum over-relaxation parameter α is correct, with the calculated optimum value 9 lying in the centre of the optimum region of $1.35 < \alpha < 1.45$ shown by the results. The results also agree with the theory that $\alpha = 2$ is the maximum value for convergence. In our results this is the exact point of inversion from conversion to diversion. This suggests the theory was all correct. A point that is important to note is that if α is below the optimum value the convergence is monotonic, and if it is greater then the convergence is oscillatory.

6 Conclusion

A over-relaxation method is used to numerically solve Laplace's equation. This is compared to the analytical solution and the values all converge to exactly the analytical solution. The results further agree with the theory behind the over-relaxation parameter, giving the same optimum and maximum values. This suggests that this method of approximation is accurate and can be used for more complex problems that cannot be solve analytically. Note also that if the over-relaxation parameter is below the optimum, convergence will be monotonic, and if it is above convergence will be oscillatory.

References

[1] EG, MK, *CO25 Laplace's Equation*, Oxford Physics, 2020[1].

[2] Numan, *Iterative Methods*, University of St Andrews [2].

[3] *Analytic Solutions to Laplace's Equation in 2D*, [3].

[1]`https://www-teaching.physics.ox.ac.uk/practical_course/scripts/srv/local/rscripts/trunk/Computing/CO25/CO25.pdf`

[2]`http://www-solar.mcs.st-andrews.ac.uk/~clare/Lectures/num-analysis/Numan_chap2.pdf`

[3]`https://newton.ex.ac.uk/teaching/CDHW/EM/CW970317-2.pdf`

Appendix Source Code for Over-Relaxation Solution

The source code for implementing the over-relaxation algorithm for Laplace's equation:

```python
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from matplotlib.lines import Line2D
%matplotlib inline

def solve_laplace (init_psi, alpha, N_iter):
    """
    Solves Laplace's equation using the over-relaxation method
    Input:        init_psi - 2D matrix containing the initial \psi, including boundaries
                  alpha - the coefficient of relaxation
                  N_iter - maximum number of iterations performed
    Output:       psi - 2D matrix of the value of \psi after (up to) N_iter iterations
                  hist_values - (N_iter x 3) matrix that contains historical values of 3 points during the iteration (1 in upper half, 1 in middle, 1 in lower half)
    Constraints:  The boundaries of \psi are kept constant during the iterations
    """
    psi = init_psi
    hist_values = [[init_psi[1,5]],[init_psi[3,3]],[init_psi[5,1]]]
    while N_iter > 0:
        sum_squares = 0
        for i in range(1,6):
            for j in range(1,6):
                Rm = psi[i,j+1] + psi[i+1,j] + psi[i-1,j] + psi[i,j-1] - (4*psi[i,j])
                sum_squares = sum_squares + (Rm ** 2)
                psi[i,j] = psi[i,j] + ((alpha * Rm)/4)
                if (i == 1) and (j == 5):
                    hist_values[0].append(psi[1,5])
                if (i == 3) and (j == 3):
                    hist_values[1].append(psi[3,3])
                if (i == 5) and (j == 1):
                    hist_values[2].append(psi[5,1])
        N_iter = N_iter - 1
        if sum_squares < 0.001:
            break
    return psi, hist_values

def get_init_psi(n):
    """
    Generates the initial psi nxn matrix with the boundary conditions implemented
    """
    init_psi = np.zeros((n, n))
    d = 1/(n-1)
    for i in range(n):
        init_psi[0,i] = np.sin(i*d)*np.sinh(1)
        init_psi[i,n-1] = np.sin(1)*np.sinh((6-i)*d)
    return init_psi

def solve_laplace_analytically(x,y):
    """
    Returns the analytical solution for Laplace's equation at a coordinate (x,y)
    """
    return np.sin(x)*np.sinh(y)

# Solve's Laplace's equation numerically

alpha = 1.4 # Over-relaxation parameter
N_iter = 30 # Maximum number of iterations
init_psi = get_init_psi(7) # Generates initial psi matrix

psi, hist_values = solve_laplace(init_psi,alpha,N_iter) # Solves equation

# Plotting saved 3 positions as they converge to the analytical results

t = np.arange(0,len(hist_values[1]),1)

plt.plot(t, hist_values[0], label='Upper Half (5/6, 5/6)', color='gold')
value_1 = solve_laplace_analytically(5/6, 5/6)
plt.axhline(y=value_1, color='gold', linestyle='--')

plt.plot(t, hist_values[1], label='Middle (1/2, 1/2)', color='darkorange')
value_2 = solve_laplace_analytically(3/6, 3/6)
plt.axhline(y=value_2, color='darkorange', linestyle='--')

plt.plot(t, hist_values[2], label = 'Lower Half (1/6, 1/6)', color='red')
value_3 = solve_laplace_analytically(1/6, 1/6)
plt.axhline(y=value_3, color='red', linestyle='--')
```