Partial Differential Equations

—

Elliptic: Poisson's Equation

Keegan W$^{\text{m.}}$ Karbach

May 13, 2020

Introduction

In this project, we examine elliptic partial differential equations. These equations are classified as having a negative discriminant and thus, complex characteristic curves. Physically this manifests as solutions that have no time dependence and are well posed to boundary value problems. As they have no physical characteristic curves, they cannot have discontinuities in the derivatives of the solution as these must propagate along these curves – the solutions are smooth. Thus, elliptic PDEs govern equilibrium problems.

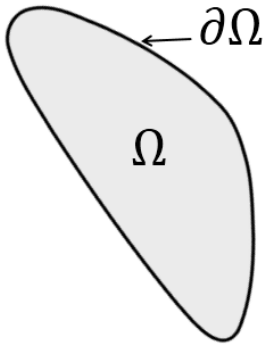

Figure 1: A general domain for a prototypical equilibrium problem

Equilibrium problems are steady-state problems in bounded domains $\Omega(x, y)$ in which the solution $\Phi(x, y)$ is governed by an elliptic PDE subject to boundary conditions specified each point on the boundary $\partial\Omega$. Equilibrium problems are solved numerically using *relaxation* methods in which the entire solution is passed on by a jury of neighboring points in Ω requiring satisfaction of all internal requirements (i.e., the solution) and all the boundary conditions simultaneously. Due to the fact that elliptic PDEs have no real characteristics, the solution at every point in the solution domain is influenced by the solution at all the other points, and the solution at each point influences the solution at all the other points. Thus, these problems are characterized by continuous, twice-differentiable solutions on Ω (Figure 2).

The prototypical elliptic PDE is Poisson's equation (1), of which the Laplace equation (2) is a special case; we will examine both in this project. These equations govern the study of potential fields and therefore form a basis for an understanding of a number of fields in physics such as ideal fluid flow, mass diffusion, heat diffusion, electrostatics, and gravity.

$$\nabla^2\Phi = -\frac{\rho}{\epsilon_0} \tag{1}$$

$$\nabla^2\Phi = 0 \tag{2}$$

In this project, we will explore three numerical methods for solving Laplace's equation: the Jacobi method, the Gauss-Seidel method, and Simultaneous Overrelaxation – each a refinement

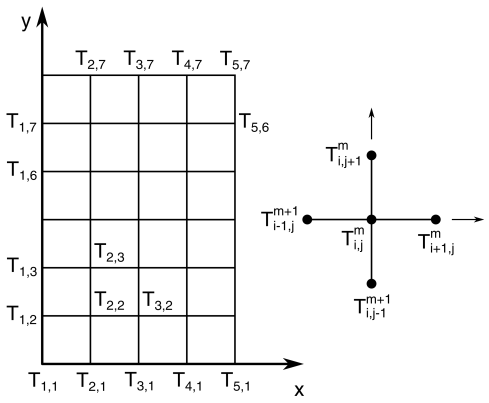

Figure 2: Jury method for solving equilibrium problems

of the previous. The Jacobi method is the canonical jury method; the values of each adjacent point are averaged to update the current point.

$$\Phi^{n+1}_{i,j} = \frac{1}{4}\left[\Phi^{n}_{i+1,j} + \Phi^{n}_{i-1,j} + \Phi^{n}_{i,j+1} + \Phi^{n}_{i,j-1}\right]$$

The Gauss-Seidel method is essentially the same as the Jacobi method, with the exception that it uses points that have already been calculated to update the current point.

$$\left[\Phi^{n}_{i-1,j}, \Phi^{n}_{i,j-1}\right] \longrightarrow \left[\Phi^{n+1}_{i-1,j}, \Phi^{n+1}_{i,j-1}\right]$$

This saves computation time and memory allocation as only one array for Φ needs to be saved – it is updated element wise during the iteration as opposed to the Jacobi method, which builds an updated copy of the current array.

A further refinement comes with the addition of an overrelaxation parameter, lending the method of Successive Overrelaxation (SOR). This accelerates the convergence of the Gauss-Seidel method by utilizing a weighted average of the Gauss-Seidel jury method with the current iteration.

1 Problem 8.1

Evaluate the potential $\Phi(x, y)$ numerically from equation (8.15) making surface and contour plots for a variety of terms. Estimate how many terms are needed to obtain about 1% accuracy.

For reference, Garcia [2] equation (8) is

$$\Phi(x, y) = \Phi_0 \sum_{n=1,3,5,\ldots}^{\infty} \frac{4}{\pi n} \sin\left(\frac{n\pi x}{L_x}\right) \frac{\sinh(n\pi y/L_x)}{\sinh(n\pi L_y/L_x)}$$

In order to implement this as a direct numerical method, we use the function

```
def phi_n(n,x,y,L):
    return (4/(np.pi*n*np.sinh(n*np.pi)))*np.sin(n*np.pi*x/L)*np.sinh(n*np.pi*y/L)
```

And iterate over the sum and both variables using a nested `for()` loop

```
for k in n_lst:
    for i in range(N) :
        for j in range(N) :
            phi[i,j] = phi0 * phi_n(k, x[i], y[j], L)
```

The Laplace equation was solved for an initial field of $\Phi_0 = 1$ on a 50×50 grid for terms through $n = 11, 21,$ and 51.

The results are as follows. Note that the contour plots are all identical, save for the top border

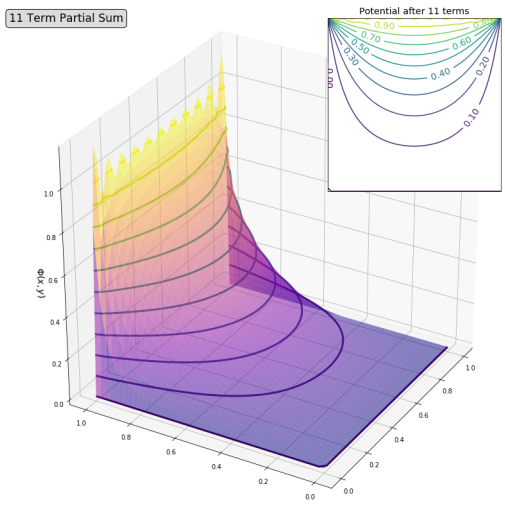

Figure 3: Series solution for $n = 11$ terms. Note the prominent Gibbs phenomenon on the nonzero border.

which does exhibit evidence of the Gibbs phenomenon in the first and second figure. We want to investigate how many terms it will take to obtain accuracy to within one percent. One method is to only examine the nonzero border, as this is where the plot changes the most over multiple terms. Another method is to examine the entire plot, determine the residual between subsequent iterations, and calculate an error from that residual. This is the method I chose to investigate due to the fact that it seemed to be more mathematically rigorous, at least to me. I decided to use Root-Mean-Square deviation (RMSD). Often used to determine the quality of a regression model to data, this can also be used to compare differences between two varying quantities, neither of which is accepted as a standard value. The rationale here is that, as the number of terms goes to infinity, the residual between subsequent iterations will go to zero – we want to determine when this RMSD value drops to below one percent. For two dimensions, the RMSD is defined as:

$$\text{RMSD} = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(\Phi_{i,j}^{n} - \Phi_{i,j}^{n-1})^2}{N^2}}$$

Where n is the temporal index and N is the spatial index. This is implemented in Python as

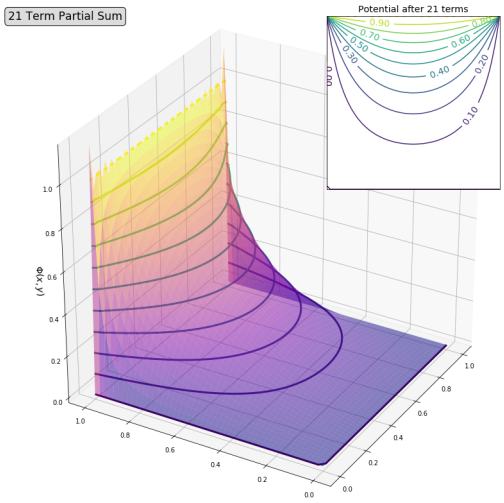

Figure 4: Series solution for $n = 21$ terms. With ten additional terms, the Gibbs phenomenon is much less pronounced.

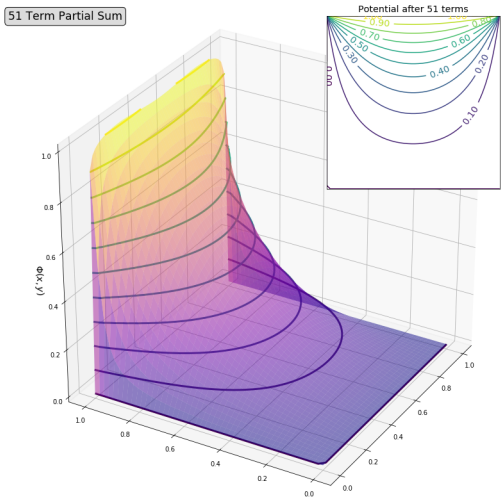

Figure 5: Series solution for $n = 51$ terms. With 51 terms, the Gibbs phenomenon is unnoticeable in the surface plot.

```
# IMPLEMENTATION OF RMSD ANALYSIS
flag = 0
for i in range(terms-1):
    resid[:,:,i] = delt_phi[:,:,i] - delt_phi[:,:,i+1]
    resid2[i] = np.sqrt(sum(sum(resid[:,:,i]**2))/(N**2))
    if resid2[i]/resid2[0]*100 < 1 and resid2[i]/resid2[0]*100 > .8 and flag == 0:
        resid_crit = i
        flag = 1
```

The results of this analysis are shown in Figure 6 – the critical number of terms for a one percent RMSD is $n = 83$ and the values for $n = 82, 83, 84$ are shown circled in the inset. Here, the number

of terms represents 83 *odd* terms, so we would take the sum to the index variable of $i, j = 165$. The figure shows a normalized RMSD value as a percentage, for comparative purposes. Note that there are two numerical artifacts where the residual change between terms was negligible – these were omitted from the analysis as they did not fit the functional form of the error curve.

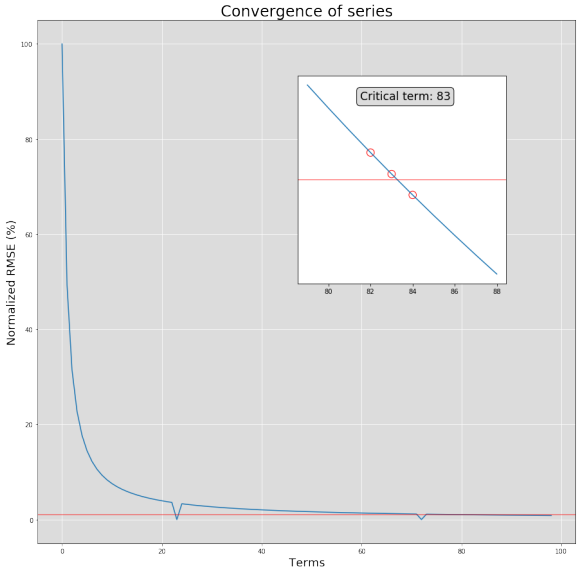

Figure 6: Root-Mean-Square deviation between subsequent terms as a function of number of partial sum terms

2 Problem 8.3

2.1 Part (a)

Find the analytical solution to the three-dimensional cubic boundary value problem using separation of variables.
Solving the Laplace equation in three dimensions with the boundaries

$$\Phi(x = 0, y, z) = \Phi(x = L, y, z) = 0$$
$$\Phi(x, y = 0, z) = \Phi(x, y = L, z) = 0$$
$$\Phi(x, y, z = 0) = \Phi(x, y, z = L) = \Phi_0$$

begins by separating the PDE into a system of ODEs:

$$\nabla\Phi(x, y, z) = 0$$
$$Y(y)Z(z)\frac{\partial^2 X(x)}{\partial x^2} + X(x)Z(z)\frac{\partial^2 Y(y)}{\partial y^2} + X(x)Y(y)\frac{\partial^2 Z(z)}{\partial z^2} = 0$$
$$\frac{1}{X(x)}\frac{d^2X(x)}{dx^2} + \frac{1}{Y(y)}\frac{d^2Y(y)}{dy^2} + \frac{1}{Z(z)}\frac{d^2Z(z)}{dz^2} = 0$$

Can be recast into the ODE system

$$\begin{cases} \frac{1}{X}X'' = -\left(\frac{1}{Y}Y'' + \frac{1}{Z}Z''\right) = -\alpha^2 \\ \frac{1}{Y}Y'' = \alpha^2 - \frac{1}{Z}Z'' = -\beta^2 \end{cases}$$

Which is separated into the system

$$\begin{cases} X'' = -\alpha^2 X \\ Y'' = -\beta^2 Y \\ Z'' = (\alpha^2 + \beta^2)Z \end{cases}$$

If we define $\gamma = \sqrt{\alpha^2 + \beta^2}$, we get the general solution:

$$\Phi(x, y, z) \propto e^{\pm i\alpha x}\ e^{\pm i\beta y}\ e^{\pm \gamma z}$$

We need the oscillatory exponentials in the x and y variables in order to force their zero boundary conditions, leaving the hyperbolic exponent for the nonzero boundaries in z. Applying the boundary conditions gives us:

$$\begin{cases} X \sim \sin\left(\frac{n\pi}{L}x\right) \\ Y \sim \sin\left(\frac{m\pi}{L}y\right) \\ Z \sim \cosh\left(\frac{\gamma\pi}{L}z\right) \end{cases}$$

Giving us the general series solution:

$$\Phi = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} A_{m,n}\left[\sin\left(\frac{n\pi}{L}x\right)\sin\left(\frac{m\pi}{L}y\right)\cosh\left(\frac{\sqrt{m^2+n^2}\pi}{L}z\right)\right]$$

We find the coefficient $A_{m,n}$ by Fourier integration, set $z = 1$ and take the integral of the x variable first.

$$\int_0^L \Phi_0 \sin\left(\frac{k\pi}{L}x\right)dx = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} A_{m,n}\cosh(\gamma\pi)\sin\left(\frac{m\pi}{L}y\right)\int_0^L \sin\left(\frac{n\pi}{L}x\right)\ \sin\left(\frac{k\pi}{L}x\right)dx$$

$$-\frac{L\Phi_0}{k\pi}\cos\left(\frac{k\pi}{L}x\right)\Big|_0^L = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} A_{m,n}\cosh(\gamma\pi)\sin\left(\frac{m\pi}{L}y\right)\frac{L}{2}\delta_{kn}$$

The delta function is zero everywhere except $n = k$, collapsing the sum over n.

$$\frac{L\Phi_0}{k\pi}\left(1-(-1)^k\right) = \frac{L}{2}\sum_{m=1}^{\infty} A_{k,m}\cosh\left(\sqrt{k^2+m^2}\pi\right)\sin\left(\frac{m\pi}{L}y\right)$$

Repeating this procedure for the y variable and solving for $A_{m,n}$ gives us:

$$A_{m,n} = \frac{4\Phi_0}{mn\pi^2\cosh\left(\sqrt{m^2+n^2}\pi\right)}\left(1-(-1)^m\right)\left(1-(-1)^n\right)$$

Note that:

$$\left(1-(-1)^n\right)\begin{cases} 0 & n \text{ even} \\ 2 & n \text{ odd} \end{cases}$$

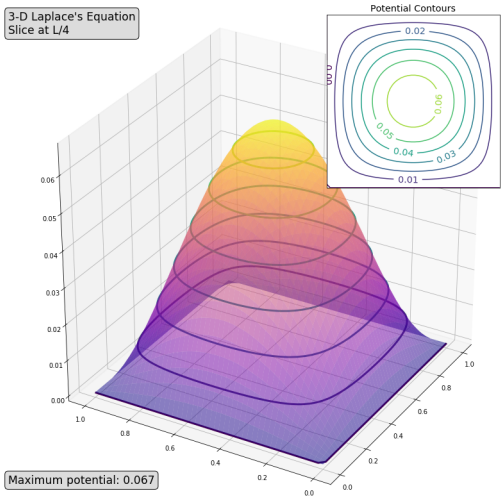

Figure 7: Series solution for $n = 11$ terms at $z = L/4$.

Which gives us the particular series solution:

$$\Phi(x,y,z) = \sum_{m\sim\text{odd}}^{\infty}\sum_{n\sim\text{odd}}^{\infty} \frac{16\Phi_0}{mn\pi^2\cosh\left(\sqrt{m^2+n^2}\pi\right)} \sin\left(\frac{n\pi}{L}x\right)\sin\left(\frac{m\pi}{L}y\right)\cosh\left(\frac{\sqrt{m^2+n^2}\pi}{L}z\right)$$

For intermediate steps, please consult Appendix B for the pencil and paper solution.

2.2 Part (b)

Numerically solve the analytical solution and plot values for $z = L/4$ and $z = L/2$
Implementation of the direct numerical solution is much the same as Problem 8.1. The results are as follows in Figures 7 and 8; the contour level spacing is equivalent over both plots to better show the differences in the magnitude of the scalar field between the two plots as their functional form is very similar. The solutions were taken to 11 terms.

2.3 Part (c)

Numerically solve the physical system using relaxation methods and plot values for $z = L/4$ and $z = L/2$
Implementation of the relaxation method solution is as follows in Figures 9 and 10; I chose to use the Gauss-Seidel method.

```
for iter in range(iterMax) :
    changeSum = 0
    ## G-S method ##
    for i in range(1,N-1) :
        for j in range(1,N-1) :
            for k in range(1,N-1):
                temp = ( phi[i+1,j,k] + phi[i-1,j,k] +
                         phi[i,j-1,k] + phi[i,j+1,k] +
```

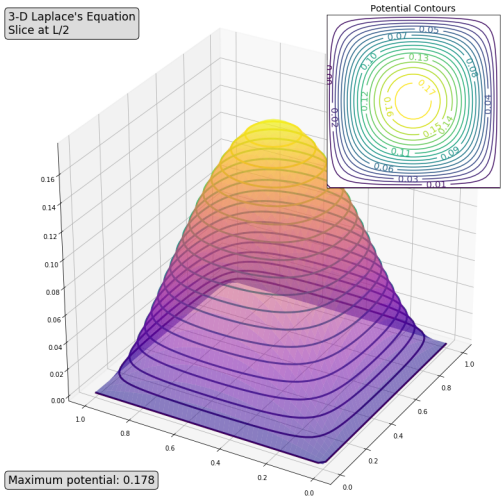

Figure 8: Series solution for $n = 11$ terms at $z = L/2$.

```
                    phi[i,j,k-1] + phi[i,j,k+1] ) / 6
            changeSum += abs( 1 - phi[i,j,k]/temp )
            phi[i,j,k] = temp
```

The results are as follows; the contour level spacing is equivalent over both plots and to the solution in part (b) to better show the differences in the magnitude of the scalar field between the plots as their functional forms are very similar. We can see that the relaxation method gives

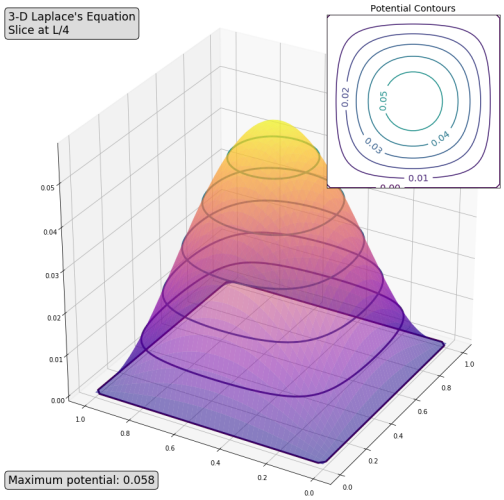

Figure 9: Relaxation solution for $n = 11$ terms at $z = L/4$.

solutions almost identical to the ones in part (b) with the percent difference 14.4% in the $L/4$ measurement and 0.563% for the $L/2$ measurement. As expected the scalar potential is greater at $z = L/2$ than it is at $z = L/4$.

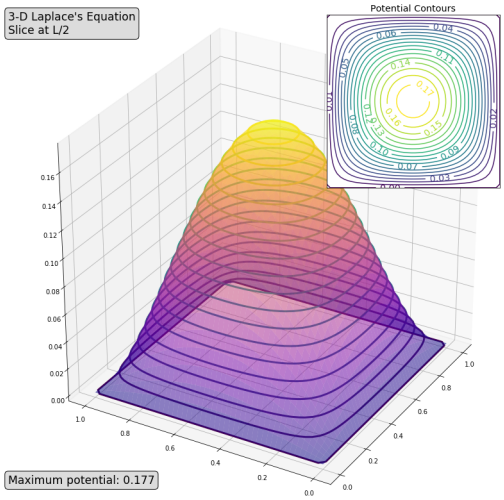

Figure 10: Relaxation solution for $n = 11$ terms at $z = L/2$.

3 Problem 8.4

3.1 Part (a)

Examine the impact of grid size on the performance of the Jacobi method using a reasonable initial guess (first partial sum).
In order to fit a function to extrapolate impact outside of the given range of $N = [10, 50]$, a series of 21 evenly-spaced values was taken. The Jacobi method was initialized with a reasonable initial guess. Figure 11 shows the number of iterations as a function of grid size. Three regression models were fit to the data:

$$\begin{aligned}\hat{Y}_P &= \beta_1 \cdot X^{\beta_2} \\ \hat{Y}_Q &= \beta_1 + \beta_2 X + \beta_3 X^2 \\ \hat{Y}_E &= \beta_1 + e^{\beta_2 X}\end{aligned}$$

The first model is the canonical 'power-law', the second model is a quadratic model, and the third is an exponential model. The equations of the models will be listed with each regression analysis.

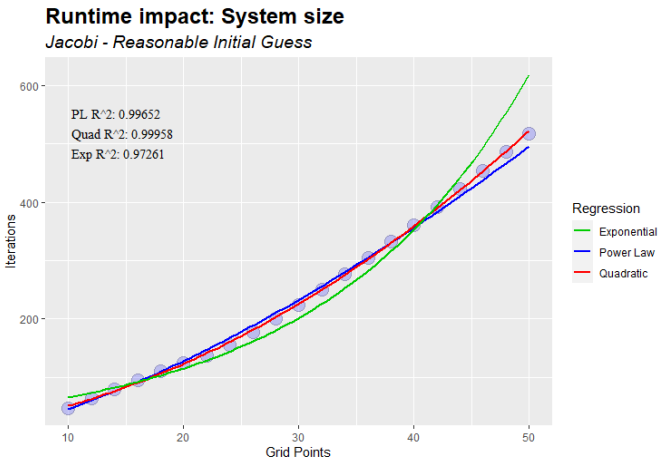

Figure 11: Analysis of runtime impact for the Jacobi method with a reasonable initial guess

The quadratic model appears to fit the model best and has the highest R^2 value. The equations are as follows:

$$\hat{Y}_P = 1.52 \cdot X^{1.48}$$
$$\hat{Y}_Q = 8.97 + 2.66X + 0.15X^2$$
$$\hat{Y}_E = 37.47 + e^{0.06X}$$

3.2 Part (b)

Examine the impact of grid size on the performance of the Jacobi method using a poor initial guess (zeros on the interior).
I noticed that the Jacobi method was numerically unstable for an initial guess of zero, so I initialized the interior points to a very small number. Figure 12 shows the results of this analysis.

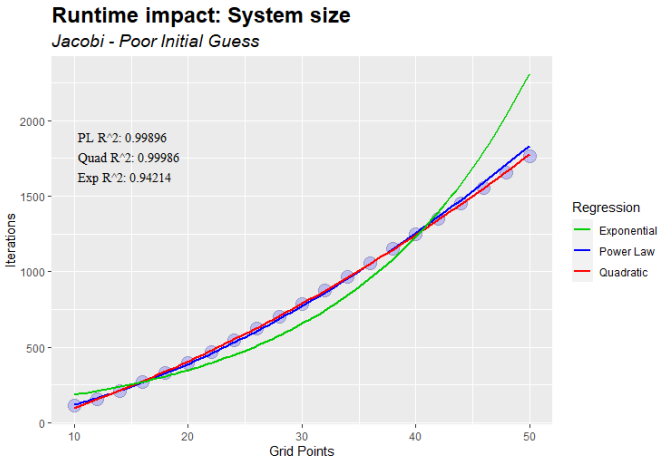

Figure 12: Analysis of runtime impact for the Jacobi method with a poor initial guess

The quadratic model again appears to fit the model best and has the highest R^2 value. The equations are as follows:

$$\hat{Y}_P = 2.45 \cdot X^{1.69}$$
$$\hat{Y}_Q = -133.99 + 19.61X + 0.37X^2$$
$$\hat{Y}_E = 98.79 + e^{0.06X}$$

Note that the number of iterations is more than three times greater with a poor initial guess than it is with a reasonable initial guess.

3.3 Part (c)

Examine the impact of grid size on the performance of the SOR method using a reasonable initial guess.
Here, the SOR method was implemented with the initial condition set as the first partial sum.

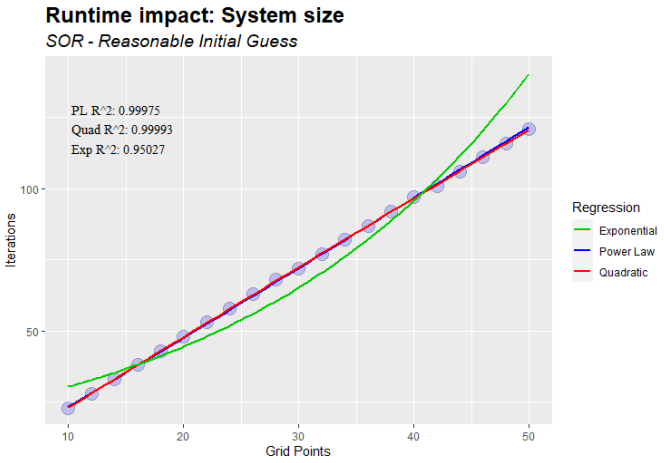

Figure 13: Analysis of runtime impact for the SOR method with a reasonable initial guess

The quadratic model again appears to fit the model best and has the highest R^2 value. The equations are as follows:

$$\hat{Y}_P = 2.22 \cdot X^{1.02}$$
$$\hat{Y}_Q = -1.99 + 2.52X + 0X^2$$
$$\hat{Y}_E = 20.69 + e^{0.04X}$$

The number of iterations for the SOR method is a fifth of what were required for the Jacobi method, illustrating the accelerated rate of convergence inherent in the method. The function is almost linear, hence the exponential coefficient being close to one and the quadratic term coefficient being zero. Here both the power law and quadratic models are almost equivalent.

3.4 Part (c)

Examine the impact of grid size on the performance of the SOR method using a poor initial guess (zeros on the interior).

Here, the SOR method was implemented with the interior array initialized to a very small number.

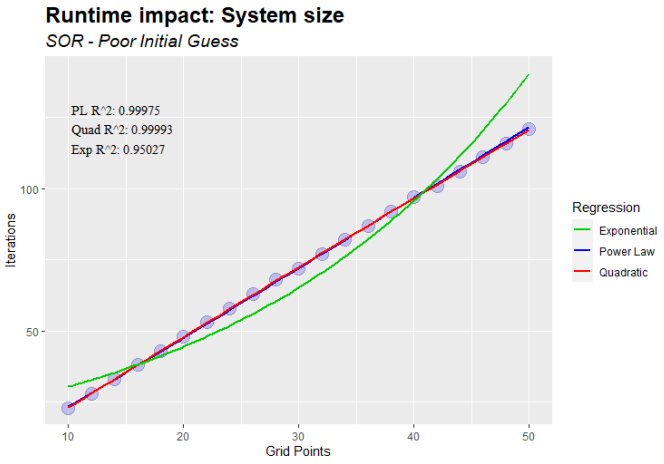

Figure 14: Analysis of runtime impact for the SOR method with a poor initial guess

The SOR method with a poor initial guess is identical to the SOR method with a reasonable initial guess. The equations are as follows:

$$\hat{Y}_P = 2.22 \cdot X^{1.02}$$
$$\hat{Y}_Q = -1.99 + 2.52X + 0X^2$$
$$\hat{Y}_E = 20.69 + e^{0.04X}$$

The power law model allows us a good single metric to determine the impact of grid size on computational performance with the exponential coefficient. The closer this exponent is to one, the closer to linear the runtime impact is. As the exponent grows, we can see that larger grid sizes will have a larger impact.

4 Problem 8.7

Use the `relax` program to calculate the electric field from the scalar potential calculated from the physical system given in problem 1
The electric field is given by:

$$\mathbf{E} = -\nabla\Phi$$

This is implemented in Python as

```
[Ex, Ey] = np.gradient(np.flipud(np.rot90(phi)))
for i in range(N) :
    for j in range(N) :
        magnitude = np.sqrt(Ex[i,j]**2 + Ey[i,j]**2)
        Ex[i,j] /= -magnitude     # Normalize components so
        Ey[i,j] /= -magnitude     # vectors have equal length
        Ex[i,j] /= -1     # Normalize components so
```

```
        Ey[i,j] /= -1      # vectors have equal length
Ex = Ex[~np.isnan(Ex)]
Ey = Ey[~np.isnan(Ey)]
...
plt.quiver(Xp,Yp,Ey,Ex,np.flipud(np.rot90(phi)), scale = 60)  # Vector field
```

Inside the iteration, there are two pairs of transforms for the electric field values. In the first, the arrows are normalized and negated to give us the proper direction. The second pair of transforms simply negates the value, allowing the magnitude of the arrows to be proportional to their value. There are boundary points where the scalar field is zero, causing `NaN` values in the electric field plot that must be removed.

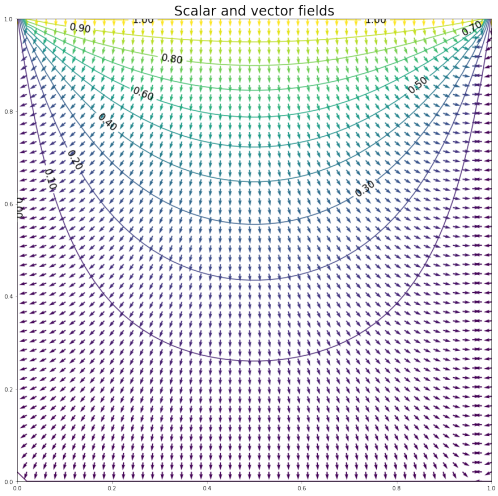

Figure 15: Homogeneous electric field vectors plotted over contour map of scalar field

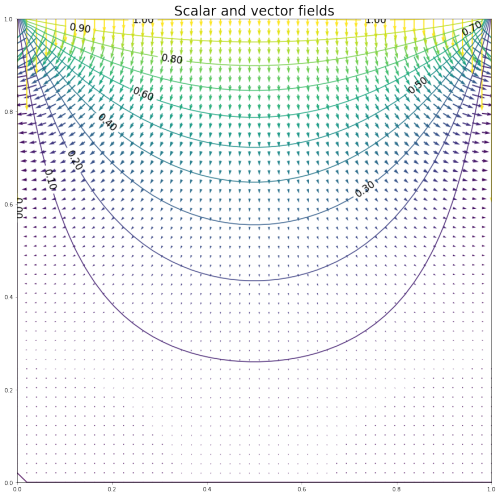

Figure 16: Proportional electric field vectors plotted over contour map of scalar field

5 Problem 8.8

Faraday cage

I did this problem by accident before I realized that it wasn't assigned, so then I changed it around a bit and made the Faraday cage continuous. Producing the results in Figures 17 and 18.

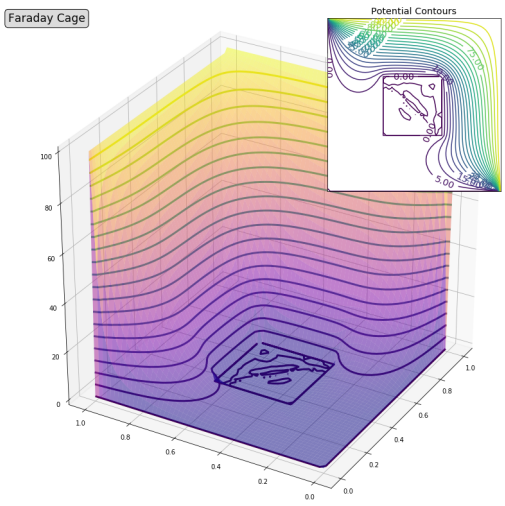

Figure 17: Potential around a Faraday cage. Coarse contours

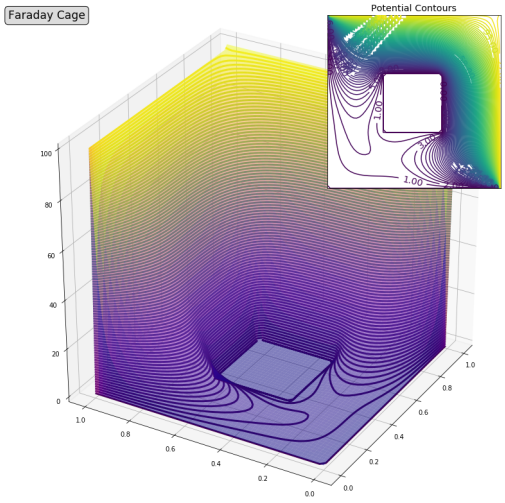

Figure 18: Potential around a Faraday cage. Fine contours.

6 Problem 8.10

6.1 Part (a)

Solve the two-dimensional Poisson equation for a single line charge in a square domain with Dirichlet boundary conditions of $\Phi = 0$ and compare to the scalar potential field of a single charge in free space.

As this problem is a comparative exercise, I decided to normalize the potential values – I removed any constants from the system and set the charge density of the source to one. This made the implementation very simple; the source was held constant over the iteration and the solution was allowed to converge identically to the method that was used for solving the Laplace equation.

```
phi[24,24] = charge
...
for iter in range(iterMax) :
    changeSum = 0
    for i in range(1,N-1) :
        for j in range(1,N-1) :
            temp = .25*omega*( phi[i+1,j] + phi[i-1,j] +
                               phi[i,j-1] + phi[i,j+1] ) + (1-omega)*phi[i,j]

            changeSum += abs( 1 - phi[i,j]/temp )
            phi[i,j] = temp
            phi[24,24] = charge
```

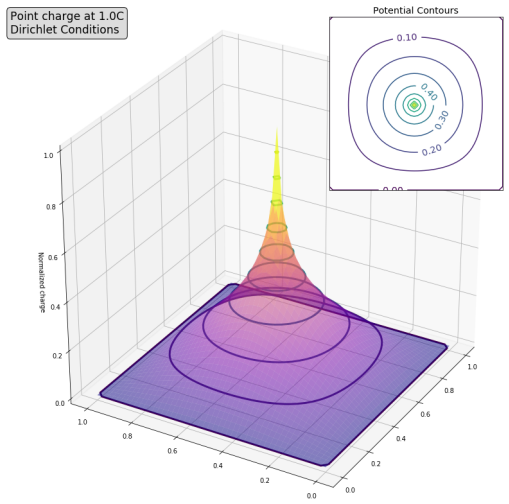

Figure 19: Potential of line charge with Dirichlet boundary conditions

Which led to the results given in Figure 19. The contours are set a one-tenth of the charge density of the line charge. The field of a charge in free space is given as:

$$\phi \propto \frac{1}{r}$$

We can plainly see the inverse radial dependence of the numerical solution to the Poisson equation in the figure. The potential does drop off more rapidly than would be expected analytically, as the solution needs to be continuous and smooth from the source to the boundaries, which are held at zero. These boundary interactions also distort the contours, which we would expect to be circular, to a more rectangular shape as the contours approach the boundary.

6.2 Part(b)

Implement periodic boundary conditions and compare with the results from part (a).
Implementation of periodic boundary conditions was a matter of adding two conditions for each pair of opposite walls.

```
phi[:,0] = phi[:,N-2]; phi[:,N-1] = phi[:,1]      #Top and bottom
phi[0,:] = phi[N-2,:]; phi[N-1,:] = phi[1,:]      #Left and right
```

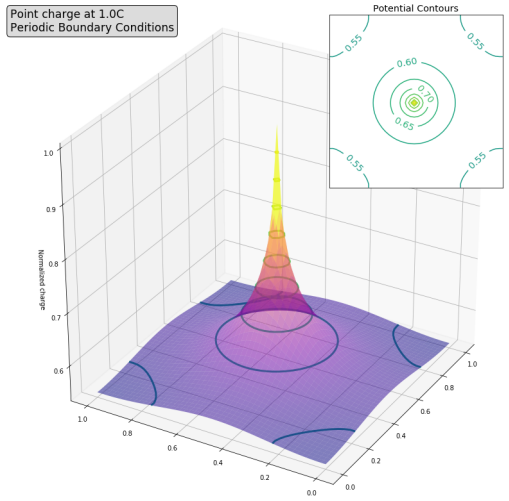

Figure 20: Potential of line charge with periodic boundary conditions

This led to the results shown in Figure 20. Note that the minimum potential here is much higher than it is in the case with Dirichlet conditions. Periodic boundaries for this physical system represent an infinite rectilinear grid of line charges; the potential of a line charge is radial, leading to the interesting pattern of contours that are exhibited in the figure. We can also see that the values of the potential fall off more slowly than they did in part (a) due to the fact that they are not forced to zero at the boundaries.

Extras

I wanted to see what would happen with Neumann boundaries as well, the results are in Figure 21.

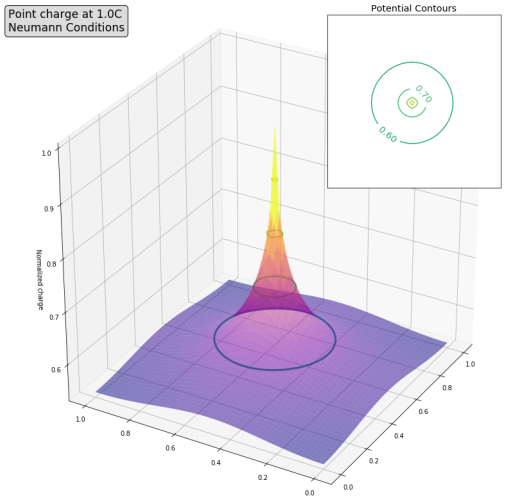

Figure 21: Potential of line charge with Neumann boundary conditions

With contour spacing equivalent to parts (a) and (b), we can see the type of radial behavior that would be more expected from a charge in free space, as the boundaries do not interfere with the value of phi.

I then decided to run simulations for an off-center charge distribution for both Neumann and periodic boundary conditions to ensure that they were correctly implemented.

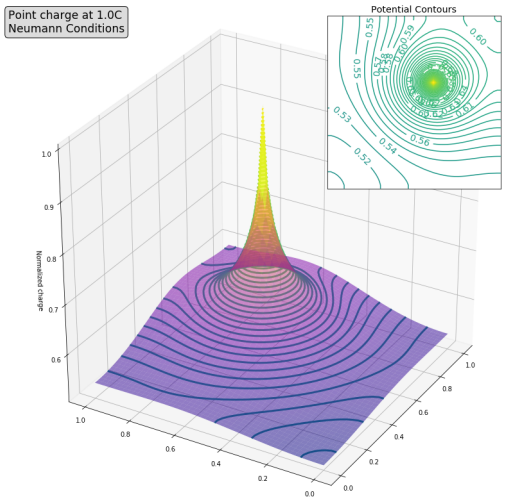

Figure 22: Potential of off-center line charge with Neumann boundary conditions

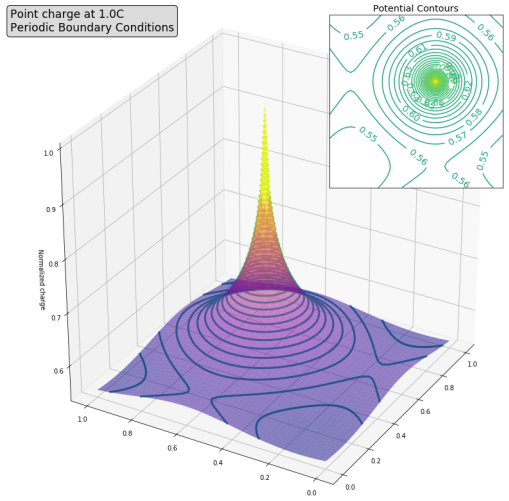

Figure 23: Potential of off-center line charge with periodic boundary conditions

Boy, that sure does look cool! Incidentally, this is also the behavior that we would expect from correctly implemented Neumann and periodic boundary conditions.

Conclusion

In this project we have examined jury and relaxation methods for use in numerically solving elliptic partial differential equations. We have utilized the Jacobi method: a prototypical method that solves for an interior point by taking the average of the adjacent points, the Gauss-Seidel method: a refinement of the Jacobi method that uses the most updated set of points in its average, and the SOR method which introduces an overrelaxation parameter (ω) to improve on the convergence of the Gauss-Seidel method.

We compared our numerical schemes with direct numerical simulations of the analytical infinite series solutions to the elliptic equations that were being investigated – both two- and three-dimensionally. We then examined the effect of both grid size and quality of initial guess on the convergence rate of both the Jacobi method and the SOR method, discovering that SOR is much more resistant to poor initial guesses and that the convergence rate of SOR is a substantial improvement over the Jacobi method. Following this, we proceeded to calculate the electric field from the numerically solved scalar potential field and plotted the results. Finally, we extended our methods to contain a source term in the domain (the Poisson equation) with varying boundary conditions to compare to a well-known physical system.

The technical implementation in this project was very straightforward, although I would like to further investigate optimization schemes for many of the code fragments – the 3-dimensional Laplacian series solution turned out to be a five level nested loop and I'm sure that there are ways of flattening those loops to significantly improve performance. I very much enjoyed the regression analysis (which was done in R as opposed to Python) and the physical implications of the power law model. The majority of difficulty in this project turned out to be in plotting functions, specifically figuring out how to implement the quiver plot for the electric field, but

I am pleased with the results produced. Another difficulty was the analytical solution to the 3-dimensional Laplace equation; I've done this same problem before, but it doesn't make it any easier. This is a strong argument in favor of numerical methods, as the numerical scheme was much easier to implement than the analytical, that and even after the analytical solution was found, we still needed to implement it numerically to get any visualizable results.

Appendices

A: Code

B: Analytical solution to 3-dimensional Laplace equation

References

[1] R. L. Burden and J. D. Faires. Relaxation techniques for solving linear systems, 2011. URL `https://www.math.ust.hk/~mamu/courses/231/Slides/CH07_4A.pdf`.

[2] Alejando L Garcia. *Numerical methods for physics*. 2015. ISBN 9781514136683.

[3] Richard Haberman and Richard Haberman. *Applied partial differential equations: with Fourier series and boundary value problems*. Pearson Prentice Hall, 4th ed edition, 2004. ISBN 9780130652430.