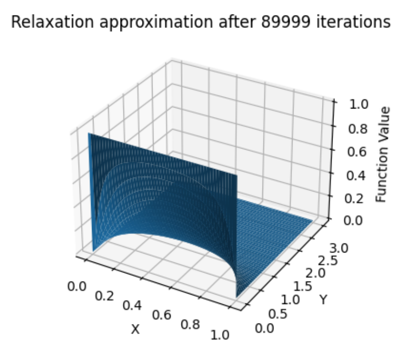
Physics 322 Honors Project

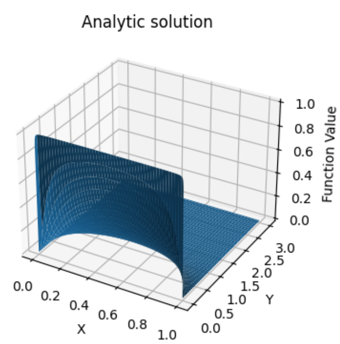
Henry Schnieders
University of Wisconsin–Madison

May 6, 2025

Abstract

This project is about solving the 2D Laplace equation using a relaxation method. The solution is compared to an analytic solution.





Descritization of $\nabla^2 u$ in a 2D domain:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}.$$

$$\begin{aligned}
u_x^+ &= \lim_{h_x \to 0} \frac{u(x+h_x, y) - u(x,y)}{h_x}, \\
u_x^- &= \lim_{h_x \to 0} \frac{u(x, y) - u(x-h_x,y)}{h_x}, \\
u_{xx} &= \lim_{h_x \to 0} \frac{u_x^+ - u_x^-}{h_x} = \lim_{h_x \to 0} \frac{1}{h_x}\Big[\frac{u(x+h_x, y) - u(x,y)}{h_x} - \frac{u(x, y) - u(x-h_x,y)}{h_x}\Big] \approx u_{xx} \quad \text{(for finite } h_x\text{)}, \\
\nabla^2 u &= \lim_{h_x \to 0} \frac{u_x^+ - u_x^-}{h_x} + \lim_{h_y \to 0} \frac{u_y^+ - u_y^-}{h_y} \approx \frac{u_x^+ - u_x^-}{h_x} + \frac{u_y^+ - u_y^-}{h_y}, \\
0 &\approx -2\,u(x,y)\Big(\tfrac{1}{h_x^2} + \tfrac{1}{h_y^2}\Big) + \frac{u(x+h_x,y) + u(x-h_x,y)}{h_x^2} + \frac{u(x,y+h_y) + u(x,y-h_y)}{h_y^2}, \\
u(x,y) &\approx \frac{h_y^2\big[u(x+h_x,y) + u(x-h_x,y)\big] \;+\; h_x^2\big[u(x,y+h_y) + u(x,y-h_y)\big]}{2\,(h_x^2 + h_y^2)}.
\end{aligned}$$

Analytic solution on the semi-infinite strip $0 < x < 1,\ 0 < y < \infty,$

$$\nabla^2 u = 0.$$

Assume $u(x,y) = X(x)\,Y(y)$. Then

$$\frac{X''(x)}{X(x)} + \frac{Y''(y)}{Y(y)} \;=\; 0 \implies \frac{X''(x)}{X(x)} \;=\; -\,\frac{Y''(y)}{Y(y)} \;=\; -\,\lambda.$$

Hence we get two ODEs:

$$\begin{cases} X''(x) + \lambda\,X(x) = 0, \\ Y''(y) - \lambda\,Y(y) = 0. \end{cases}$$

(1) The X–problem:

$$X'' + \lambda X = 0, \qquad X(x) = A\cos(\sqrt{\lambda}\,x) + B\sin(\sqrt{\lambda}\,x),$$

with boundary conditions $X(0) = 0,\ X(1) = 0$ imply

$$A = 0, \quad \sin(\sqrt{\lambda}) = 0 \implies \sqrt{\lambda} = n\pi,\ \lambda = (n\pi)^2,\ n \in \mathbb{N},$$

so

$$X_n(x) = \sin(n\pi x).$$

(2) The Y–problem:

$$Y'' - \lambda Y = 0, \quad Y(y) = C\, e^{\sqrt{\lambda}\, y} + D\, e^{-\sqrt{\lambda}\, y},$$

and the decay condition $\lim_{y\to\infty} Y(y) = 0$ forces $C = 0$, hence

$$Y_n(y) = e^{-n\pi y}.$$

By superposition,

$$u(x, y) = \sum_{n=1}^{\infty} C_n \sin(n\pi x)\, e^{-\, n\pi y}.$$

Enforcing the boundary data at $y = 0$,

$$u(0, x) = 1 = \sum_{n=1}^{\infty} C_n \sin(n\pi x)$$

gives the Fourier–sine coefficients

$$C_n = \frac{1}{2}\int_0^1 \sin(n\pi x)\, dx = \begin{cases} \dfrac{1}{n\pi}, & n \text{ even}, \\ 0, & n \text{ odd}. \end{cases}$$

Error metric:

In this project I used the Root-Mean-Square error (RMSE) to measure the error between the numerical and analytical solutions.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i,j} e_{ij}^2}, \quad e_{ij} = u_{\text{num}}(x_i, y_j) - u_{\text{analy}}(x_i, y_j), \quad N = N_{\text{rows}} \cdot N_{\text{cols}} = 500 \cdot 100.$$

$$\text{RMS}\big(u_{\text{analy}}\big) = \sqrt{\frac{1}{N}\sum_{i,j}\big(u_{\text{analy}}(x_i, y_j)\big)^2}.$$

$$\text{Percent accuracy} = 100\left[1 - \frac{\text{RMSE}}{\text{RMS}\big(u_{\text{analy}}\big)}\right].$$

Coded solution:

For simplicity, I chose y_{max} sufficiently large so that the RMSE is small for boundary condition $u(x, y_{max}) = 0$. Setting $y_{max} = 3$, the percent error of u_{num}, u_{analy} at $y = y_{max}$ is less than or equal to $100 * e^{-3*\pi} \approx 0.19\%$.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

ROWS = 100
COLS = 500
ITERS = 90000

# set the relative length scales
Lx = 1.0
Ly = 3.0

# hx, hy
dx = Lx / (COLS - 1)
dy = Ly / (ROWS - 1)

u_0 = np.zeros((ROWS,COLS))

# set the boundary conditions
u_0[0, :]        = 1
u_0[ROWS-1, :]   = 0
u_0[:, 0]        = 0
u_0[:, COLS-1]   = 0

# define the analytic solution
def analytic_result(x, y, nmax=1000):
    total = 0
    for n in range(1, nmax, 2):
        coeff   = 4 / (n * np.pi)
        x_comp  = np.sin((n*np.pi*x) / Lx)
        y_comp  = np.exp((-n*np.pi*y) / Lx)
        total  += coeff * x_comp * y_comp
    return total

x = np.linspace(0, Lx, COLS)
y = np.linspace(0, Ly, ROWS)
X, Y = np.meshgrid(x, y)

Z = np.zeros_like(u_0)
for row in range(ROWS):
    for col in range(COLS):
```

```python
        Z[row, col] = analytic_result(X[row, col], Y[row, col])

denom = 2*(dx**2 + dy**2)

u_k = np.zeros_like(u_0)
u_k[0, :]      = 1
u_k[ROWS-1, :] = 0
u_k[:, 0]      = 0
u_k[:, COLS-1] = 0

# Relaxation approximation loop
for iter in range(ITERS):

    if iter % 100 == 0:
        if iter % 1000 == 0:
            print(f"iteration {iter}")
        err = Z - u_0
        rms_err    = np.sqrt(np.mean(err**2))
        rms_exact  = np.sqrt(np.mean(Z**2))
        percent_accuracy = 100 * (1 - rms_err / rms_exact)
        with open('proj_output.txt','a') as f:
            f.write(f"Iteration {iter}\n")
            f.write(f"RMS based % accuracy: {percent_accuracy:.2f}%\n")

    for row in range(1, ROWS-1):
        for col in range(1, COLS-1):
            u_k[row, col] = (
                dy**2*(u_0[row, col+1] + u_0[row, col-1])
                + dx**2*(u_0[row+1, col] + u_0[row-1, col])
            ) / denom

    u_0 = u_k

# Re compute X, Y for plotting
X, Y = np.meshgrid(x, y)

fig = plt.figure(figsize=(12,4))
ax = fig.add_subplot(121, projection='3d')
ax.plot_surface(X, Y, u_0, linewidth=0, antialiased=True)
ax.set_xlabel('X'); ax.set_ylabel('Y'); ax.set_zlabel('Function Value')
ax.set_title(f'Relaxation after {iter} iterations')

ax = fig.add_subplot(122, projection='3d')
ax.plot_surface(X, Y, Z, linewidth=0, antialiased=True)
ax.set_xlabel('X'); ax.set_ylabel('Y'); ax.set_zlabel('Function Value')
ax.set_title('Analytic solution')

plt.show()
```

Results:

```
Iteration 25100
RMS-based % accuracy: 89.98%
Iteration 25200
RMS-based % accuracy: 90.03%

Iteration 71600
RMS-based % accuracy: 98.99%
Iteration 71700
RMS-based % accuracy: 99.00%
```

The method takes approximately 25150 iterations to converge to a solution with 90% accuracy and at most 71700 iterations to converge to a solution with 99% accuracy.