PROJECT 1

JOHN S GEORGE (JOHNS15) *

1. Part 1: An Explicit Solver.

1.1. Problem Setup.

1.1.1. Geometry. The geometry for 225x225 mesh is shown in Figure 1.1. The

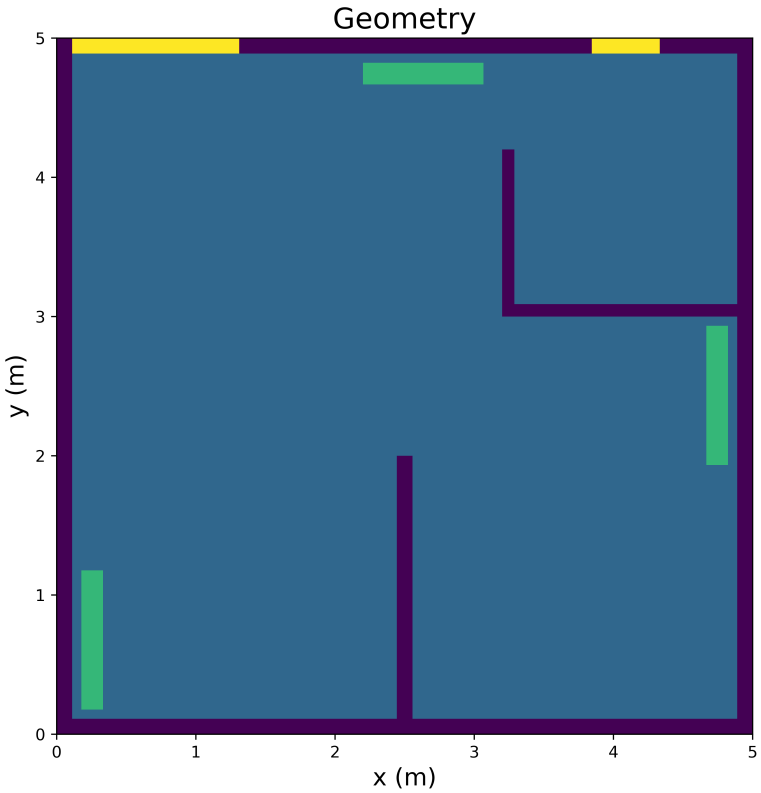


FIG. 1.1. *Room dimension - 5m x 5m. Yellow - window, green - heater.*

result can be reproduced by running the file geometry.py.

1.1.2. Boundary condition handling. The discretized form of the 2D unsteady heat equation using FTCS scheme is shown below:

$$\frac{u_{i,j}^{l+1} - u_{i,j}^{l}}{\Delta t} = \frac{u_{i,j-1}^{l}}{\Delta y^2} + \frac{u_{i-1,j}^{l}}{\Delta x^2} - \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right) u_{i,j}^{l} + \frac{u_{i+1,j}^{l}}{\Delta x^2} + \frac{u_{i,j+1}^{l}}{\Delta y^2} \tag{1.1}$$

For Dirichlet boundary conditions, the known values were substituted in equation (1.1). For Neumann boundary condition a centered difference approach was taken.

*johns15@illinois.edu

Consider the case of a left wall. We have,

$$\frac{\partial u}{\partial x} = 0$$

For a fixed y coordinate, from Taylor's expansion,

$$u(x + \Delta x, y) = u(x, y) + \Delta x \frac{\partial u}{\partial x} + \frac{\Delta x^2}{2} \frac{\partial^2 u}{\partial x^2} + O(\Delta x^3) \tag{1.2}$$

$$u(x - \Delta x, y) = u(x, y) - \Delta x \frac{\partial u}{\partial x} + \frac{\Delta x^2}{2} \frac{\partial^2 u}{\partial x^2} - O(\Delta x^3) \tag{1.3}$$

Subtracting equation (1.3) from equation (1.2), gives,

$$\frac{\partial u}{\partial x} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + O(\Delta x^2) \tag{1.4}$$

At the wall, equation (1.4) can be approximated as,

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} = 0 \tag{1.5}$$

Equation (1.4) is second - order accurate in Δx. $u_{i-1,j}$ can be treated as a ghost point and can be substituted for, in equation (1.1) from equation (1.5). The same approach can be extended to other walls.

1.1.3. Second order accuracy of numerical results.. For numerically testing the second order accuracy of the results, a solution was manufactured for the unsteady equation. The geometry is a rectangular plate with height H (y-axis) and length L(x-axis). Second order implicit midpoint method was used to obtain the unsteady solution. The solution at a final time, t = 0.001 seconds with a time step of 0.0001 was used. The reason for such small time step and final time was due to the time dependence of the right hand side of the equation, which caused a significant increase in computation time. East, west and south ends of the geometry are at a fixed temperature of 273.16 K and north end is insulated. The problem description is as follows:

$$\Delta u + r(x, y, t) = \frac{\partial u}{\partial t} \tag{1.6a}$$

$$u(x, 0, t) = 273.16K \tag{1.6b}$$

$$u(0, y, t) = 273.16K \tag{1.6c}$$

$$u(L, y, t) = 273.16K \tag{1.6d}$$

$$\frac{\partial u}{\partial y}_{(x,H,t)} = 0 \tag{1.6e}$$

Manufactured solution that satisfy equations (1.6) is given by:

$$u(x, y, t) = \sin\left(\frac{\pi x}{L}\right) \sin\left(\frac{\pi y}{2H}\right)(373.16 + \cos(t)) + 273.16 \tag{1.7}$$

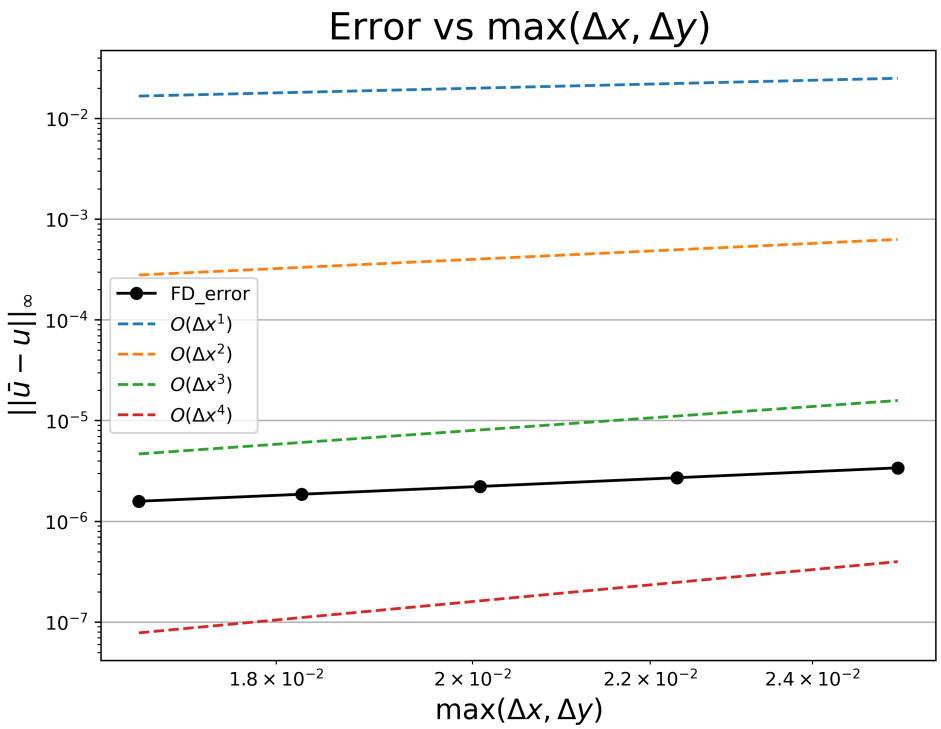


FIG. 1.2. *Accuracy test for unsteady system using implicit midpoint method.*

Figure 1.2 shows the log-log infinity norm absolute error of the numerical solution at t = 0.001 s, with respect to $max(\Delta x, \Delta y)$. The test was conducted for mesh sizes: 200x200, 225x225, 250x250, 275x275 and 300x300. From Figure 1.2, it can be observed that the numerical scheme is second-order accurate. A test for same grid spacing was also conducted for steady state solution given by:

$$u(x, y) = \sin\left(\frac{\pi x}{L}\right) \sin\left(\frac{\pi y}{2H}\right) + 373.16 \tag{1.8}$$

The boundary conditions are same as in equation (1.6). The result is shown below:
These results can be obtained by running the file testfdm.py.

1.2. Von Neumann Stability Analysis. The inverse discrete fourier transform for an infinite vector on spatial domain is given by,

$$u^l_{j,k} = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \hat{u}^l(\theta, \phi) e^{\iota k\theta} e^{\iota j\phi} d\theta d\phi \tag{1.9}$$

Substituting equation (1.9) in equation (1.1), we get,

$$\frac{\hat{u}^{l+1}(\theta, \phi)}{\hat{u}^l(\theta, \phi)} = \frac{\Delta t}{(\Delta y)^2}(e^{-\iota\theta} + e^{\iota\theta}) + \frac{\Delta t}{(\Delta x)^2}(e^{-\iota\phi} + e^{\iota\phi}) + \Delta t\Big(\frac{1}{\Delta t} - \frac{2}{(\Delta y)^2} - \frac{2}{(\Delta x)^2}\Big)$$

which on simplification yields,

$$\frac{\hat{u}^{l+1}(\theta, \phi)}{\hat{u}^l(\theta, \phi)} = 1 - \Big(\frac{4\Delta t}{(\Delta x)^2} \sin^2(\phi/2) + \frac{4\Delta t}{(\Delta y)^2} \sin^2(\theta/2)\Big)$$

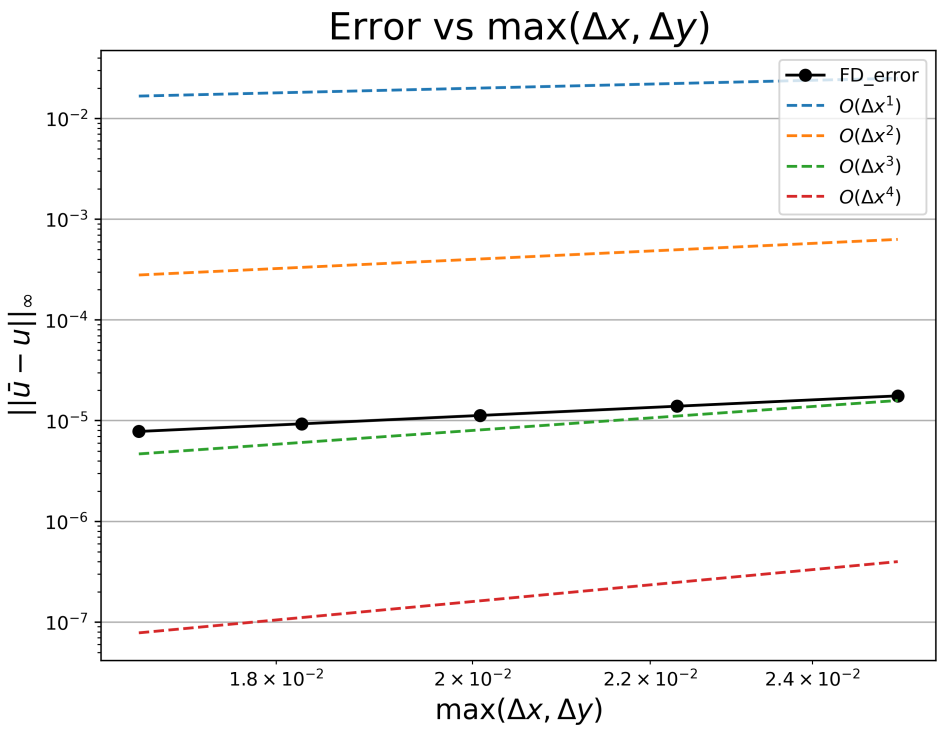


FIG. 1.3. *Accuracy test for steady state system.*

For stability,

$$\left|1 - \frac{4\Delta t}{(\Delta x)^2}\sin^2(\phi/2) - \frac{4\Delta t}{(\Delta y)^2}\sin^2(\theta/2)\right| \leq 1$$

or,

$$\frac{4\Delta t}{(\Delta x)^2}\sin^2(\phi/2) + \frac{4\Delta t}{(\Delta y)^2}\sin^2(\theta/2) \leq 2 \tag{1.10}$$

Let $\Delta s = \min(\Delta x, \Delta y)$, then,

$$\frac{4\Delta t}{(\Delta s)^2} * max\Big(\sin^2(\phi/2) + \sin^2(\theta/2)\Big) \leq 2$$

$$\frac{4\Delta t}{(\Delta s)^2} * 2 \leq 2$$

So, the condition for stability is,

$$\frac{4\Delta t}{(\Delta s)^2} \leq 1$$

From the point of view of accuracy, there is no benefit in using an explicit time integrator that has an order of accuracy higher than one. For forward Euler scheme, the stability criteria demands that time step, $\Delta t \leq (\Delta x)^2/4$, which in itself make the error in first order scheme less.

1.3. Unsteady solution using explicit time integrator. Forward Euler was chosen as the explicit time integrator for solving the unsteady equation. The result is shown in Figure 1.4, at T = 150 seconds (30*max(sizex, sizey) for a 225x225 grid. The result can be obtained by running the file explicit.py.

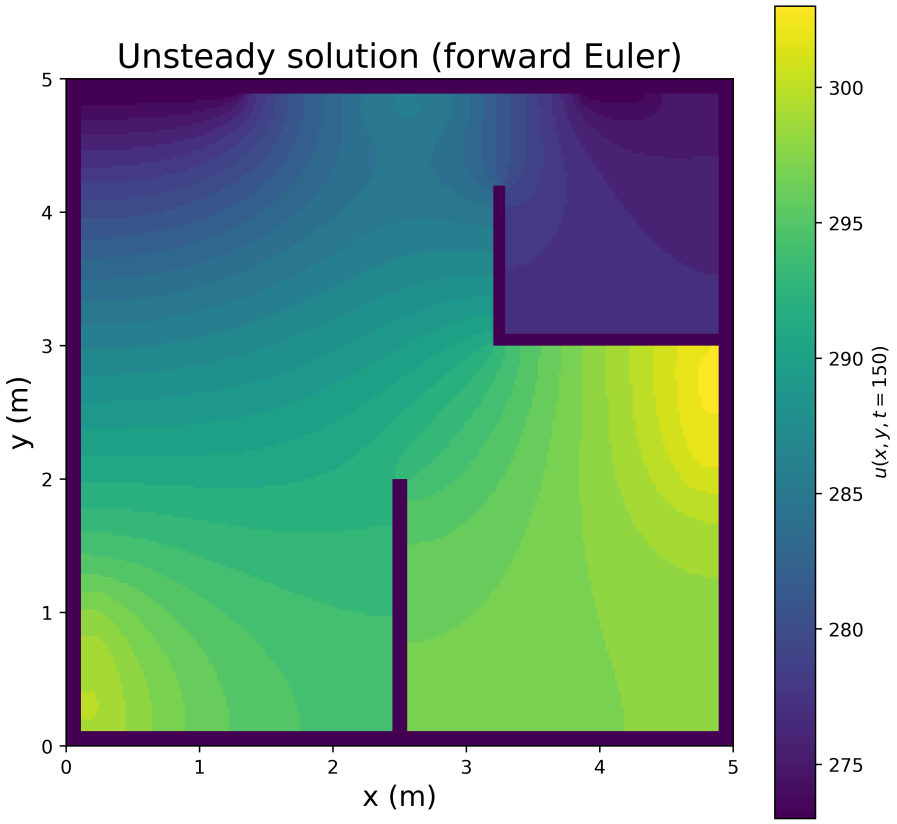


FIG. 1.4. *Unsteady solution using forward Euler at T = 150 seconds*

2. Part 2: Implicit Solver, Steady-State Solution.

2.1. Unsteady solution using implicit time integrator. Second order implicit midpoint method was chosen as the time integrator. The result is shown in Figure 2.1, at T = 150 seconds. The result can be obtained by running the file implicit.py.

2.2. Steady state solution. To solve for steady state, $\frac{\partial u}{\partial t}$ is set to zero. The resulting PDE would be,

$$\Delta u + r(x, y) = 0$$

The result after solving for steady state is shown in Figure 2.2. The result can be obtained by running the file steady.py.

2.3. Jacobi Iteration. The initial guess is shown in Figure 2.3a and the result of 300 iterations of undamped Jacobi is shown in Figure 2.3b. The result of undamped

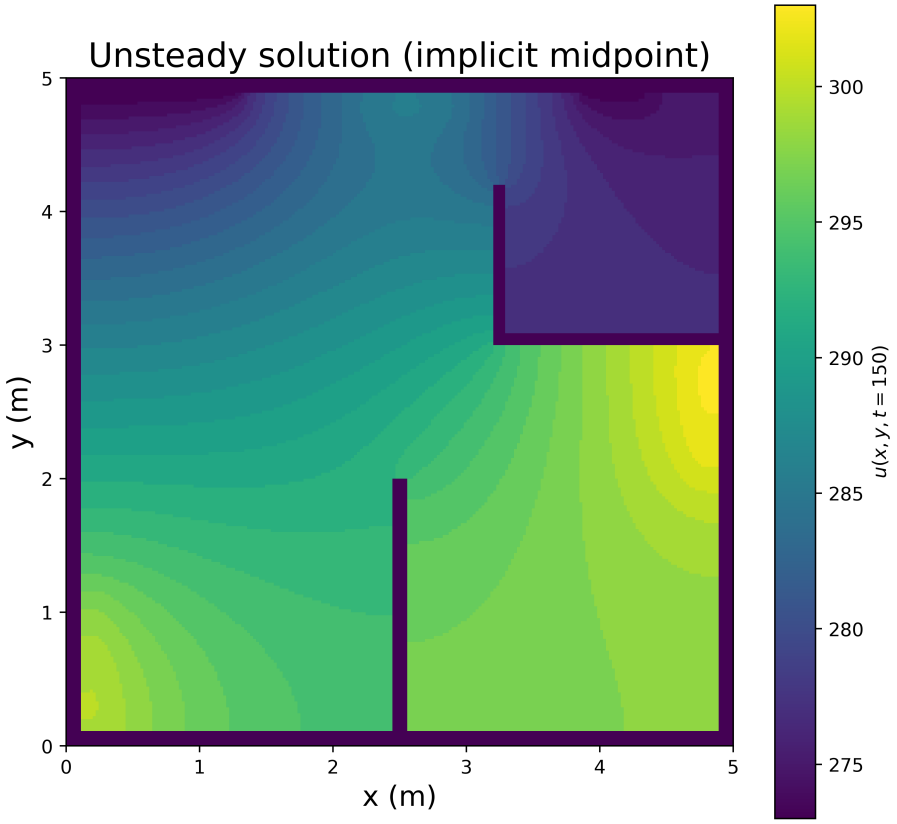


FIG. 2.1. *Unsteady solution using 2nd order implicit midpoint method at $T = 150$ seconds*

Jacobi iteration is grainier compared to damped Jacobi. For solving a system of equation $Au = b$, the Jacobi iteration is given by,

$$u^{l+1} = (I - D^{-1}A)u^l + D^{-1}b = Mu^1 + b^{'} \tag{2.1}$$

where D is the diagonal matrix composed of diagonal elements of A. Equation (2.1) can be interpreted as explicit integration in time. The eigenvalues of matrix M lies in the range [-1,1]. From the stability theory of explicit integrators, the spectral radius of matrix M should be less than 1, for the method to be stable. However, for undamped Jacobi, the spectral radius is at the stability limit. The Jacobi iteration can also be thought of as a power method. u^{l+1} will eventually start to approximate the eigenvector corresponding to the eigenvalue with maximum absolute value. In the case of undamped Jacobi, this eigenvector is oscillatory, which explains the apparent graininess of the result.

2.4. Jacobi Smoother. The result of 300 iterations of damped Jacobi for $\alpha = 0.05$ is shown in Figure 2.4. From equation (2.1), the damped Jacobi iteration can be written as,

$$u^{l+1} = (\alpha I + (1 - \alpha)M)u^l + (1 - \alpha)b^{'} = I - (1 - \alpha)D^{-1}A \tag{2.2}$$

In case of damped Jacobi, the eigenvector corresponding to the dominant eigenvalue is relatively less oscillatory which explains the apparent smoothness in the result.

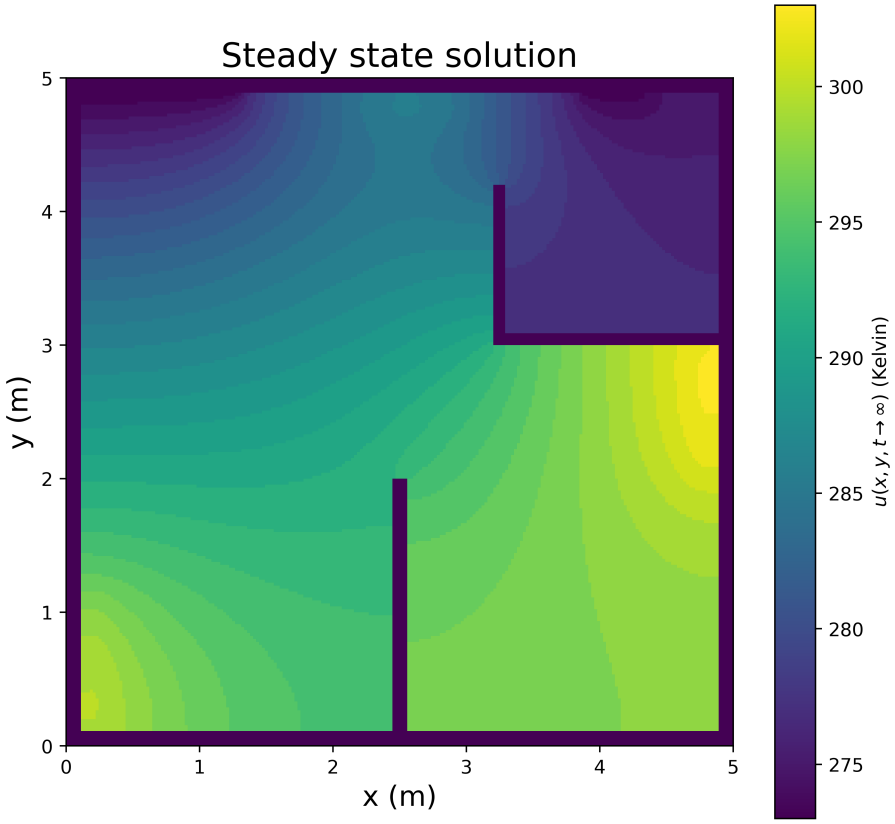


FIG. 2.2. *Steady state solution*

The results for damped and undamped jacobi can be obtained by running the file jacobi.py

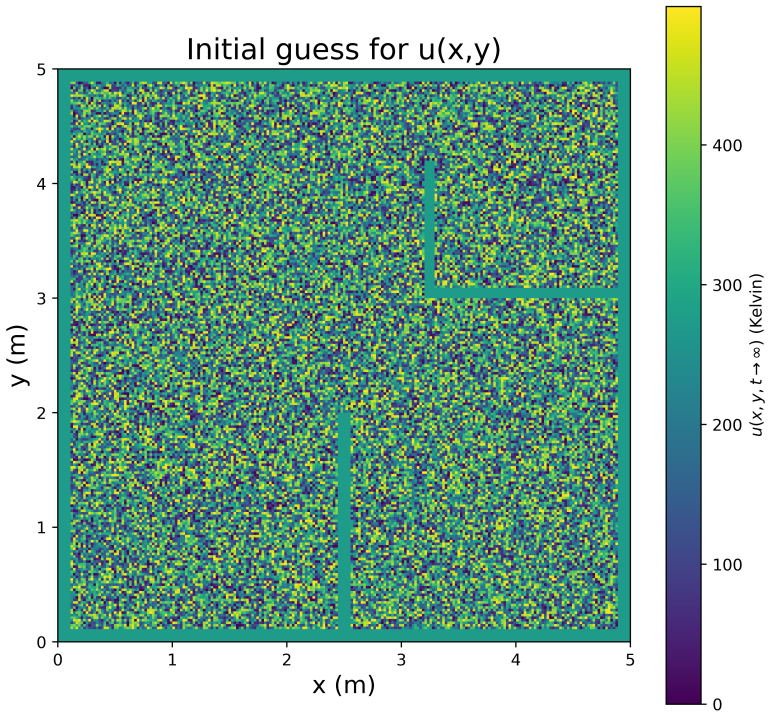
3. Part 3: Geometric Multigrid.

3.1. Accuracy of interpolation and restriction. For the purpose of testing the accuracy of interpolation and restriction, the manufactured solution described in equation (1.8) with the same boundary conditions was used. Solution on the finest grids of size 301x301, 275x275, 251x251, 225x225, 201x201 was restricted to the corresponding coarse grid of size 151x151, 138x138, 126x126, 113x113, 101x101 respectively and interpolated back. The result of these experiments are shown in Figure 3.1 and it indicates that restriction and prolongation operations are second order accurate. The result can be obtained by running mg_test.py.

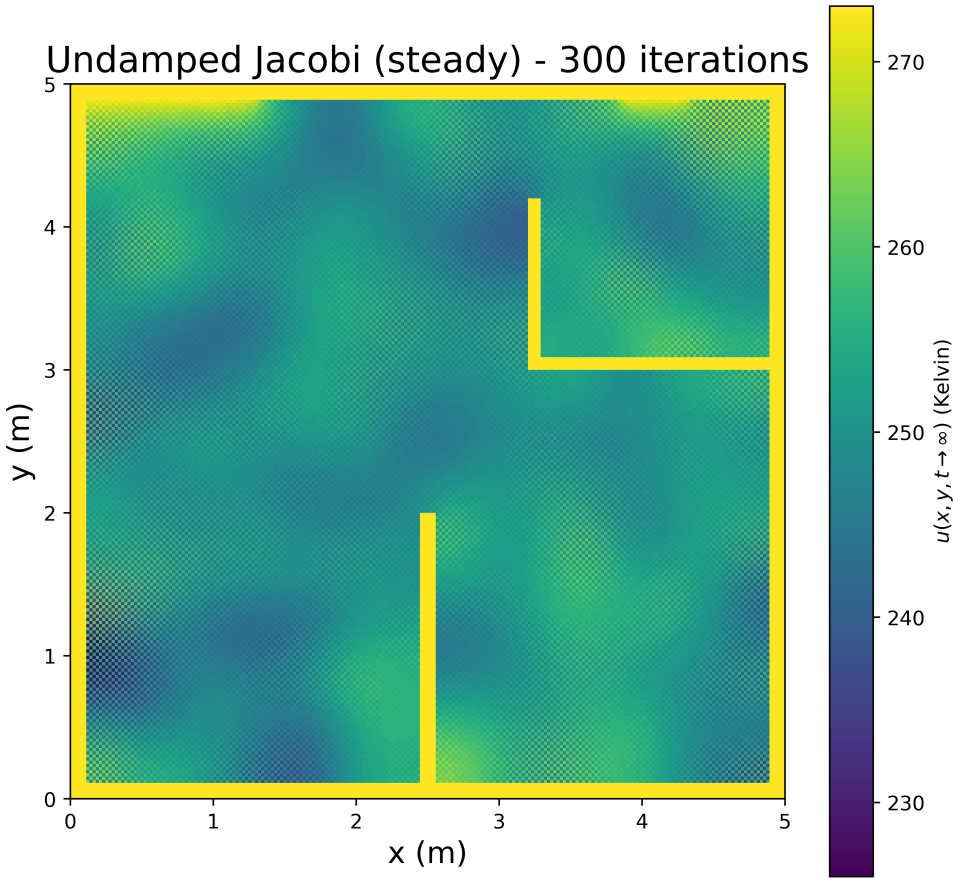
3.2. Multigrid V-cycle. A two-level multigrid v-cycle of coarsening factor 2 was implemented. The dimensions of meshes are 401x401, 201x201 and 101x101. The semi-logarithmic plot of the residual norm against the iteration count is shown in Figure 3.2. The iterations were run until the residual norm decreased by a factor of 10^{12}.

The temperature distribution is shown in Figure 3.3.

The results can be obtained by running the file mg_solve.py.



(a) Initial random guess



(b) Undamped Jacobi - 300 iterations

FIG. 2.3.

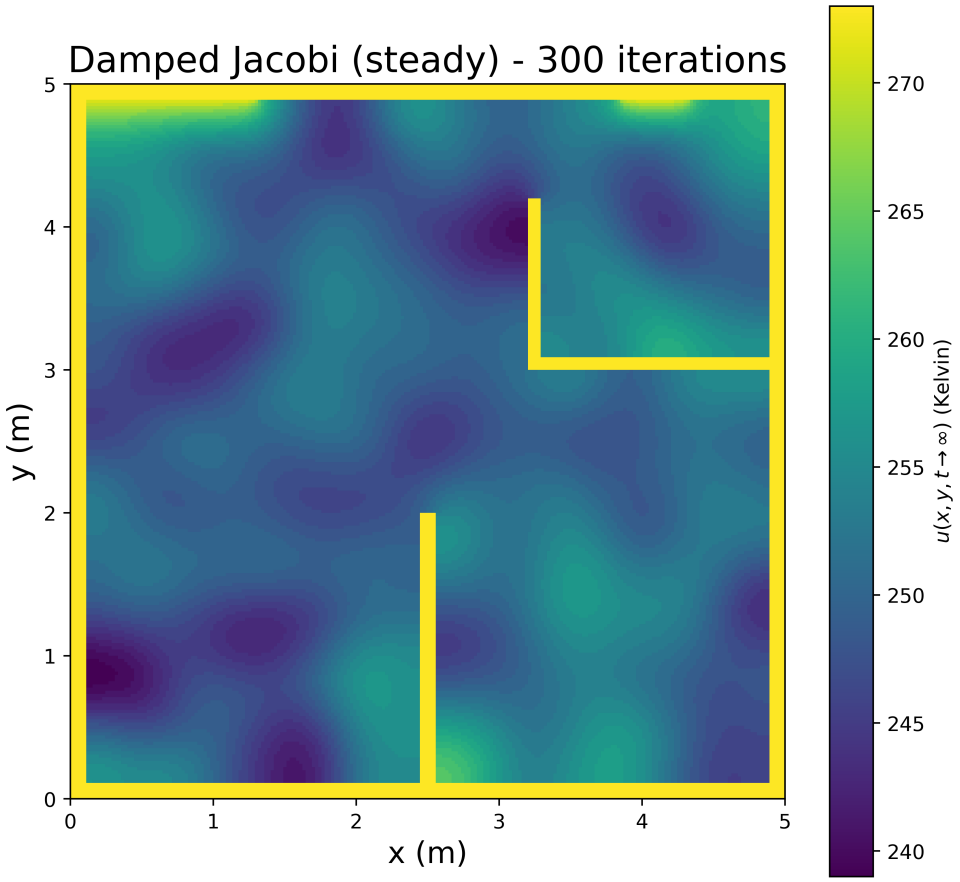


FIG. 2.4. *Damped Jacobi - 300 iterations, $\alpha = 0.05$*

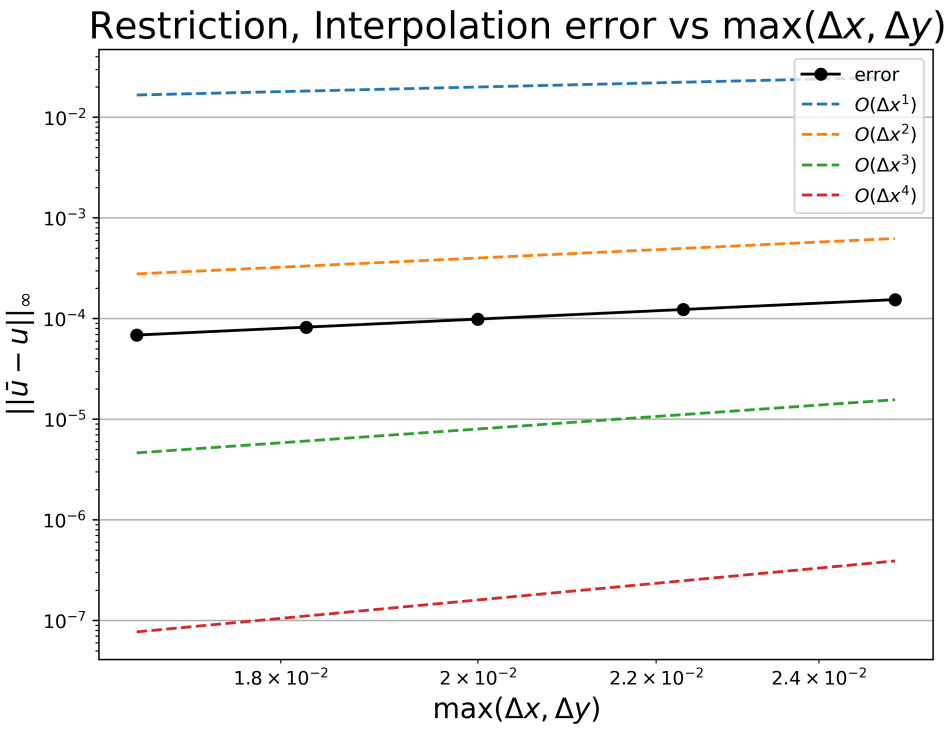


FIG. 3.1. *Restriction-prolongation error on fine grid of size 301x301, 275x275, 251x251, 225x225, 201x201.*

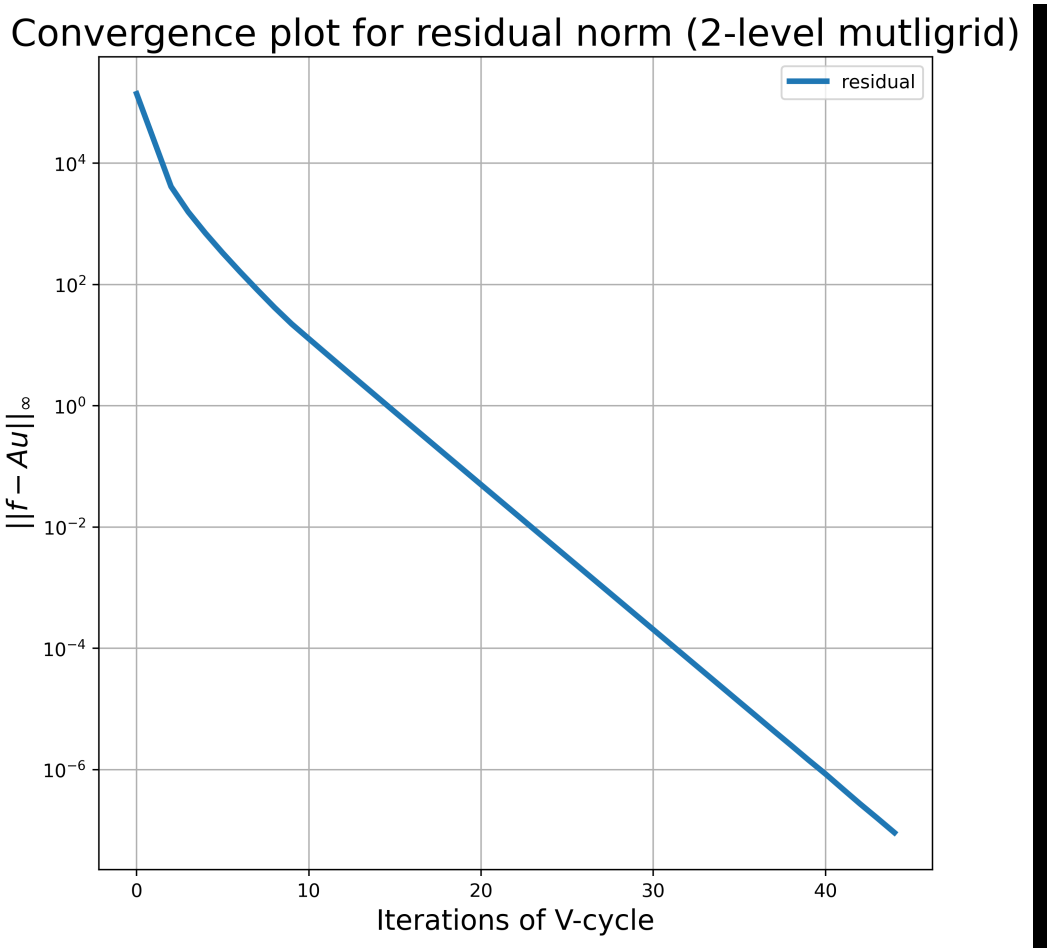


FIG. 3.2. *Two-level multigrid v-cycle convergence plot for steady state system.*

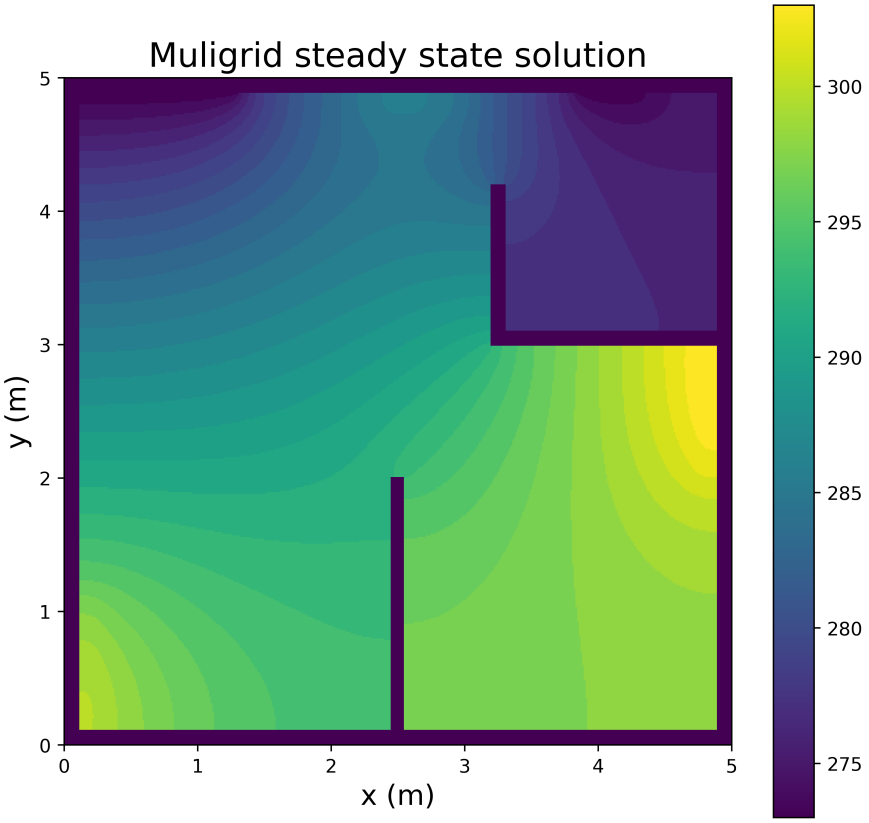


FIG. 3.3. *Two-level multigrid solution. Finest mesh - 401x401, 2-factor coarsening.*