Project 1

Zhanna Sakayeva

March 2022

We are given the heat equation

$$
\begin{aligned}
u_t = \Delta u + r(x) = (\partial_x^2 + \partial_y^2)u + r(x) \\
u(x,0) = 0 \quad (x \in \Omega), \\
u(x,t) = 0 \quad (x \in \Gamma_W), \\
\hat{n} \cdot \nabla u(x,t) = 0 \quad (x \in \partial\Omega \setminus \Gamma_W)
\end{aligned}
\tag{1}
$$

on a complex two-dimensional domain. Where r is a time-constant right-hand-side function corresponding to the heat sources. The condition Γ_W is called a homogeneous *Dirichlet boundary condition.* The condition $\partial\Omega \setminus \Gamma_W)$ is called a homogeneous *Neumann boundary condition.*

Part 1: An Explicit Solver

1. Below is my room geometry that I will be using in my further analysis.

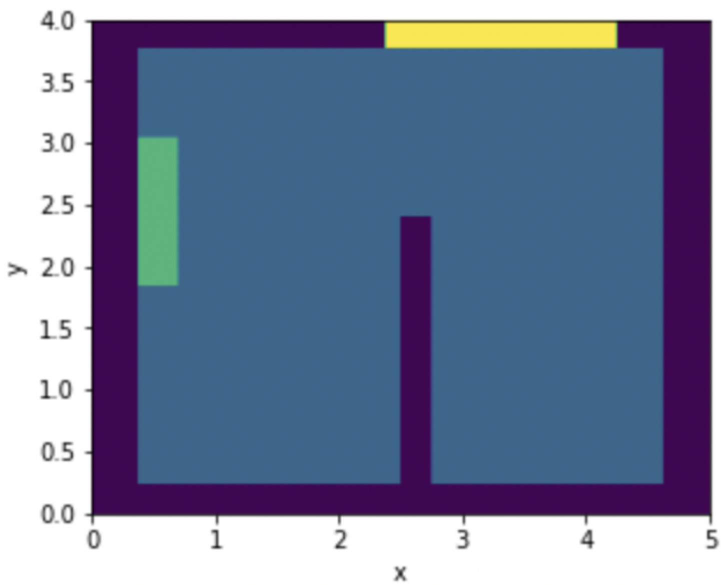

Figure 1: Room geometry

Laplacian matrix can be created using

$$\frac{u_{k-1,l} - u_{k,l}}{h_x^2} + \frac{u_{k+1,l} - u_{k,l}}{h_x^2} + \frac{u_{k,l-1} - u_{k,l}}{h_y^2} + \frac{u_{k,l+1} - u_{k,l}}{h_y^2} = 0.$$

The accuracy and order for Neumann boundary condition. We apply second order forward finite difference and by Taylor expansion

$$u_{i+1} = u_i + hu_i' + h^2 \frac{u_i''}{2!} + \xi_1,$$

$$u_{i+2} = u_i + 2hu_i' + (2h)^2 \frac{u_i''}{2!} + \xi_1 + \xi_2.$$

This will give us

$$\partial_n u(x_i) = u_i' = \frac{-\frac{3}{2}u_i + 2u_{i+1} - \frac{1}{2}u_{i-1}}{h^2} + O(h^2).$$

We can investigate the convergence by considering the simpler geometry and by estimating the infinity norm of difference between true solution and numerical solution.

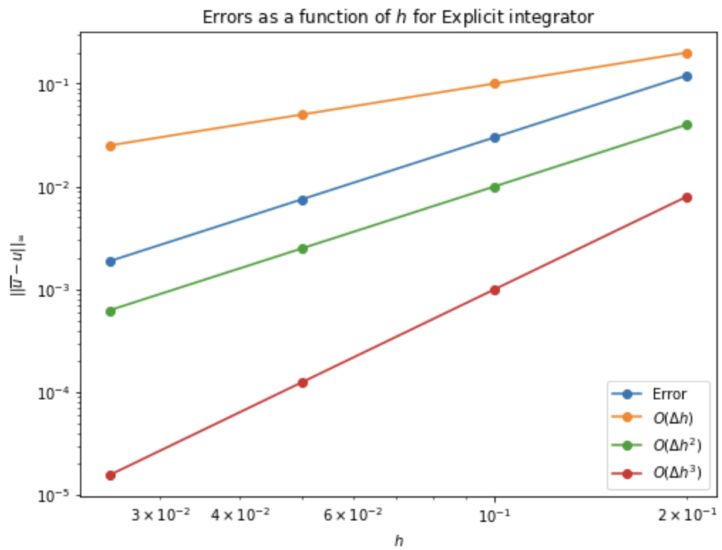

Figure 2: Convergence plot

2. In order to perform von Neumann analysis, we will first apply finite difference method

$$\frac{u_{k,m,l+1}-u_{k,m,l}}{h_t}=\frac{u_{k+1,m,l}-2u_{k,m,l}+u_{k-1,m,l}}{h_x^2}+\frac{u_{k,m+1,l}-2u_{k,m,l}+u_{k,m-1,l}}{h_y^2},$$

$$u_{k,m,l+1}=u_{k,m,l}+\frac{h_t}{h_x^2}(u_{k+1,m,l}-2u_{k,m,l}+u_{k-1,m,l})+\frac{h_t}{h_y^2}(u_{k,m+1,l}-2u_{k,m,l}+u_{k,m-1,l}). \tag{2}$$

Now let us consider discrete solution using Fourier transform

$$u_{k,m,l}=\sum_{v=0}^{n_x}\sum_{z=0}^{n_y}\alpha e^{2i\pi(\frac{vk}{n_x}+\frac{zm}{n_y})},\quad k\in 0,...,n_x, m\in 0,...,n_y$$

We can plug this in into (2) by letting $\omega_x=v/n_x$ and $\omega_y=z/n_y$. If we consider

$$\begin{aligned}(u_{k+1,m,l}-2u_{k,m,l}+u_{k-1,m,l}) &= e^{i2\pi\omega(k+1)-2e^{i2\pi\omega k}+e^{i2\pi\omega(k-1)}}\\ &= e^{i2\pi k\omega}(e^{i2\pi\omega}+e^{-i2\pi\omega}-2)=(2cos(2\pi\omega)-2)u_k\\ &= -4sin^2(\pi\omega)w_k.\end{aligned}$$

So we can utilize the above expression in our analysis for x and y variables. Now apply the Forward Euler method to (2).

$$u_{l+1}=(1-\frac{4h_t}{h_x^2}sin^2(\pi\omega_x)-\frac{4h_t}{h_y^2}sin^2(\pi\omega_y))u_l.$$

So from that we get CFL condition

$$\left|(1 - \frac{4h_t}{h_x^2} sin^2(\pi\omega_x) - \frac{4h_t}{h_y^2} sin^2(\pi\omega_y))\right| \leq 1,$$

or

$$h_t \leq \frac{1}{2} \frac{1}{\frac{1}{h_x^2} + \frac{1}{h_y^2}}.$$

This is the time step for a forward Euler solver. I will be using second order Runge-Kutta method instead of first order Euler. Because error of Euler's method will be corrected.

3. I am using explicit second order Runge-Kutta. Results of the calculation with explicit time integrator and with grid 200×200.

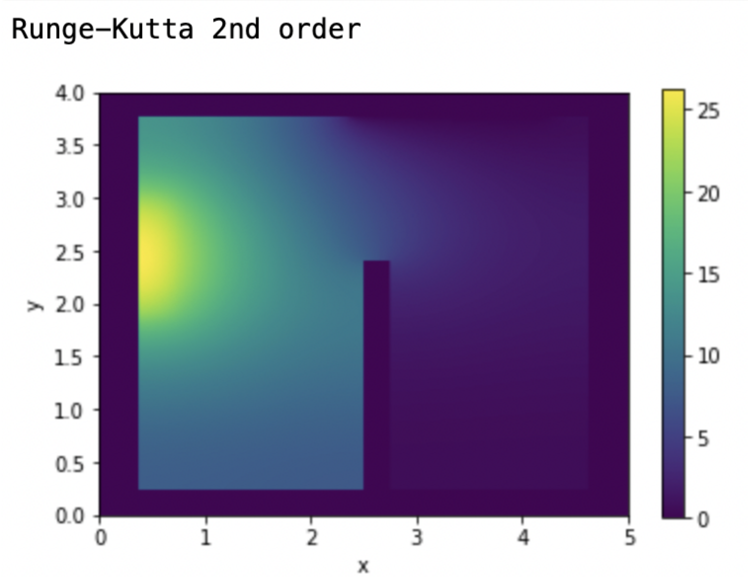

Figure 3: Second order Explicit Runge-Kutta method

Part 2: Implicit Solver, Steady-State Solution

1. I am using second order implicit midpoint method. It was faster than explicit integrator and stable. This is the result with grid 200×200.

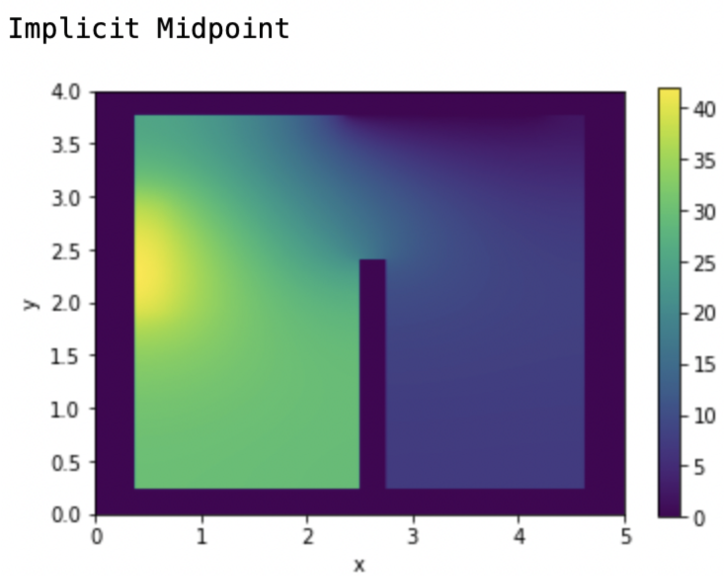

Figure 4: Second order implicit midpoint method

2. So we set $u_t = 0$. Hence we can solve $\Delta u = -r$. The result is given below.

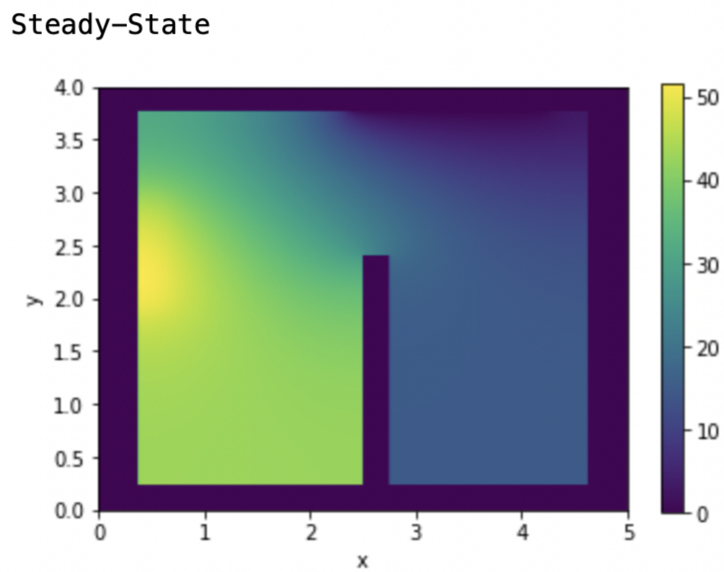

Figure 5: Steady-State

3. Jacobi solution is grainier than damped Jacobi solution. He said that we can interpret Jacobi as Forward Euler. We have

$$Dx_{i+1} = -r - (\Delta - D)x_i.$$

So there is negative Laplacian which positive definite and have positive eigenvalues. The result with grid 50×50 and 300 iterations.

4. We see that damped Jacobi is smoother. It improves the convergence of Jacobi. Since each time we are using the previous Jacobi iterate. So eventually it'll reach steady state. Undamped Jacobi is equivalent to time stepping exactly at the stability limit, so it repeatedly overshoots. The result with grid 50×50 and 300 iterations.

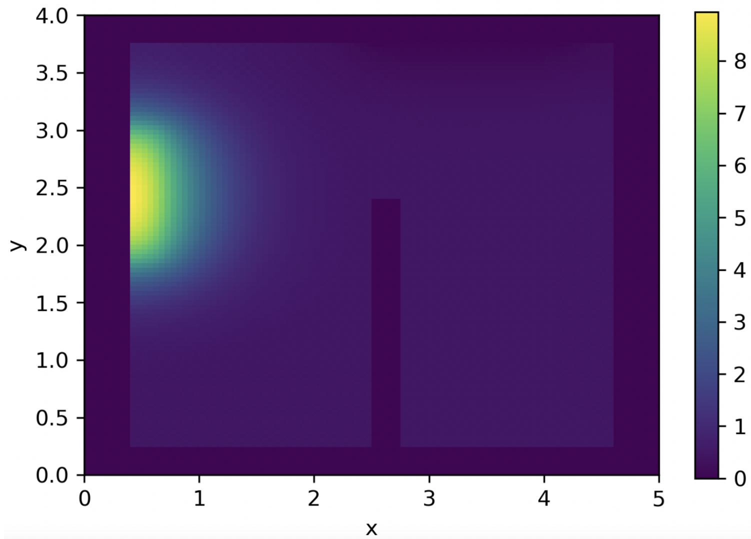

Figure 6: Jacobi method

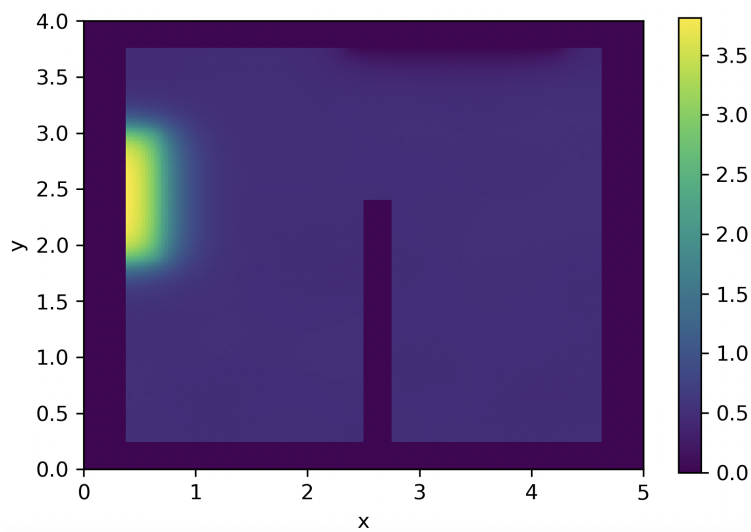

Figure 7: Damped Jacobi Method