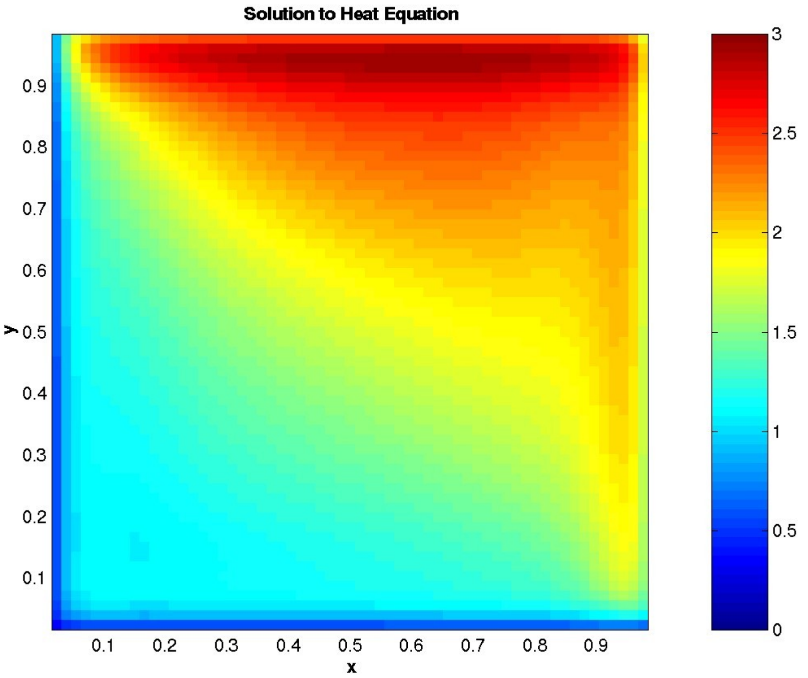
Solving the 2D Heat Equation

Background

$$\frac{\partial\theta}{\partial t} = \kappa(\frac{\partial^2\theta}{\partial x^2} + \frac{\partial^2\theta}{\partial y^2})$$

- Modeling the propagatio of heat on a 2D surface.
- Let $\theta(x, y, t)$ be the temperature at (x, y) at time t, and κ be the conductivity.



Source:
http://people.math.sfu.ca/~mkropins/math922/lectures/heatsolvers.html

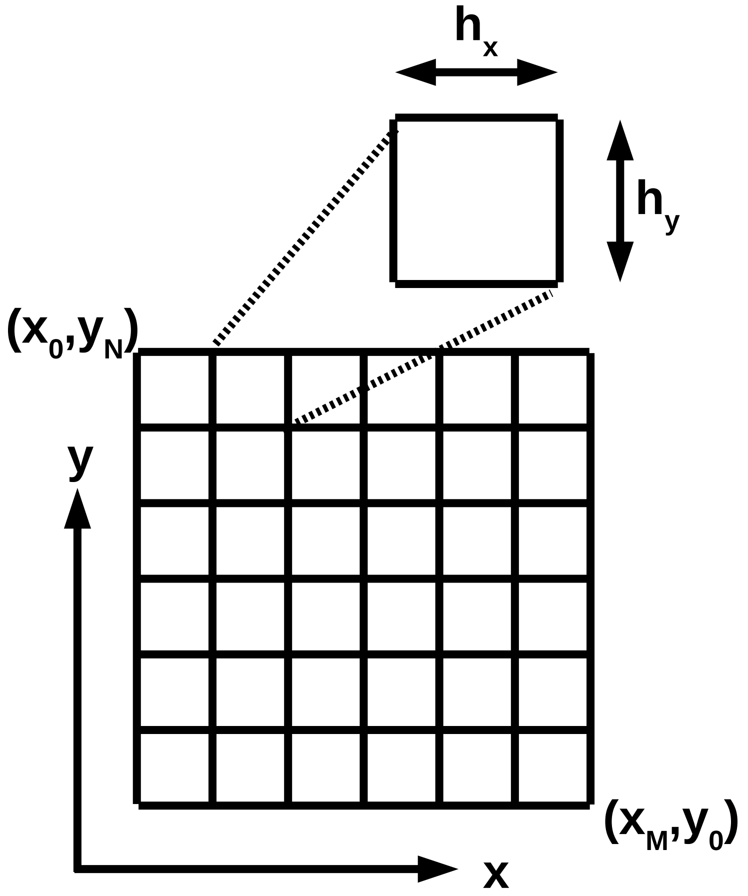
Numerical Method - 1

- Using Taylor series expansion:

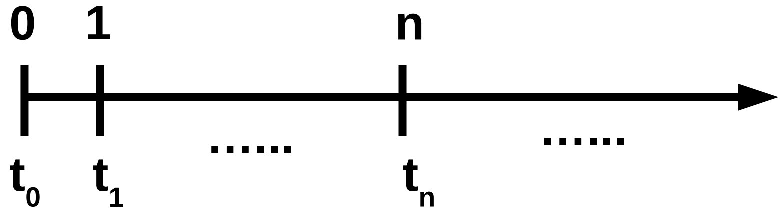
$$\theta(x_m + h_x) = \theta(x_m) + h_x \frac{\partial \theta(x_m)}{\partial x} + \frac{h_x^2}{2}\frac{\partial^2 \theta(x_m)}{\partial x^2} + \frac{h_x^3}{6}\frac{\partial^3 \theta(x_m)}{\partial x^3} + \mathcal{O}(h_x^4)$$

$$\theta(x_m - h_x) = \theta(x_m) - h_x \frac{\partial \theta(x_m)}{\partial x} + \frac{h_x^2}{2}\frac{\partial^2 \theta(x_m)}{\partial x^2} - \frac{h_x^3}{6}\frac{\partial^3 \theta(x_m)}{\partial x^3} + \mathcal{O}(h_x^4)$$

$$\frac{\partial^2 \theta(x_m)}{\partial x^2} = \frac{\theta_{m+1} - 2\theta_m + \theta_{m-1}}{h_x^2} + \mathcal{O}(h_x^2)$$



Numerical Method - 2



$$\frac{\partial^2\theta}{\partial x^2} + \frac{\partial^2\theta}{\partial y^2} = \frac{\theta_{m+1,n} - 2\theta_{m,n} + \theta_{m-1,n}}{h_x^2} + \frac{\theta_{m,n+1} - 2\theta_{m,n} + \theta_{m,n-1}}{h_y^2}$$

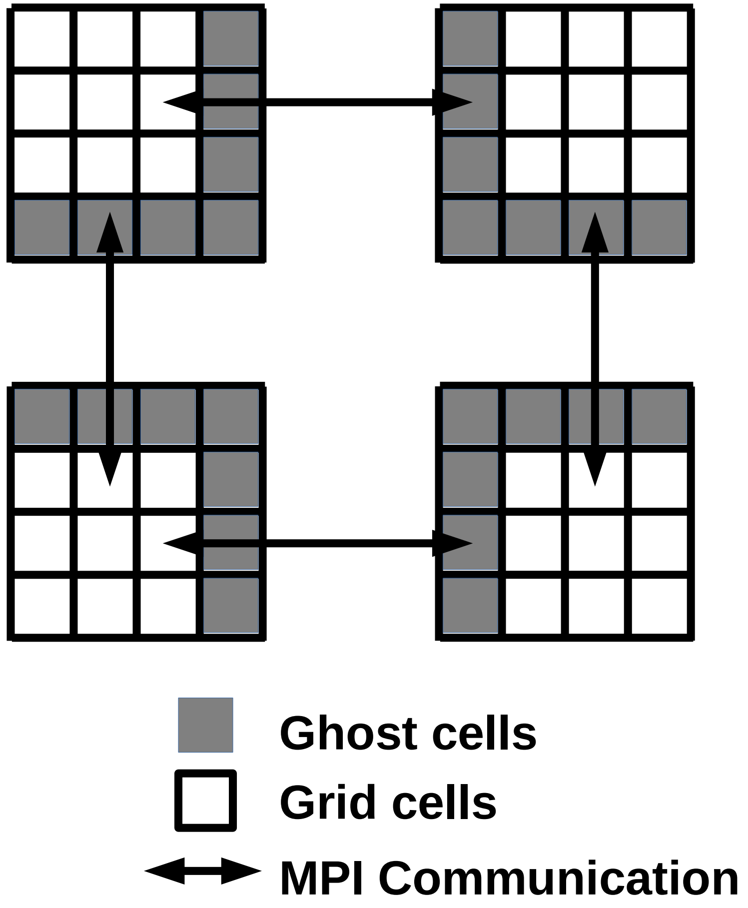
- Taking the time step as Δt, $\frac{\partial\theta_{i,j}^n}{\partial t} = \frac{\theta_{i,j}^{n+1} - \theta_{i,j}^n}{\Delta t}$
- Putting everything together,

$$\boxed{\theta_{i,j}^{n+1} = \theta_{i,j}^n + \kappa\Delta t\left(\frac{\theta_{i+1,j}^n - 2\theta_{i,j}^n + \theta_{i-1,j}^n}{h_x^2} + \frac{\theta_{i,j+1}^n - 2\theta_{i,j}^n + \theta_{i,j-1}^n}{h_y^2}\right)}$$

Note: You need to take $\Delta t \leq \frac{min(h_x^2, h_y^2)}{4\kappa}$ for convergence.

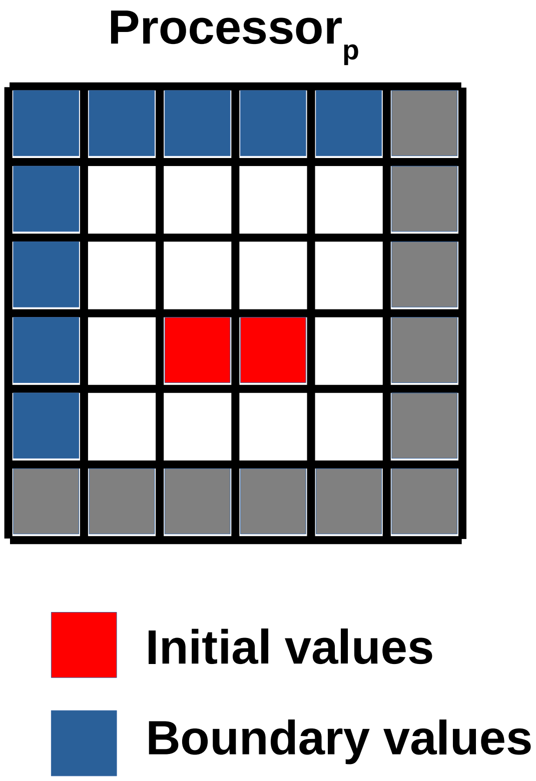
Implementation - 1

- Domain decomposition:
  - Create new communicators (MPI_Comm_split or MPI_Cart_create).
  - Divide problem domain into smaller matrices for each process
  - Add ghost cells for communicating boundary values.

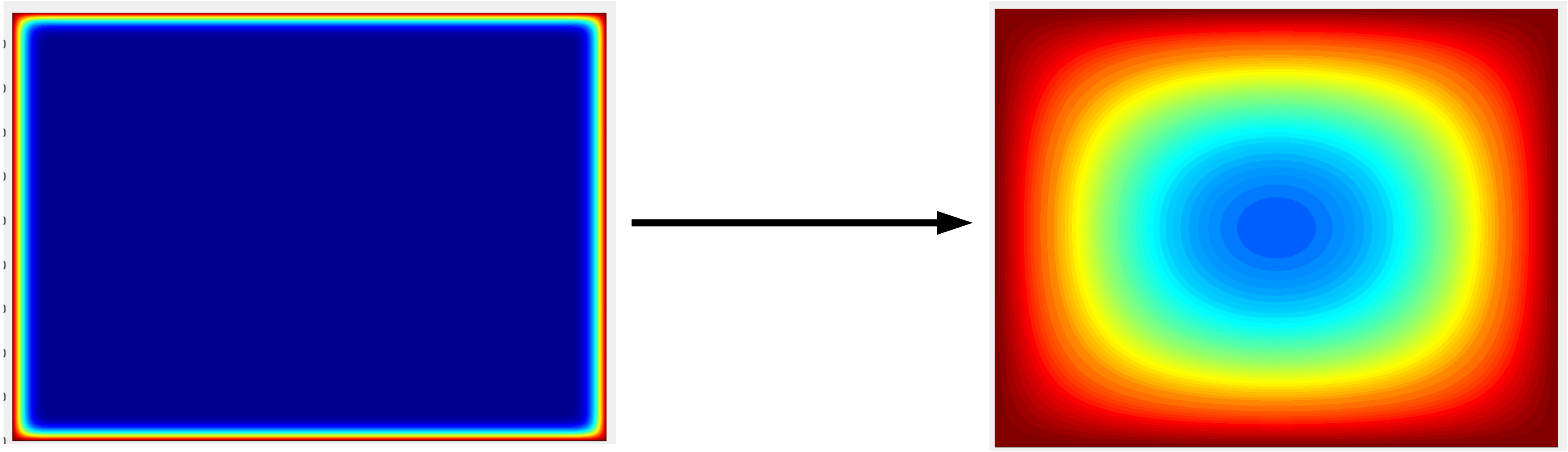


Implementation - 2

- Boundary Conditions:
  - Temperature values constant for the entire simulation.
- Initial Conditions:
  - Non-zero initial temperature values set at t=0.
- Run the simulation for ‘T’ computational cycles



Project



- Set high temperatures on boundary (as a boundary condition) and simulate the diffusion of heat towards the center of the grid.

Report

- Introduction
- Methodology
  - Implementation(s)
  - Validation
- Results
  - Scaling test
- Discussion and Conclusion

- How do you validate your implementation?
- How do you write results to file?
  - MPI I/O?
- Why does your scaling test look like that?

No need to include tables. Use plots instead

References

- For math about solving the 2D equation refer to these slides.
- A Python implementation is given here.
- A Matlab implementation with explanations can be found here.