Parallel Computing for 2D Heat Equation

/* Author: Chih-Cheng Huang

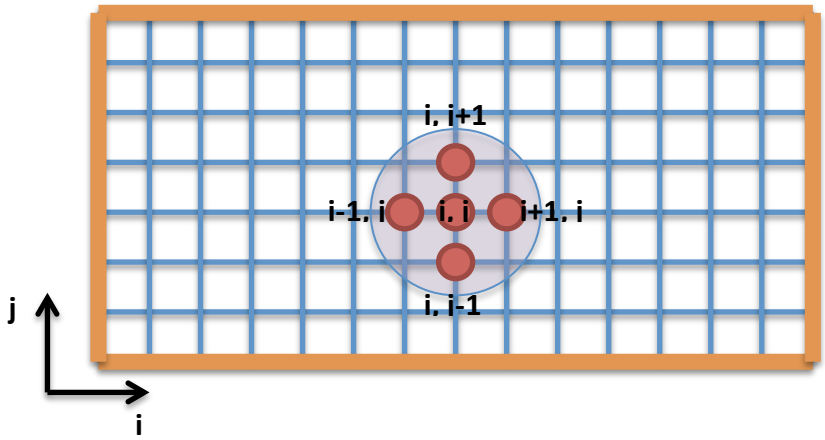
1. This is a brief glance of part of the project I'm working on recently.
2. The code is self-contained in this folder. However, at least one of the MPI open source library should be installed in case you would like to run the code successfully.
3. The command line I used to run the code: $ mpirun -np 4 ./Test
4. The output file: final_temperatures.csv

*/

Implementation:

Discretization – Finite Difference Method

First of all, finite difference method is used to build up the simulation model of the two-dimensional heat equation. The finite difference method is widely used for solving PDE equations. It is derived from the definition of the derivative and Taylor series. Here, we applied the finite difference method to discretize the two-dimensional heat equation and approximate the observation temperature at the given time T_end. We will use these data later to conduct parameter estimation.



There are two kinds of schemes that we considered to implement in finite difference method: 1. Explicit Scheme 2. Implicit Scheme. Both of them use central differences in space, but they're different in working on the time domain. The details are described below,

Explicit Scheme:

$$\frac{v_{i,j}^{k+1} - v_{i,j}^{k}}{\Delta t} = a\left(\frac{v_{i+1,j}^{k} - 2v_{i,j}^{k} + v_{i-1,j}^{k}}{\Delta x^2} + \frac{v_{i+1,j}^{k} - 2v_{i,j}^{k} + v_{i,j-1}^{k}}{\Delta y^2}\right) \quad -(1)$$

To make it clear, we can simplify the equation (1) to the equation below:

$$v^{k+1} = Av^{k}$$

Here, A is the coefficient matrix

Explicit scheme uses forward difference. It calculates the state of a system at a later time from the state at the current time. Also, the explicit scheme is inherently suitable for parallel-implementation. Every grid point v can be updated independently of others. However, A serious drawback of the explicit scheme arose in the implementation process. It is the restriction of the size of time step. Theoretically, the convergence and stability of the scheme has to be satisfied by conducting the following equation:

$$\Delta t \leq \frac{1}{2a}\frac{\Delta x^2 \Delta y^2}{\Delta x^2 + \Delta y^2}$$

Implicit Scheme:

$$\frac{v_{i,j}^{k+1} - v_{i,j}^{k}}{\Delta t} = a\left(\frac{v_{i+1,j}^{k+1} - 2v_{i,j}^{k+1} + v_{i-1,j}^{k+1}}{\Delta x^2} + \frac{v_{i,j+1}^{k+1} - 2v_{i,j}^{k+1} + v_{i,j-1}^{k+1}}{\Delta y^2}\right) \quad -(2)$$

Implicit scheme uses backward difference. It finds a solution by solving an equation involving both the current state and the later one of the system. The implicit method is unconditionally stable so that it doesn't have to satisfy the restriction equation that we should obey in the explicit scheme. However, certain numerical method, such as the Jacobi iteration or the Gauss-Seidel method, has to be conducted to solve the linear system below.

We simplify the implicit scheme from equation (2):

$$Av^{k+1} = v^{k} \quad -(3)$$

Here, A is the coefficient matrix.

It's clear that the implicit scheme requires an extra computation for solving the inverse problem above. Basically it costs more time for each time steps. However, the implicit scheme is usually used because many real problems are stiff problems where certain numerical methods, such as the explicit scheme, are numerically unstable for solving the problems, unless the time step size is taken to be impractically small. For such reasons, to achieve given accuracy, it takes much less computational time by using the implicit scheme with larger time steps, even though the computation time of the inverse problem is taken into account. That's why we used the implicit scheme with Jacobi iteration instead of explicit scheme in our project.

Parallelization

Message Passing Interface (MPI)

MPI was used in this project to achieve parallel computing. It provides a language-independent communication protocol for programming parallel computers. It is a message-passing application programmer interface, together with protocol and semantic specifications for how its features must behave in any implementation. With the help of MPI, we developed our code in a way we called it "Master/ Worker Scheme". The details are described below:

Master/Worker Scheme

The basic idea of the Master/Worker Scheme is as follows:

1. Assign and create a number of processors.
2. Assign the Master processor to initialize the two-dimensional heat equation model, including:
   - Initial condition - 273 Kelvin everywhere except for the boundaries
   - Boundary condition - 300 Kelvin along the boundaries
3. Separate the data in the Master processor into several sub grids and send them individually to the Worker processors to work on it.

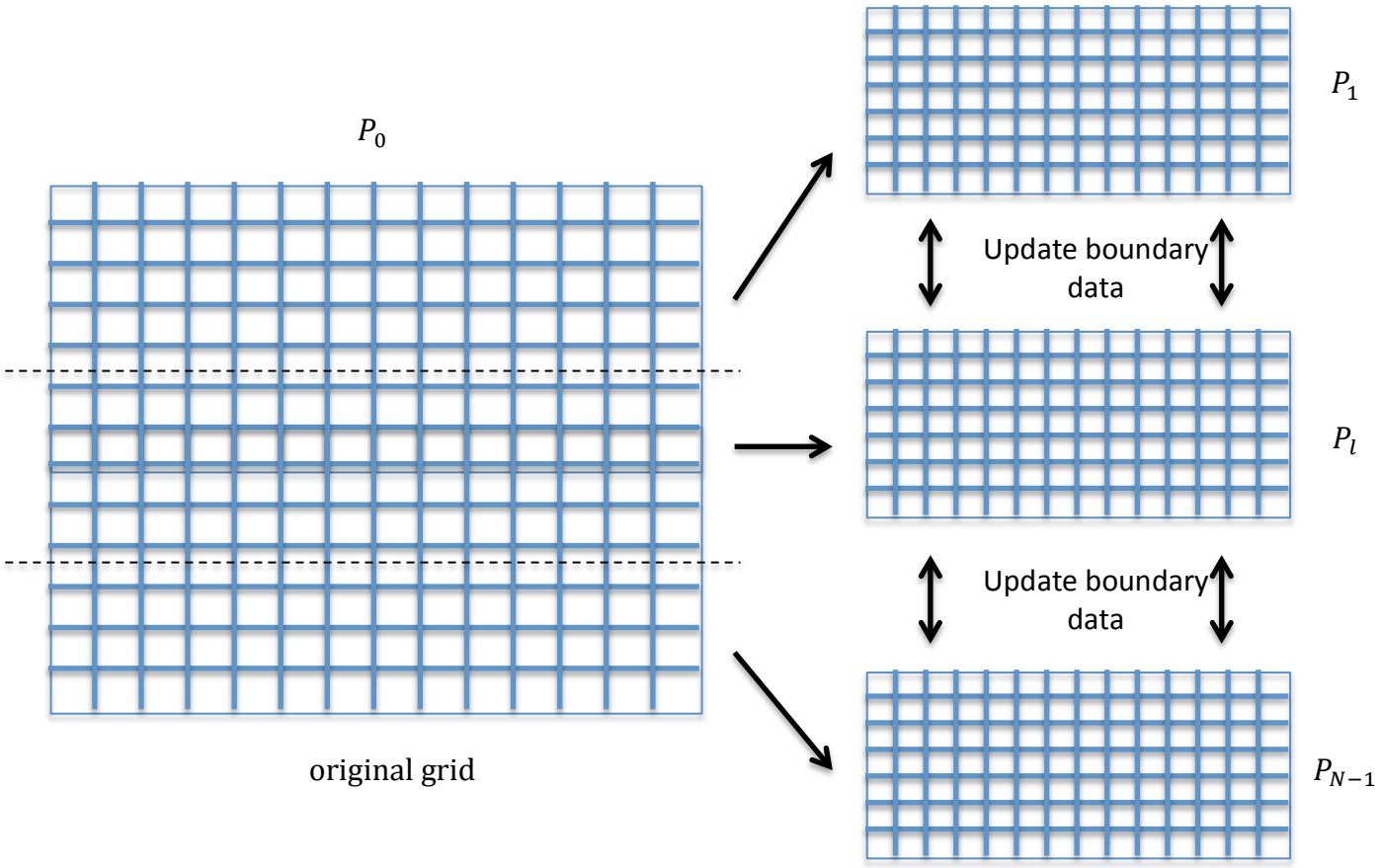


Figure. Implicit scheme processed in parallel

4. Solve equation (3) for each time step in the Worker processors. The heat source is present at this step as a constant for calculation. Besides, during the calculation, the Worker processors have to share the data of the adjacent boundaries for each Jacobi iteration to update the data and solve the inverse problem.
5. Send the data from the Worker processors back to the Master processor to get the final result – the final temperature distribution at the assigned time duration T_end.

Based on the steps described above, we can develop the pseudo code as follows:

```
set initial and boundary conditions on uold(i,j), for all i,j;
if ( 0==matsterID ) send chunks of data in uold to the Worker processors;
if ( 0!=matsterID ) recv dara from the Master processors
for n=0; n<nt; n=n+1; do          // Compute the results for each time step
        JacobiIteration();        // Implement Jacobi Iteration
end
if ( 0!=matsterID ) then
        sent uold data from each Workers back to Master processor;
else
        recv uold from processors 1 to np-1 and place in uold in Master processor;
end if
```

Simulation Result

Simulation Result

The simulation result for the above code at T_end equal to 100 seconds is shown as the figure below:

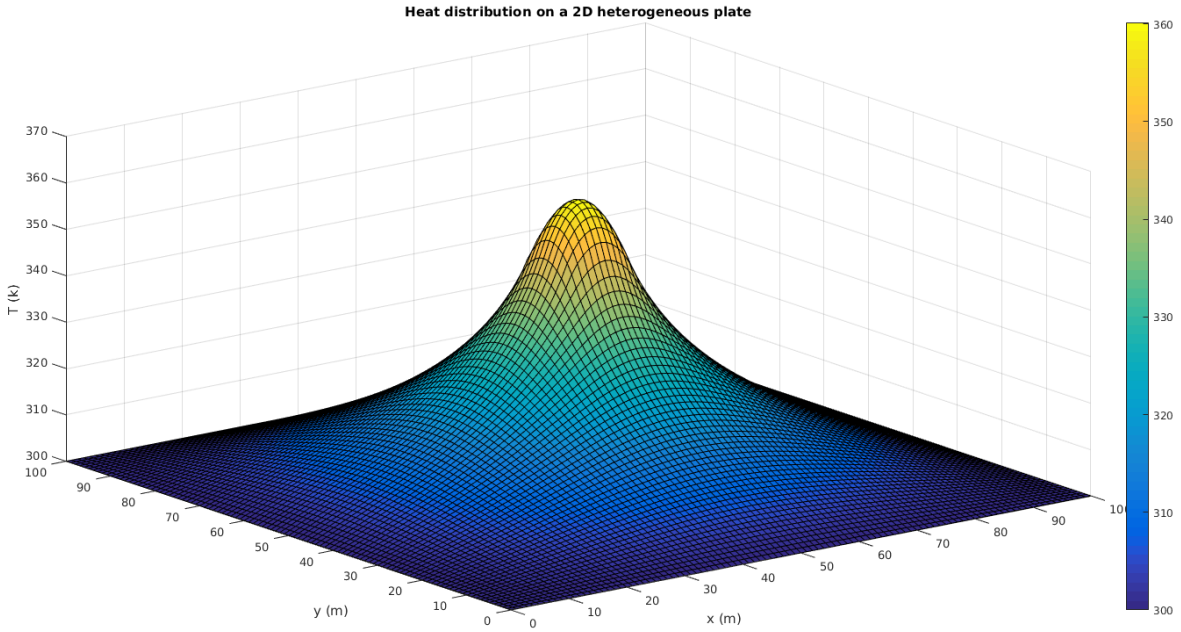


Figure. Simulation results for the observation temperature at T_end equal to 100 seconds

Here, we can observe high temperature distribution in the middle area where the heat source is given and then the temperature drops steeply to constant boundaries.