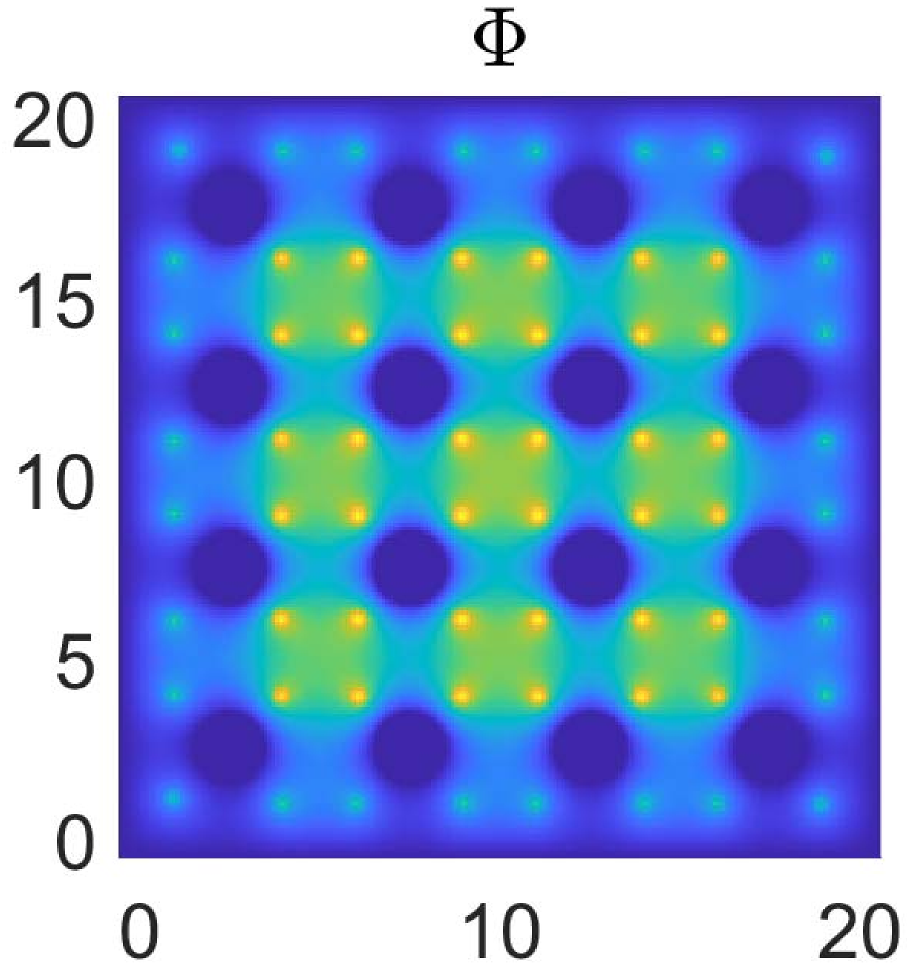
Overlapping Domain Decomposition Finite Element Method (FE-DDM) of Electrostatic Problem

TEAM 10

Junda Feng
(jundaf2@illinois.edu)

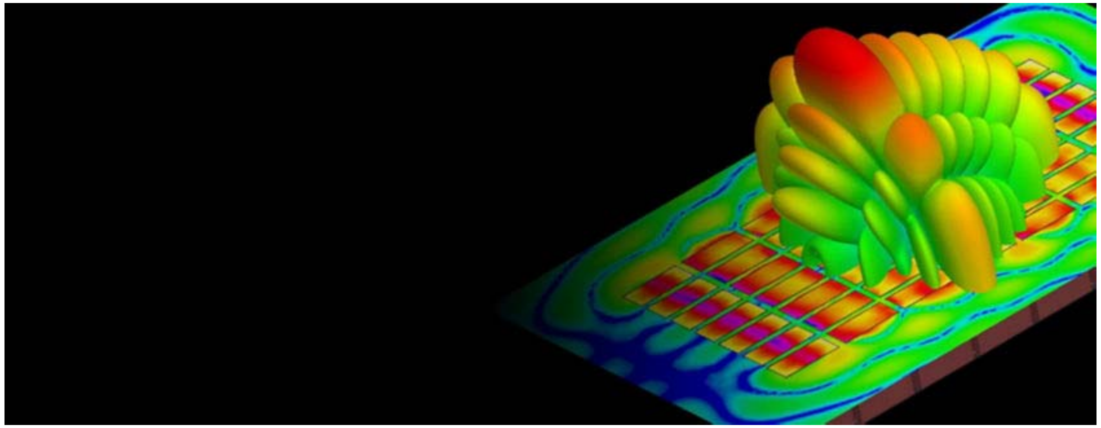
2021/5/8



What is FEM?

The finite element method (FEM) is a widely used method for numerically solving partial differential equations (PDEs) arising in engineering and mathematical modeling.

-- Wikipedia



https://www.ansys.com/products/electronics/ansys-hfss

Poisson Equation

The governing PDE of the electrostatic problem is Poisson equation.

$$-\nabla \cdot (\varepsilon(x, y)\nabla\Phi) = \rho_e(x, y)$$

FEM Formulation

2021/5/8

$$-\int_\Omega \omega_i \left[\nabla \cdot (\varepsilon(x,y)\nabla\Phi)\right] d\Omega = \int_\Omega \omega_i \rho_e(x,y) d\Omega \tag{3}$$

$$\int_\Omega \varepsilon(x,y)\nabla\omega_i \cdot \nabla\Phi d\Omega = \int_\Omega \omega_i \rho_e(x,y) d\Omega + \oint_{\Gamma_D} \hat{n} \cdot (\varepsilon(x,y)\nabla\Phi)\omega_i d\Gamma_D \tag{4}$$

$$\sum_{j=1}^{N} \phi_j \int_\Omega \varepsilon(x,y)\nabla N_i \cdot \nabla N_j d\Omega = \int_\Omega N_i \rho_e(x,y) d\Omega - \sum_{j=1}^{N_D} \phi_j^D \int_\Omega \varepsilon(x,y)\nabla N_i \cdot \nabla N_j^D d\Omega \tag{5}$$

$$\sum_{j=1}^{N} K_{ij}\phi_j = b_i, \qquad i = 1, \ldots, N \tag{6}$$

$$K_{ij} = \begin{cases} 1 & \text{if} \quad i = j \quad \text{and node} \quad j \text{ is on } \Gamma_D \\ \int_\Omega \varepsilon(x,y)\nabla N_i \cdot \nabla N_j d\Omega & \text{if} \quad \phi_j \text{ is unknown} \end{cases} \tag{7}$$

$$b_i = \begin{cases} \int_\Omega N_i \rho_e(x,y) d\Omega - \sum_{j=1}^{N_D} \phi_j^D \int_\Omega \varepsilon(x,y)\nabla N_i \cdot \nabla N_j^D d\Omega & \text{if } \phi_i \text{is unknown} \\ \phi_i^D - \sum_{j=1, j\neq i}^{N_D} \phi_j^D \int_\Omega \varepsilon(x,y)\nabla N_i \cdot \nabla N_j^D d\Omega & \text{if } \phi_i \text{is prescribed (the potential of ground is zero)} \end{cases} \tag{8}$$

Basis function

Solution Process

(Simplified)

- Use the equations to formulate the linear sparse system.
- Solve it using conjugated gradient method for the coefficients of the corresponding basis functions.

$$\boldsymbol{K}\boldsymbol{\phi} = \boldsymbol{b}$$

Conjugated Gradient Method

Pseudo-code.

Many parallel versions studied in detail.

Simply using OpenMP.

Algorithm 1 CG algorithm

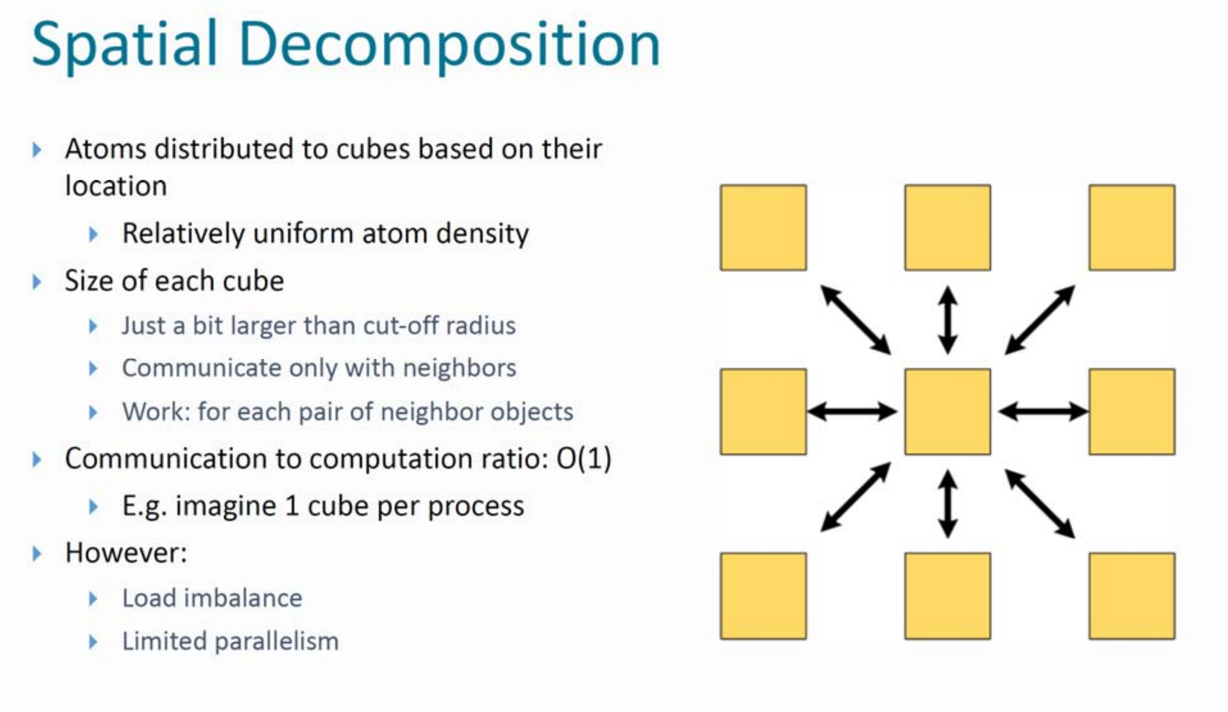
Initialize $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{K}\phi_0$ and $\boldsymbol{p}_0 = \boldsymbol{r}_0$
for $k = 1, 2, \ldots$ **do**
  $\rho_k = \|\boldsymbol{r}_k, \boldsymbol{r}_k\|_2$
  $\boldsymbol{q}_k = \boldsymbol{K}\boldsymbol{p}_k$
  $\alpha_k = \frac{\rho_k}{\|\boldsymbol{p}_k, \boldsymbol{q}_k\|_2^2}$
  $\phi_{k+1} = \phi_k + \alpha_i \boldsymbol{p}_i$
  $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_i \boldsymbol{q}_i$
  $\boldsymbol{r}_{k+1} = \boldsymbol{M}^{-1}\boldsymbol{r}_{k+1}$
  $\rho_{k+1} = \|\boldsymbol{r}_{k+1}, \boldsymbol{r}_{k+1}\|_2^2$
  $\beta_{k+1} = \frac{\rho_{k+1}}{\rho_k}$
  $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_{k+1}\boldsymbol{p}_k$
  Check convergence status
end for

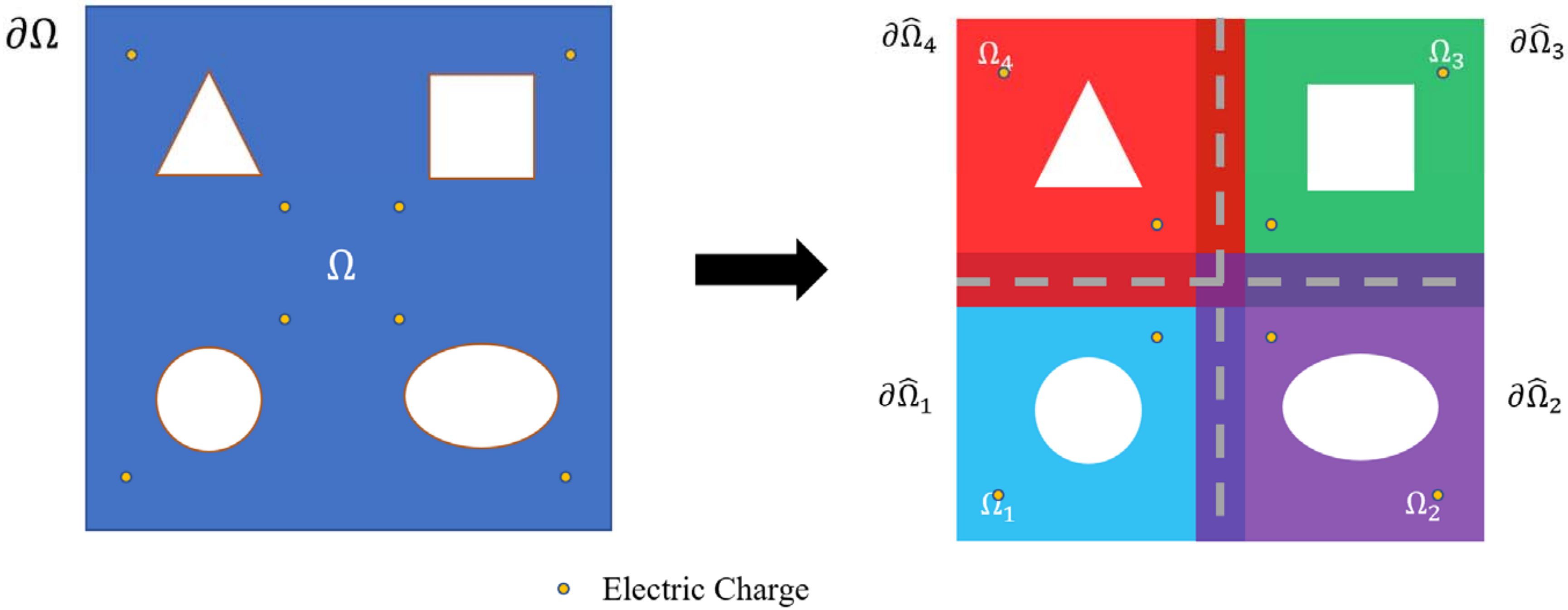
- vector update (SAXPY)
- inner product
- matrix-vector multiplication

Domain Decomposition

Some thing like this picture in the course slides.

Using MPI between ranks.

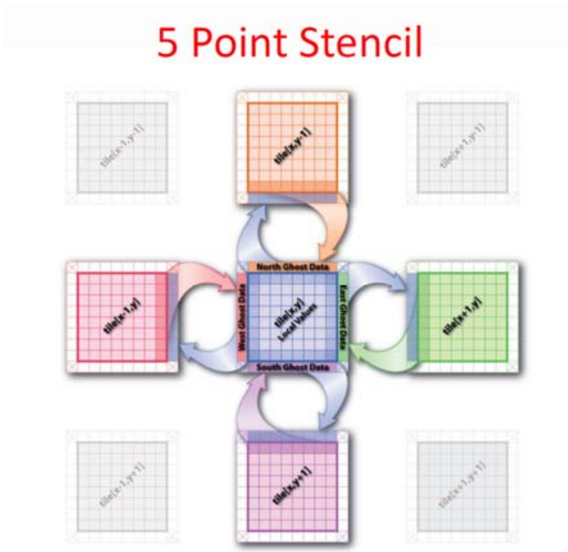


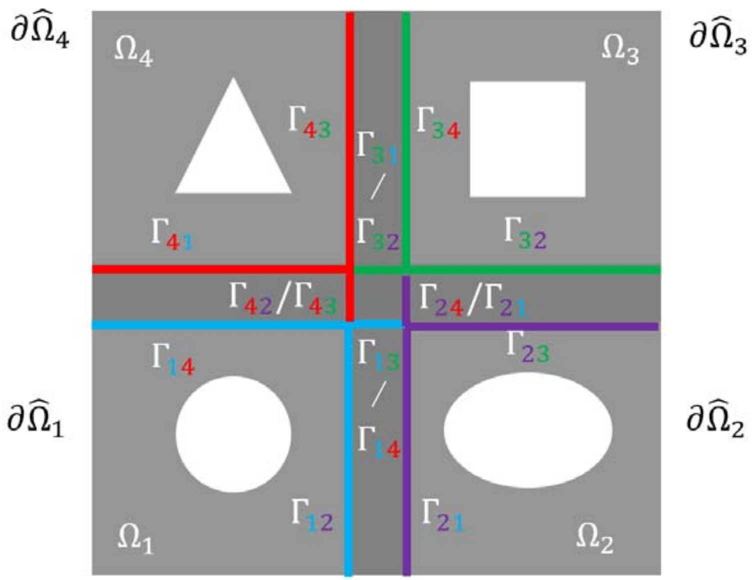


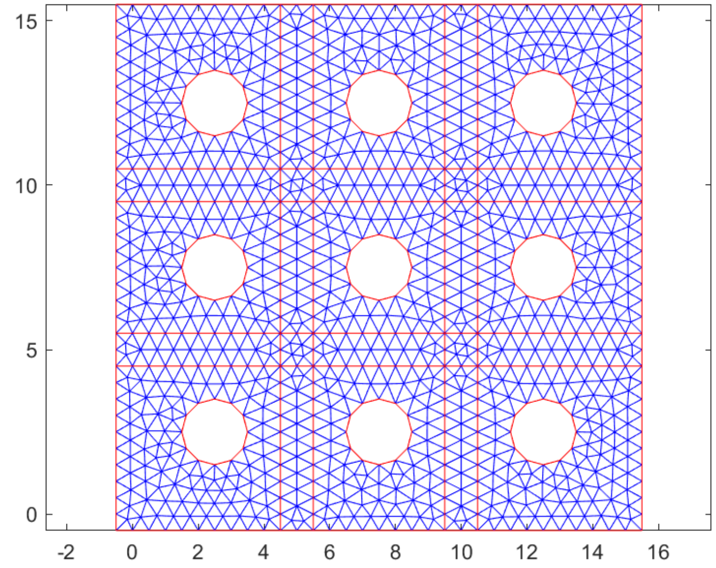
Domain Decomposition

To be more specific (as an example).

Divide the original domain into 2x2 sub-region.



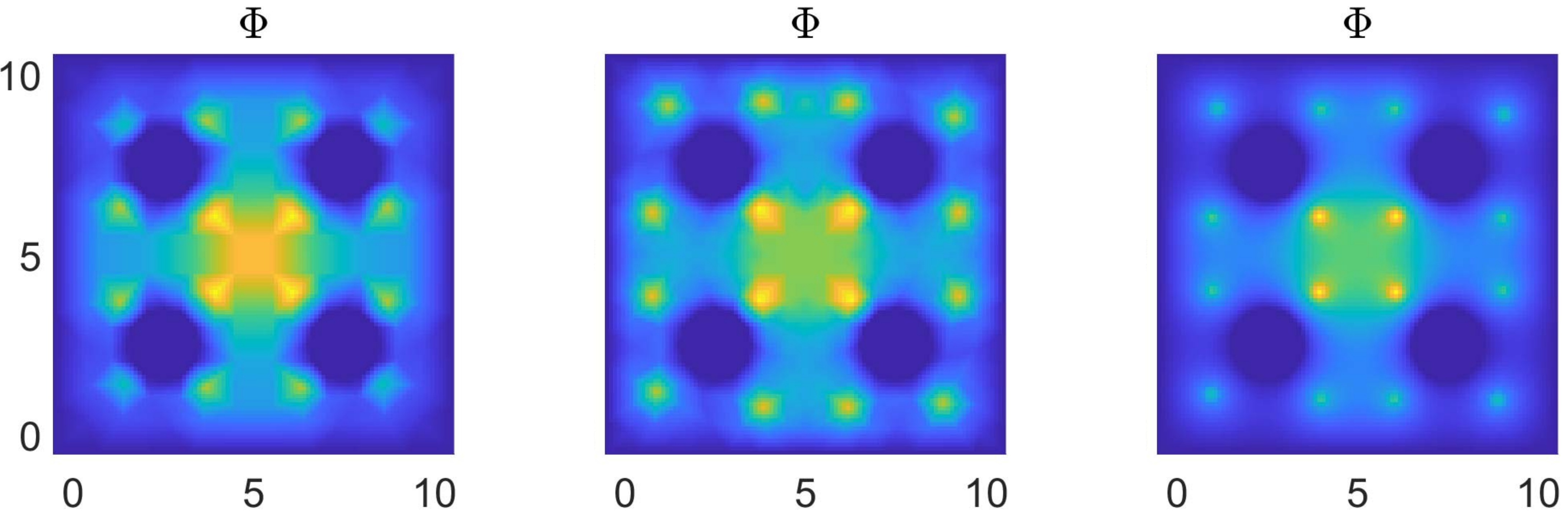




Artificial boundary (ghost boundary)

But this time the ghost is in the interior region of its nearby sub-regions.

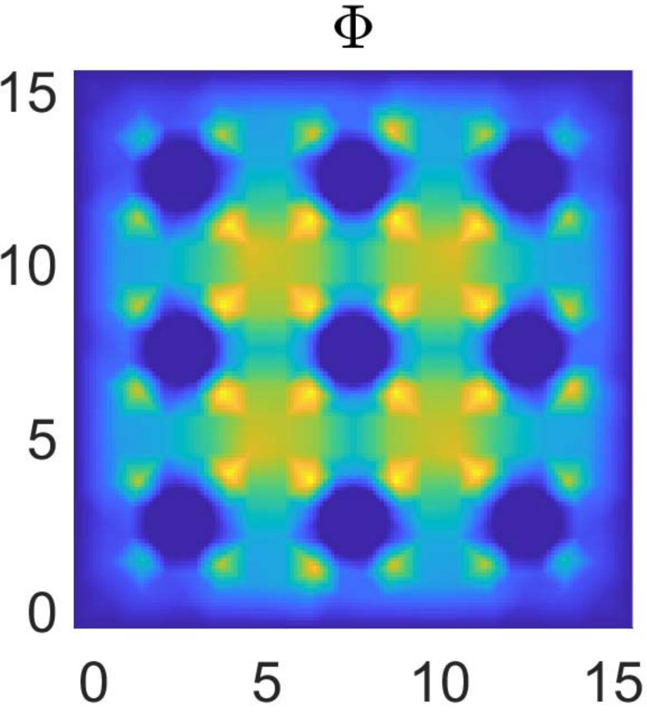
If some sub-region is not on the outer boundary, it may have 8 adjacent sub-regions.

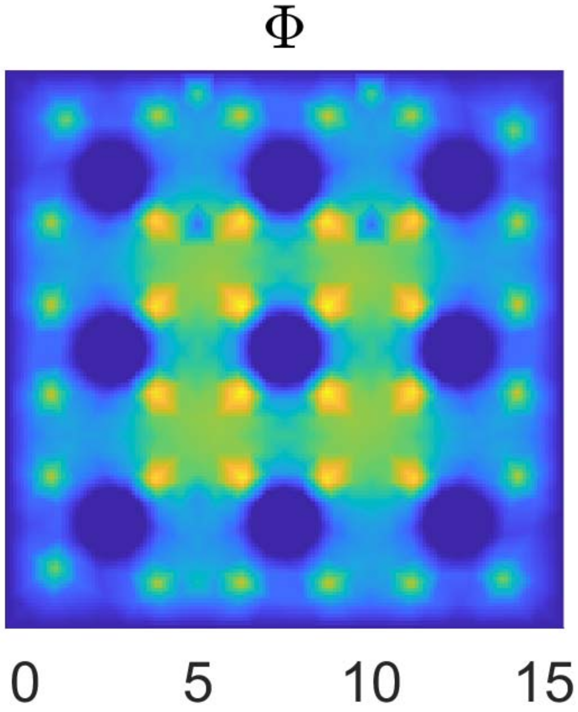


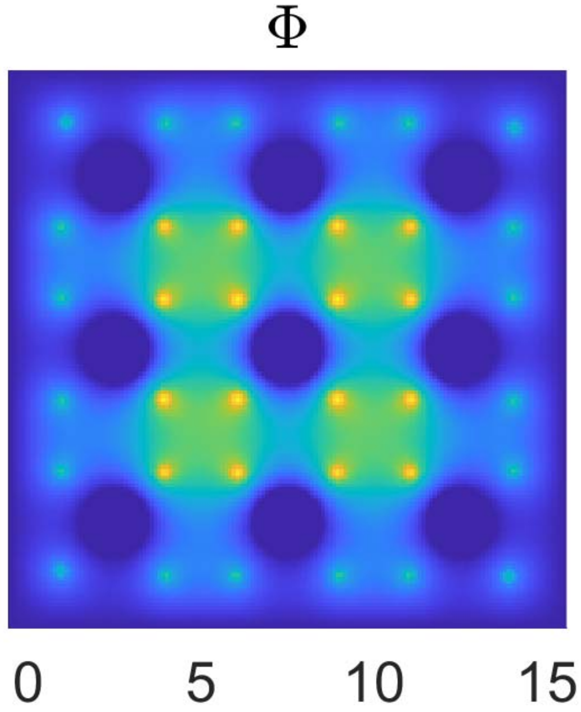
Electric potential visualization

2 by 2

2021/5/8

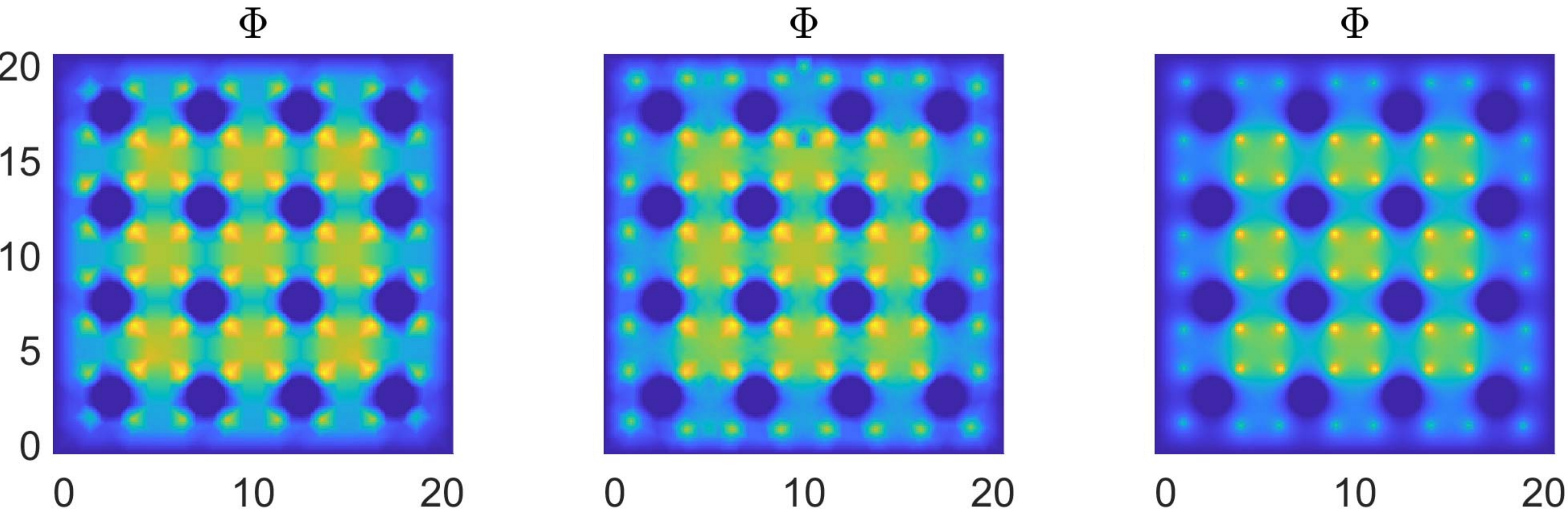






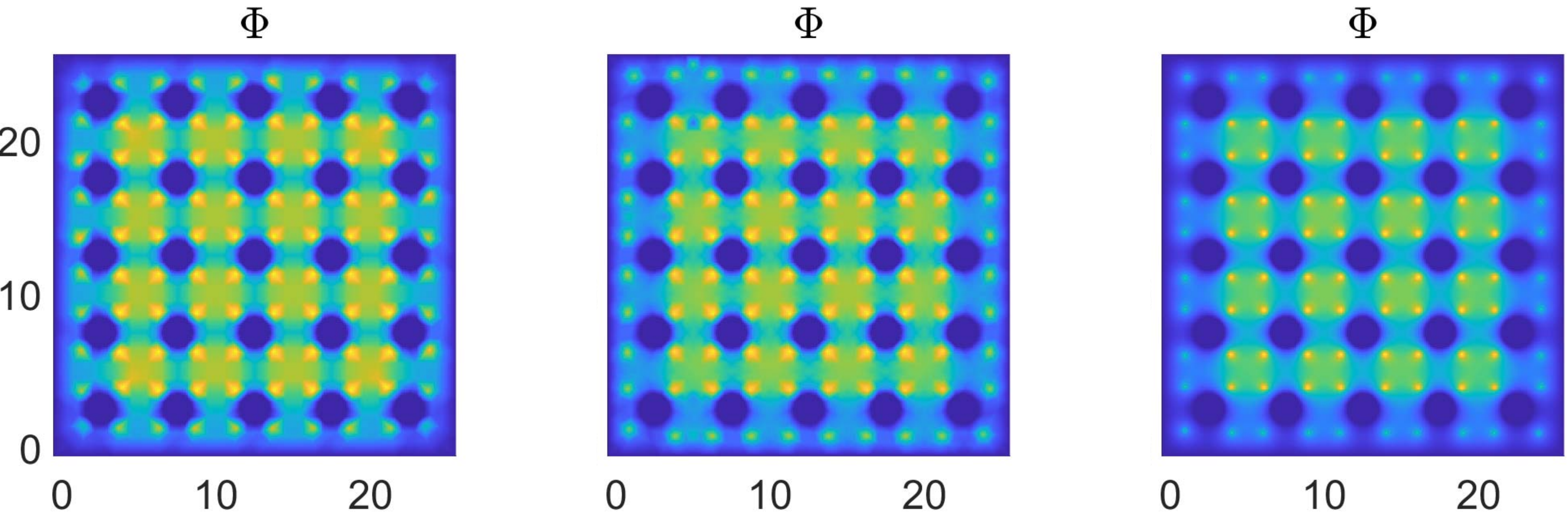
Electric potential visualization

3 by 3



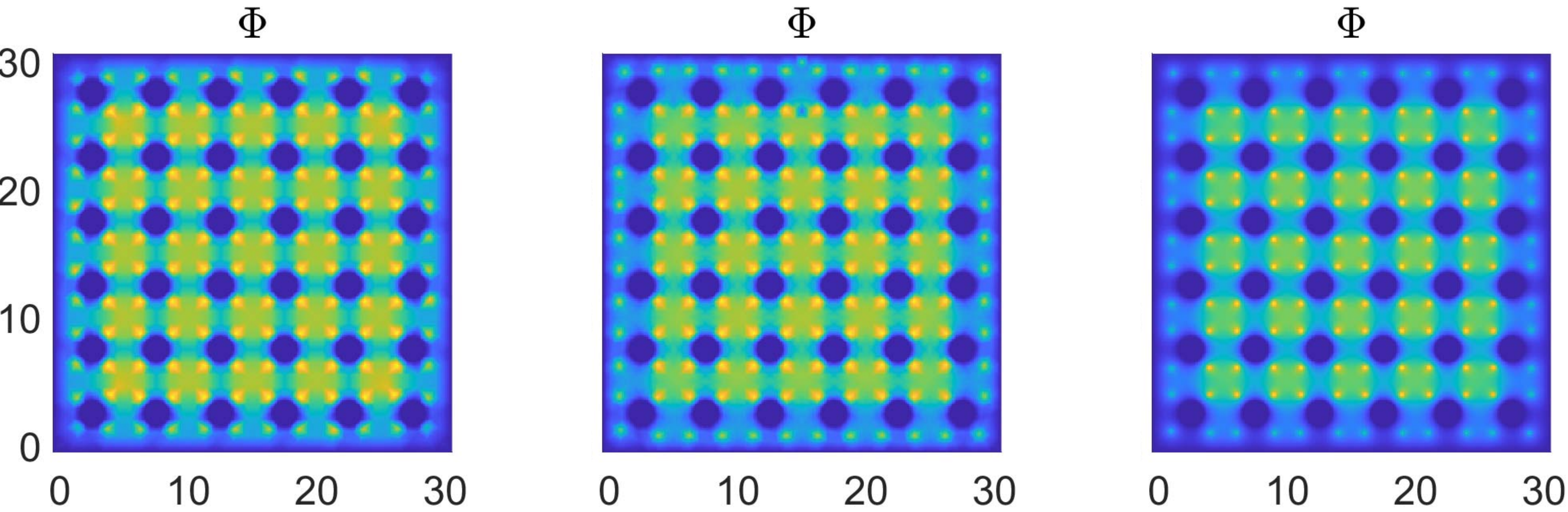
Electric potential visualization

4 by 4



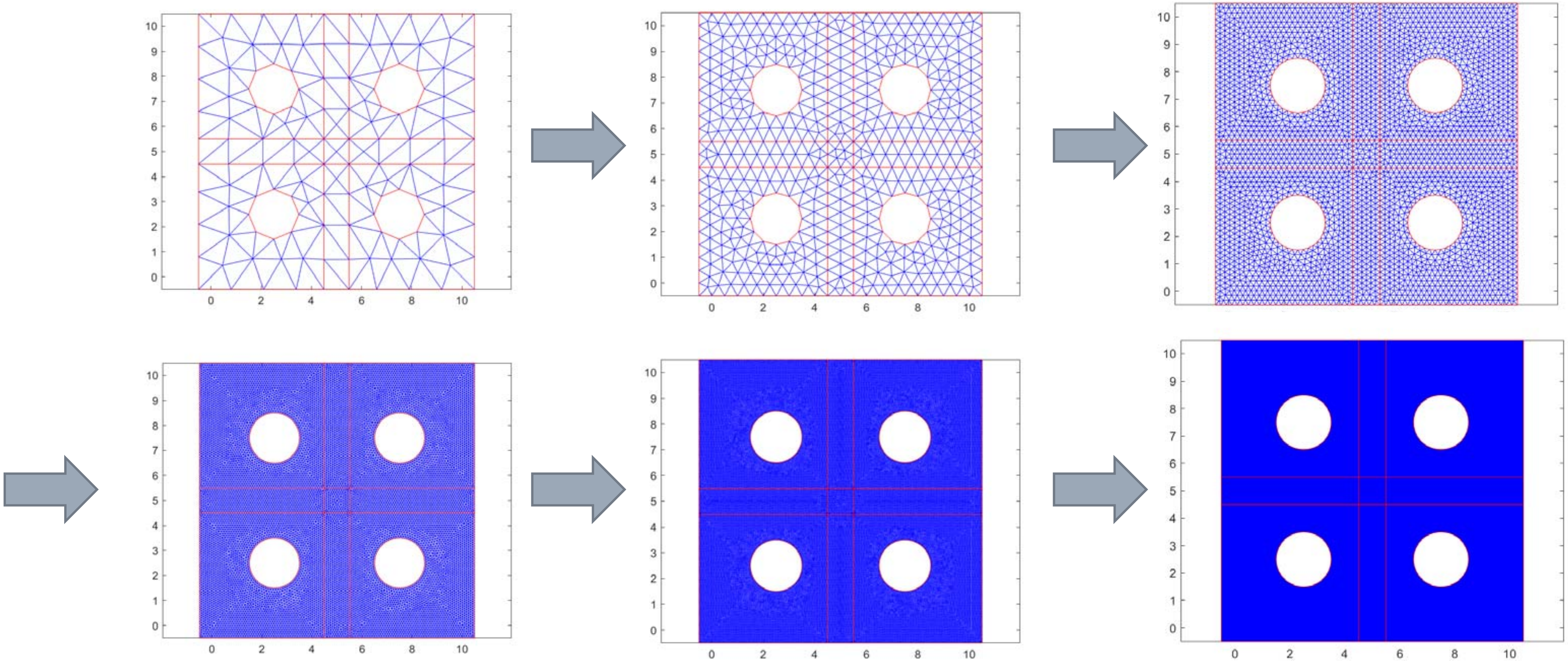
Electric potential visualization

5 by 5



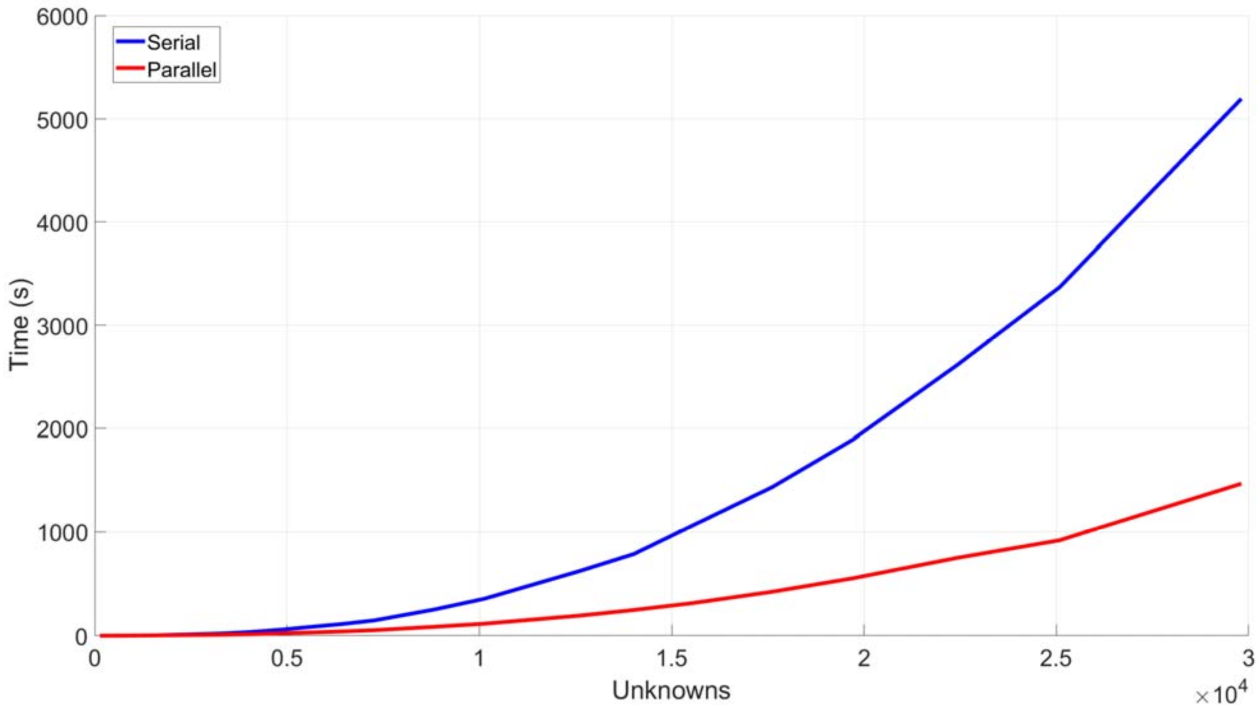
Electric potential visualization

6 by 6



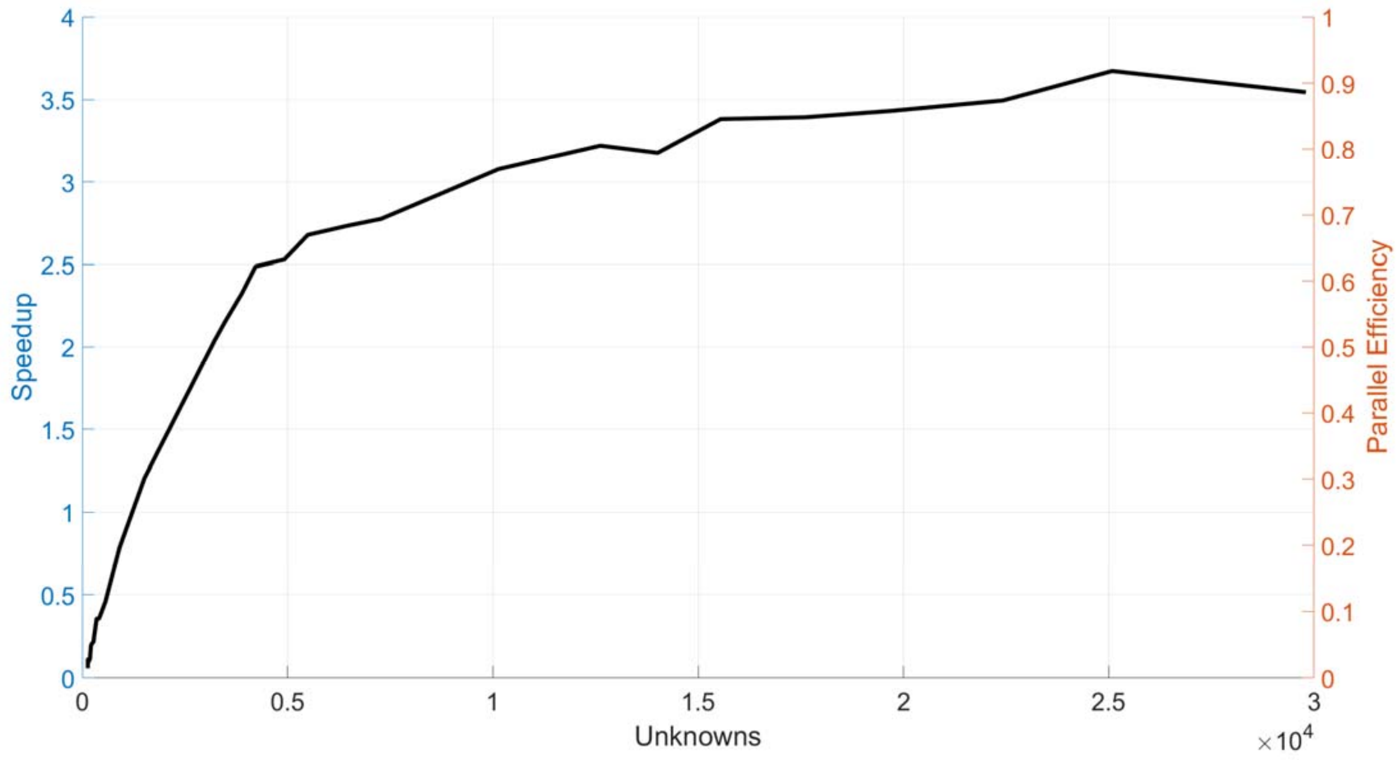
Benchmarking: Increase the number of unknowns

4 by 4



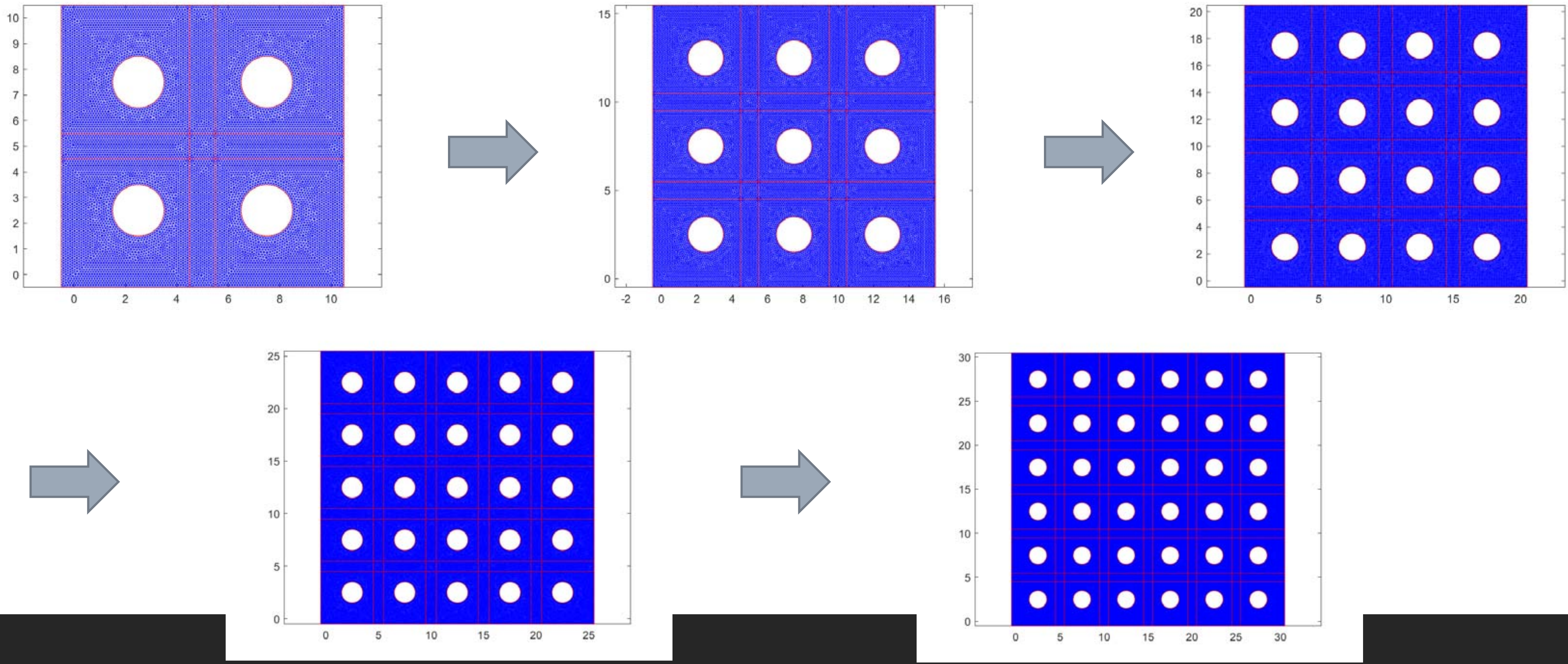
Comparison of time consumption

Serial version vs Parallel version



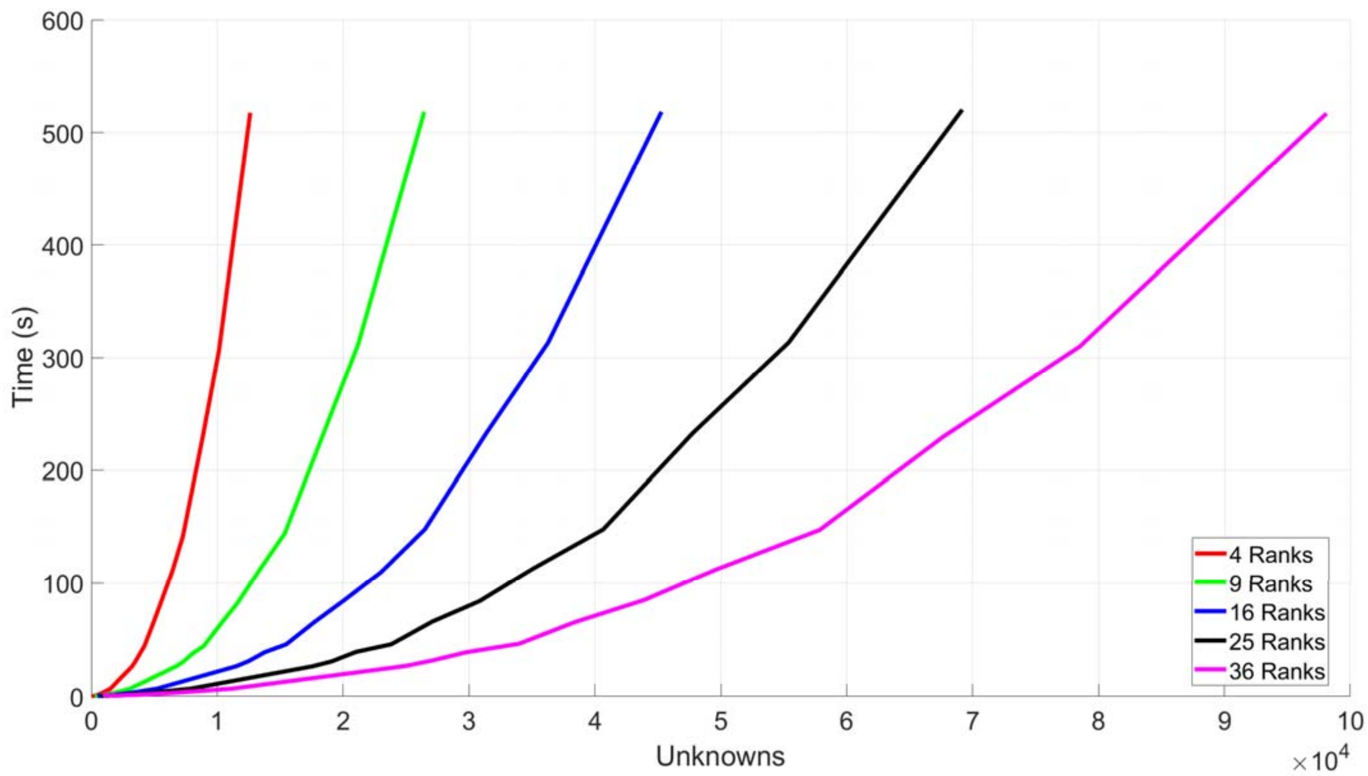
Speed up and parallel efficient

According to the definition.



Benchmarking: Scaling

Increase decomposition, more sub-regions

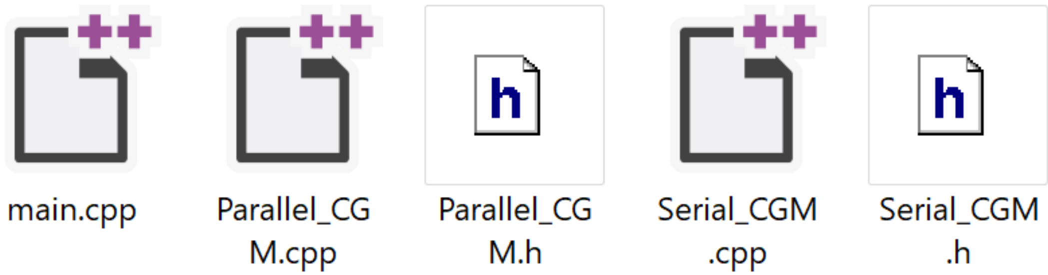


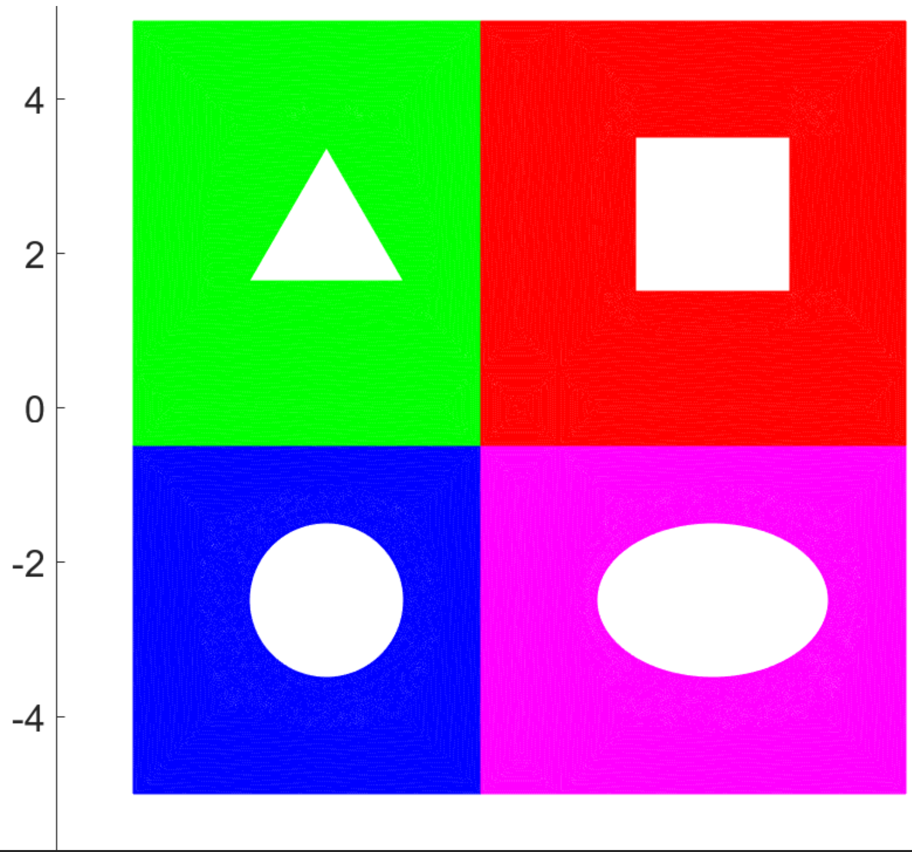
Scaling Result (Weak & Strong)

Increasing number of ranks (different color)

Code Demonstration

Let me open the VS Code if needed.





Q&A Session

2021/5/8