二维 Poisson 方程的并行求解算法

主要内容

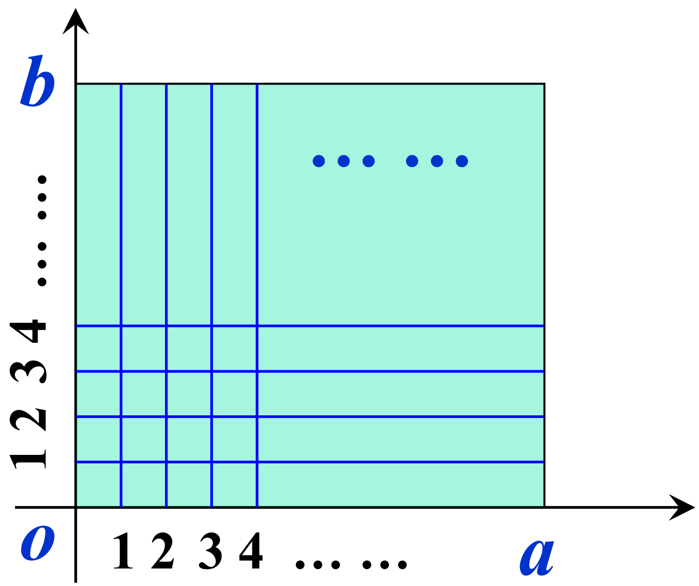
- 二维 Poisson 方程的差分离散
- 差分方程的 Jacobi 算法
- 串行算法
- 并行算法
- 红黑排序的 GS 算法

二维Poisson方程

二维 Poisson 方程

$$\begin{cases} -\Delta u(x, y) = f(x, y), & (x, y) \in \Omega \\ u(x, y) = g(x, y), & (x, y) \in \partial\Omega \end{cases}$$

其中 $\Omega=(0, a) \times (0, b)$，$\partial\Omega$ 为边界



五点差分离散

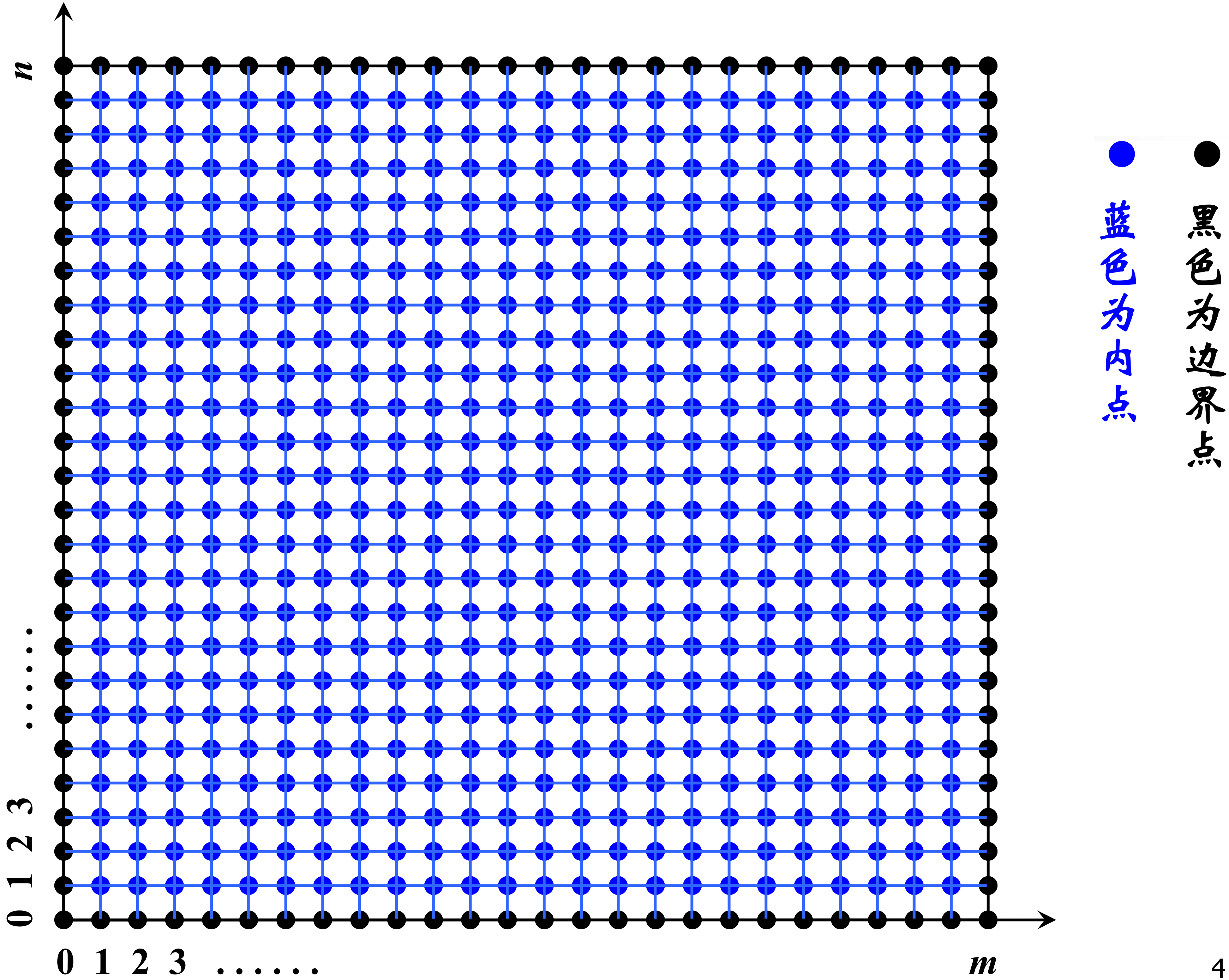
- x-方向和 y-方向的步长分别取为

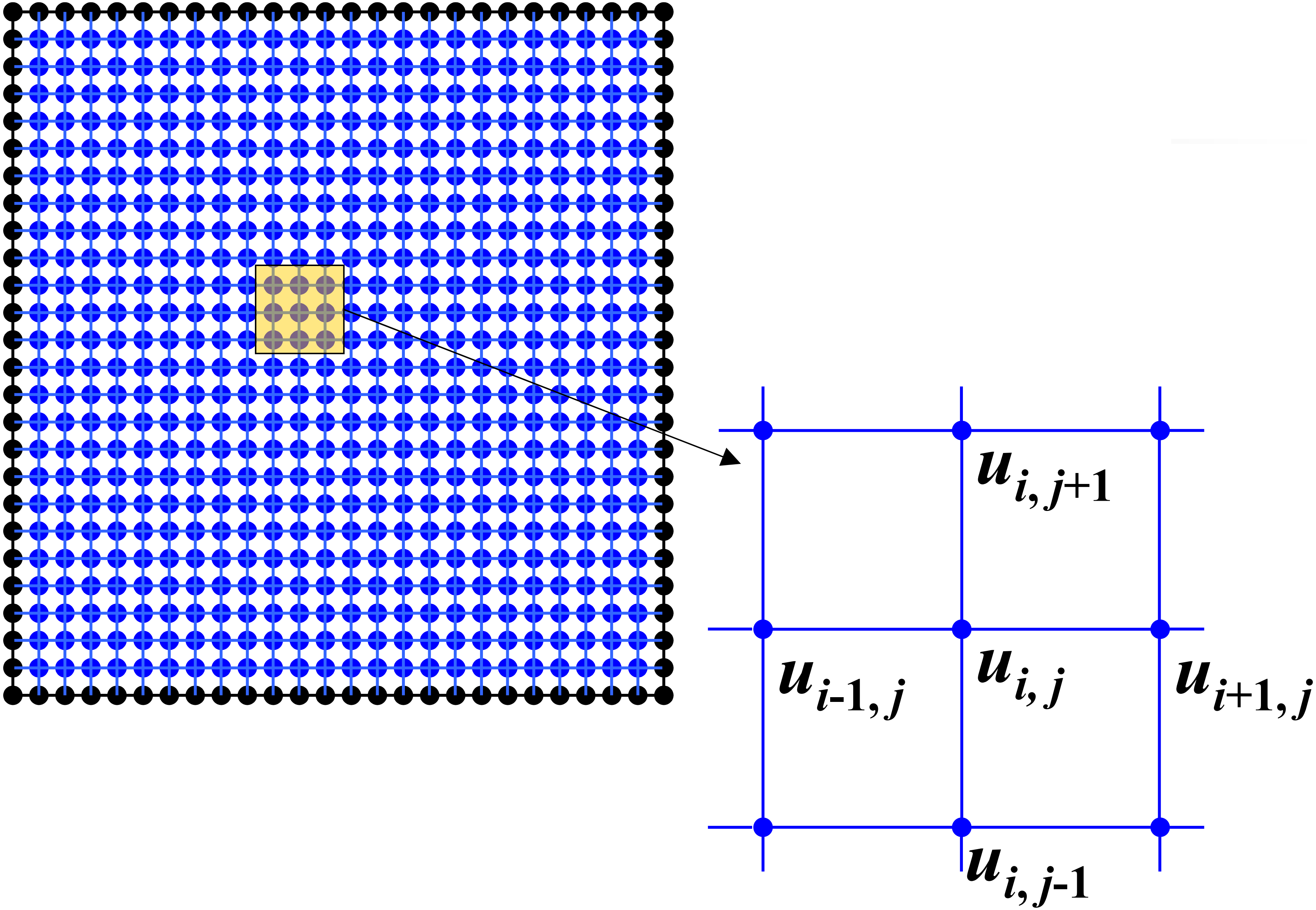
$$\boxed{h_x = \frac{a}{m}, \quad h_y = \frac{b}{n}}$$

- 网格点：(x_i, y_j)，其中

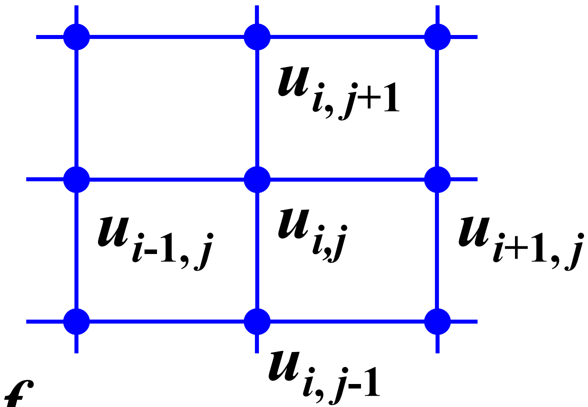
$$x_i = i*h_x, \quad y_j = j*h_y, \; i = 0, 1, \dots, m, \; j = 0, 1, \dots, n$$

- u 在 (x_i, y_j) 点的近似值记为 $u_{i,j}$


n
0 1 2 3 ......
0 1 2 3 ......
m
蓝色为内点
黑色为边界点


$u_{i,j+1}$
$u_{i-1,j}$
$u_{i,j}$
$u_{i+1,j}$
$u_{i,j-1}$

二维Poisson方程



- 离散后的差分方程为

$$\frac{2u_{ij}-u_{i+1,j}-u_{i-1,j}}{h_x^2}+\frac{2u_{ij}-u_{i,j+1}-u_{i,j-1}}{h_y^2}=f_{ij}$$

$$\boxed{i=1,\ldots,m-1,\ j=1,\ldots,n-1}$$

- 边界条件：

$$\boxed{u_{i0}=g_{i0},\quad u_{in}=g_{in},\quad u_{0j}=g_{0j},\quad u_{mj}=g_{mj}}$$

$$\boxed{f_{ij}=f(x_i,y_j),\quad g_{ij}=g(x_i,y_j)}$$

- 整理后可得

$$u_{ij}-d_x(u_{i+1,j}+u_{i-1,j})-d_y(u_{i,j+1}+u_{i,j-1})=d\cdot f_{ij}$$

其中 $\boxed{d=\frac{h_x^2h_y^2}{2(h_x^2+h_y^2)},\quad d_x=\frac{d}{h_x^2},\quad d_y=\frac{d}{h_y^2}}$

Jacobi 迭代

- 求解该差分方程组的 Jacobi 迭代格式为

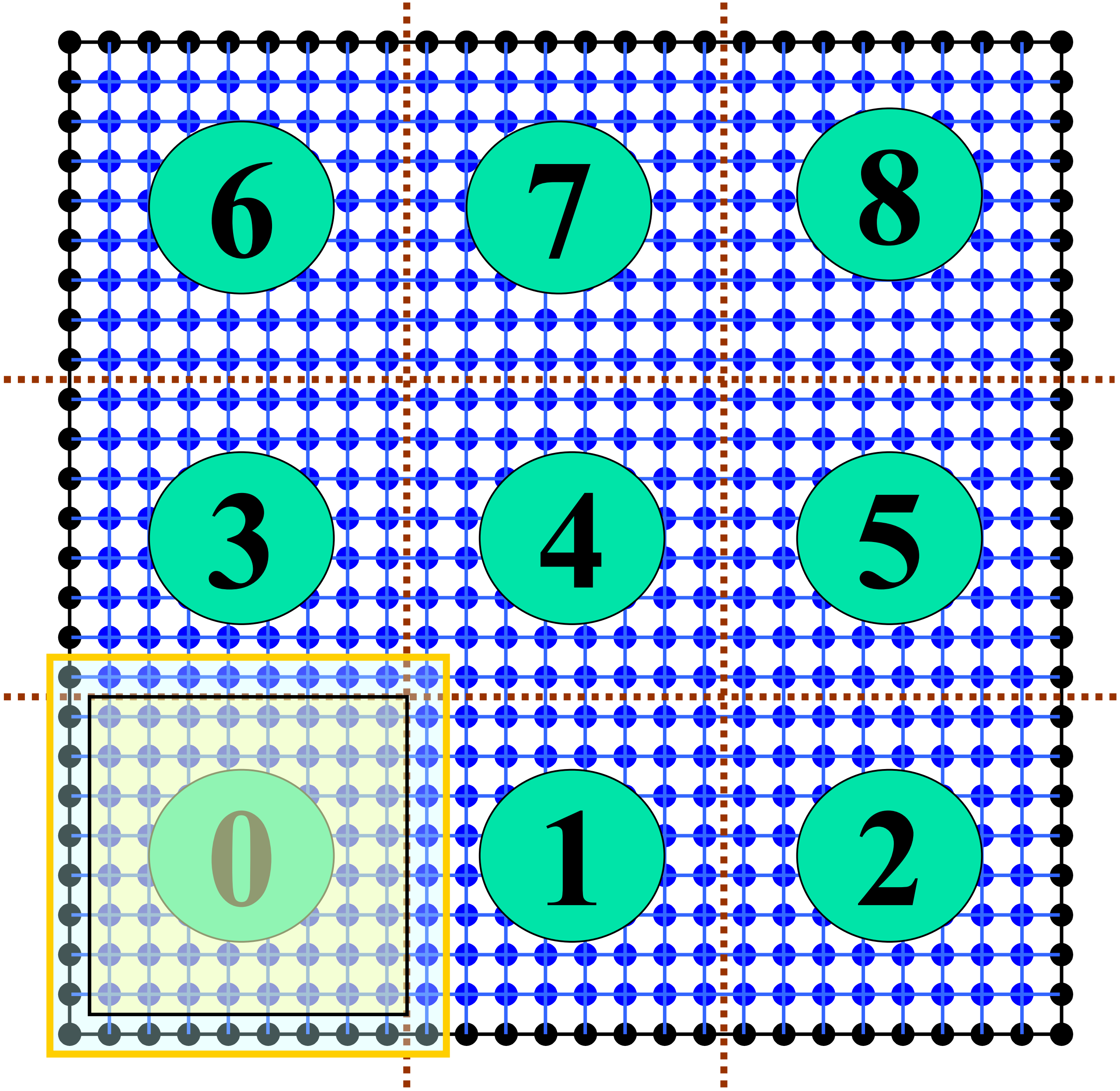
$$u_{ij}^{(k+1)} = d \cdot f_{ij} + d_x \left(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)}\right) + d_y \left(u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)}\right)$$

$$\boxed{i = 1, \dots, n-1, \quad j = 1, \dots, m-1}$$

$$\boxed{k = 0, 1, 2, \dots}$$

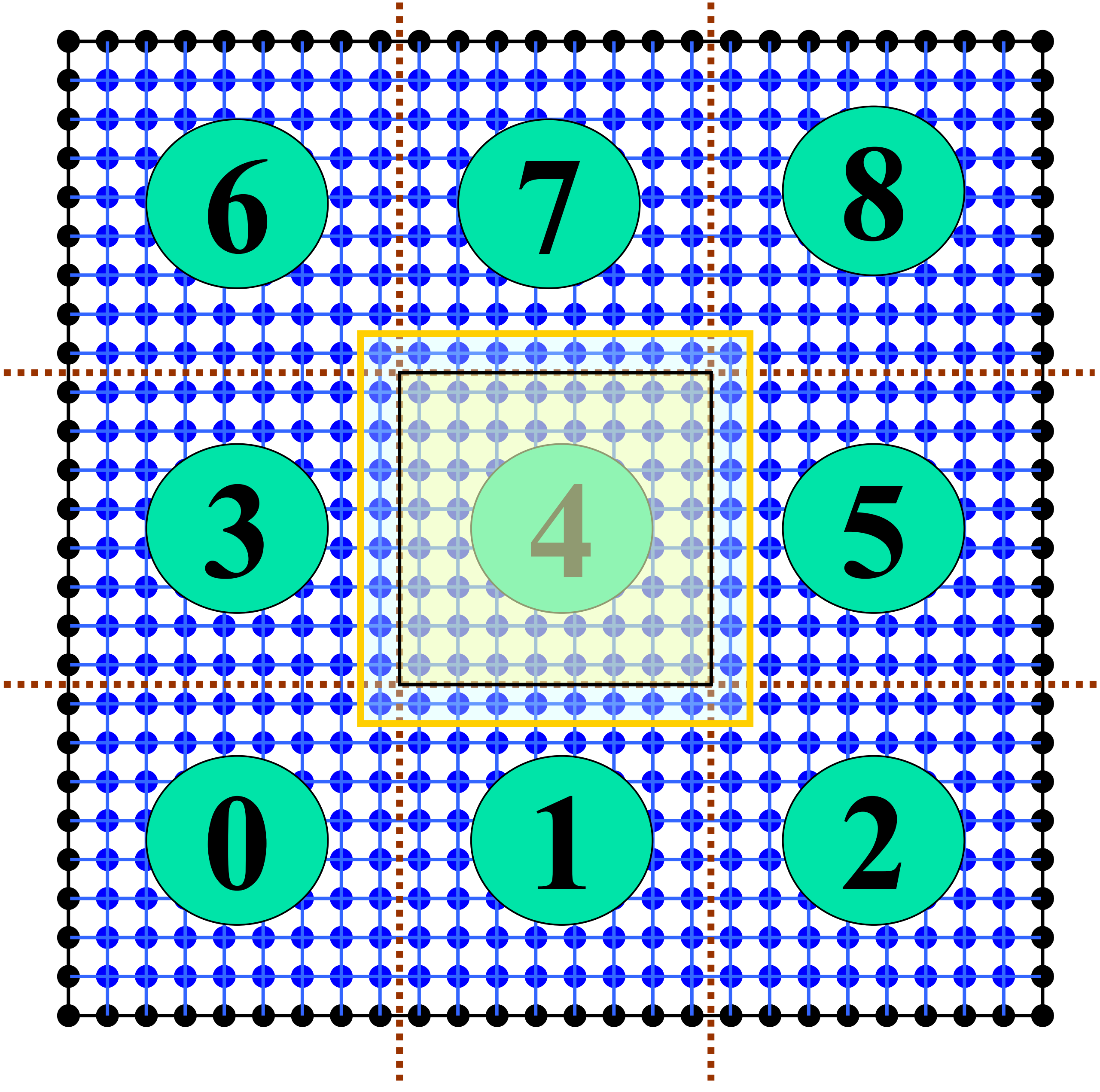
并行算法

- **并行求解的基本思想：区域分解**
  - 采用区域分解技术：
    假设使用 **np** 个进程并行求解，则将整个求解区域分解成 **npx** × **npy** 个子区域，其中 **npx** × **npy** = **np**
  - 每个进程负责求解一个子区域
  - 相邻两个子区域有一个网格步的重叠：
    便于子区域间的数据传递
  - 每个子区域包含的网格点大致相等
  - 以 3 × 3 的区域分解为例



● 蓝色为内点

● 黑色为边界点



蓝色为内点

黑色为边界点

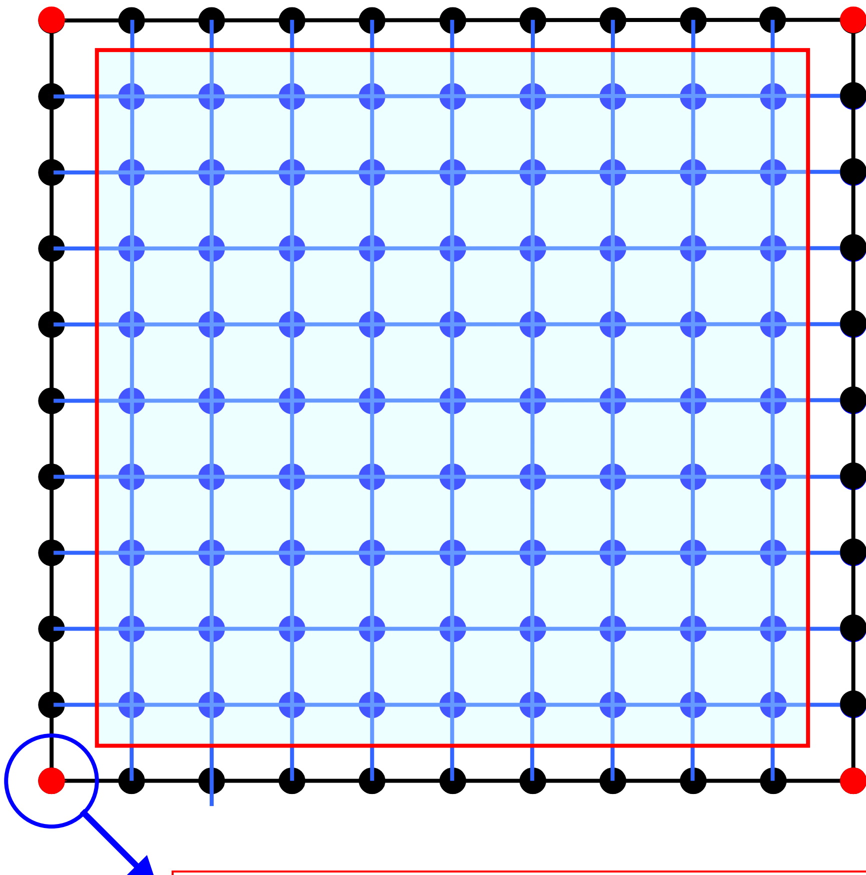
并行算法

- 程序中使用的一些参数:

`np`	进程个数
`npx, npy`	x-方向和 y-方向的进程个数
`myid`	当前进程的进程号
`myidx, myidy`	当前进程的 x-方向和 y-方向的进程坐标
`nx, ny`	整个区域 x-方向和 y-方向的网格点数 `-1`
`nlx, nly`	子区域的 x- 向和 y-方向的网格点数 `-1`
`x0, y0`	子区域的左下角网格点 (0, 0) 在整个区域中的位置（用于计算解析解）

并行算法

子区域



(0,0)：子区域的左下角

- 蓝色为子区域内点
- 黑色为子区域边界点（伪边界）

- 网格点：(0:nlx, 0:nly)
- 内点：
  (1:nlx-1, 1:nly-1)
- “边界点”：
  (0,   1:nly-1)
  (nlx, 1:nly-1)
  (1:nlx-1, 0)
  (1:nlx-1, nly)

并行算法

几个关系式:

- **myidx, myidy 与 myid 的关系式:**

```
myidx = myid % npx
myidy = myid / npx
myid  = myidx + myidy * npx
```

- **nlx 与 nx 的关系式:**

$$nlx = \begin{cases} (nx-1)/npx + 2, & (myidx < rx) \\ (nx-1)/npx + 1, & (myidx \geq rx) \end{cases}$$

其中: rx = (nx-1) % npx

- **nly 与 ny 的关系式类似**

并行算法

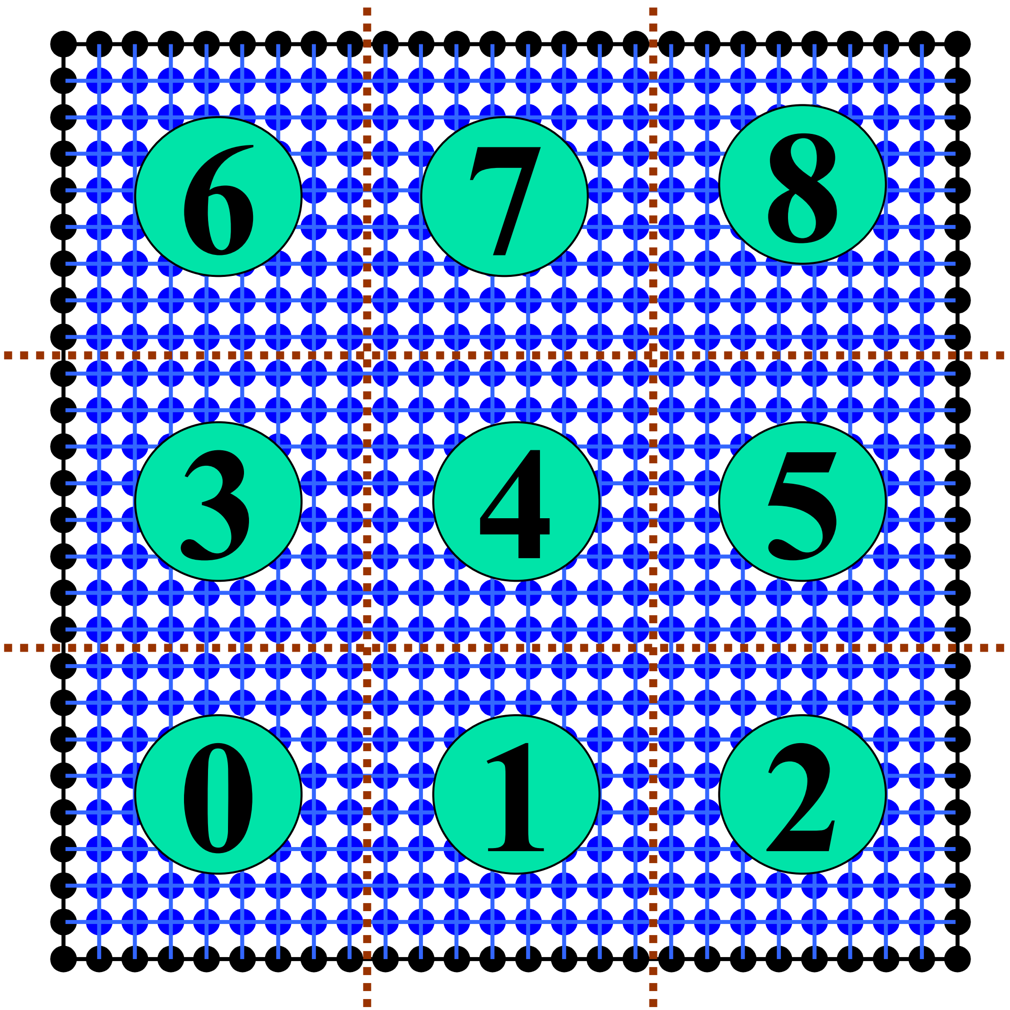
- 子区域中的原点 (0,0) 在整个网格中的坐标

```
x0 = myidx * (nx-1)/npx + min(myidx, rx)
y0 = myidy * (ny-1)/npy + min(myidy, ry)
```

其中：
```
rx = (nx-1) % npx
ry = (ny-1) % npy
```

上机作业

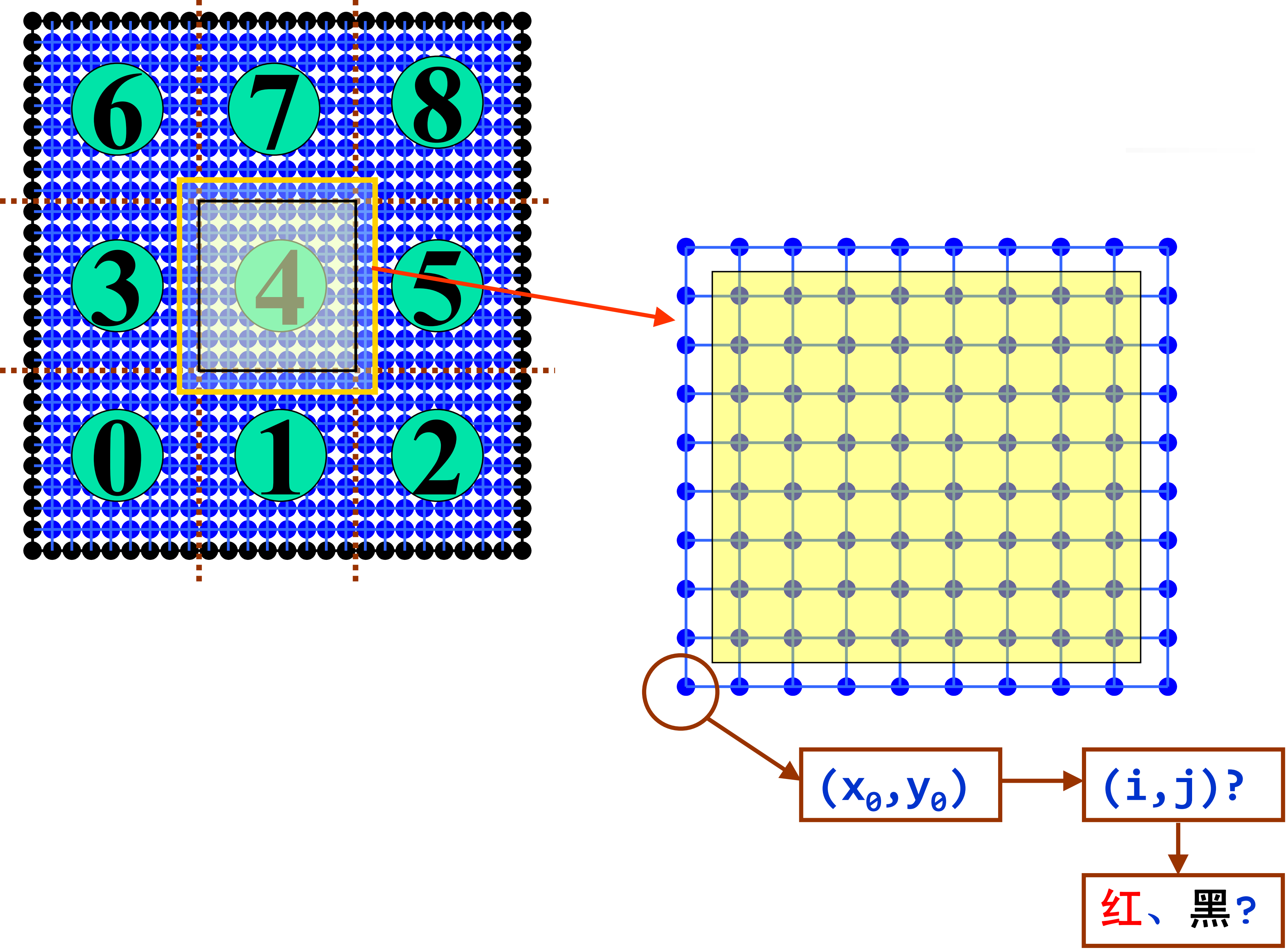
- **将 Jacobi 迭代改为 红黑排序的 Gauss-Seidel 迭代**



红点： i+j=2k
黑点： i+j=2k+1

红黑排序 GS 算法:

- 先更新红点的值
- 再更新黑点的值
- 依次类推，不断循环


6
7
8
3
4
5
0
1
2
(x0,y0)
(i,j)?
红、黑?