Jacobi method

Professor Seokkoo Kang
Department of Civil & Environmental Engineering
kangsk78@hanyang.ac.kr

Jacobi method

- A method for solving a system of linear equations, Ax=b.

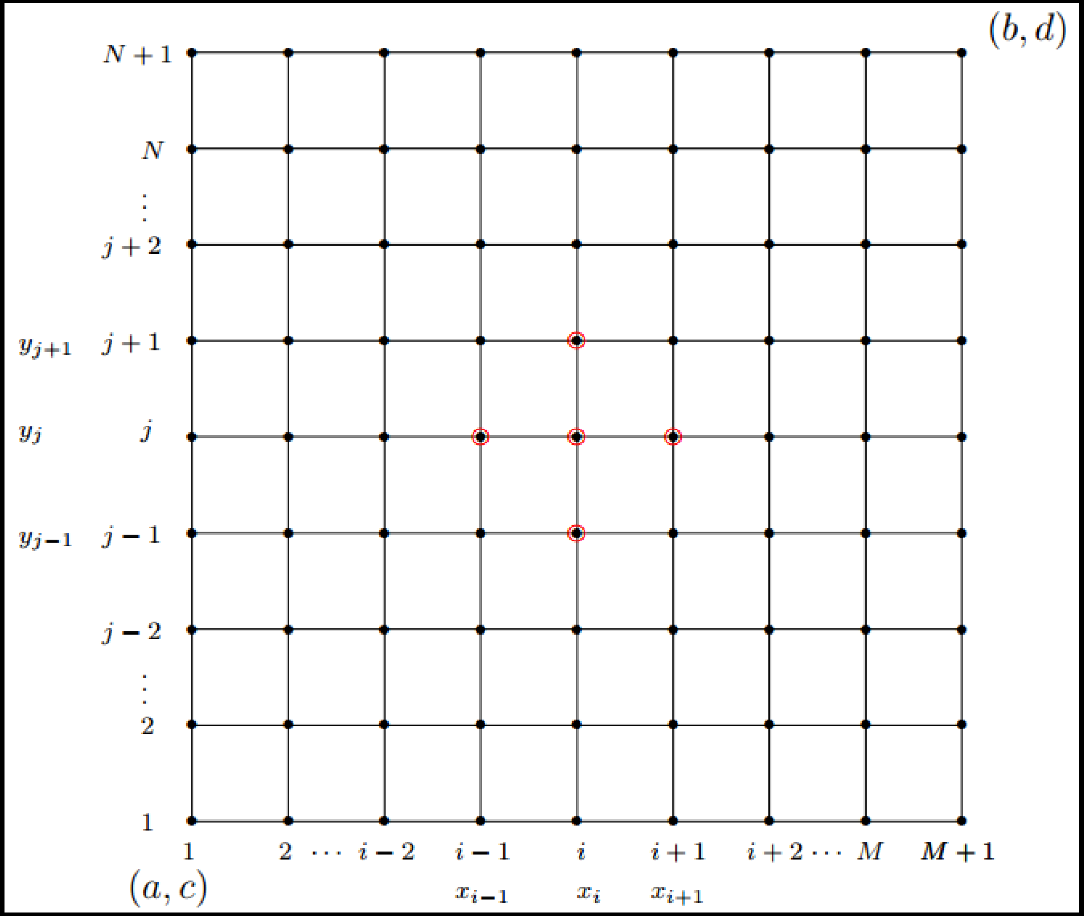
$$x_m^{(k+1)} = \frac{1}{a_{mm}}\left(b_m - \sum_{m \neq n} a_{mn} x_n^{(k)}\right)$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

- You need $x_i^{(0)}$, which is an initial guess.
- x_i at the (k+1)-th iteration is obtained using x_j ($i \neq j$) at the k-th iteration.

2D Poisson Equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \quad \text{(Poisson equation)}$$



Finite Difference Method

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \quad \text{(Poisson equation)}$$

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta y^2} = f_{i,j}$$

$$Ax = b$$

$$\begin{pmatrix} d & & & & & & & \\ \times & d & & \times & \times & & \times & \\ \times & & d & & \times & \times & \times & \\ & & & \cdots & & \times & \times & \\ & & & & \cdots & & & \times \\ & & & & & \cdots & & \\ & \times & \times & & \times & & d & \times \\ & \times & \times & & \times & \times & & d \end{pmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ \\ \\ \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \cdots \\ \\ \\ \\ f_{N-1} \\ f_N \end{bmatrix}$$

Apply

$$x_m^{(k+1)} = \frac{1}{a_{mm}} \left(b_m - \sum_{m \neq n} a_{mn} x_n^{(k)} \right)$$

$$x_m^{(k+1)} = \frac{1}{a_{mm}}\left(b_m - \sum_{m \neq n} a_{mn} x_n^{(k)} \right)$$

Jacobi method: $u_{i,j}^{k+1} = \dfrac{f_{i,j} - (u_{i-1,j}^{k} + u_{i+1,j}^{k}) / \Delta x^2 - (u_{i,j-1}^{k} + u_{i,j+1}^{k}) / \Delta y^2}{-2(1/\Delta x^2 + 1/\Delta y^2)}$

$$u_{i,j}^{k+1} = \frac{\Delta y^2 (u_{i-1,j}^{k} + u_{i+1,j}^{k}) + \Delta x^2 (u_{i,j-1}^{k} + u_{i,j+1}^{k}) - \Delta x^2 \Delta y^2 f_{i,j}}{2(\Delta x^2 + \Delta y^2)}$$

Repeat until max error = $\max |u_m^{k+1} - u_m^{k}| < 10^{-4}$

Assignment (by 11/27)

- Write a serial program that solves the following Poisson equation.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \cos[\pi x]\sin[y] + \pi^2 \cos[\pi x]\sin[y] \quad (0 \le x \le 2,\ 0 \le y \le 2)$$

- Consider grid nodes Np*Np (Np=50,100,200)
- Prescribe a Dirichlet BC on the four boundaries.
- Compare the numerical solution with the exact solution by plotting 2D contours for different Np. $u_{exact}(x, y) = \cos[\pi x]\sin[\pi + y]$
- Plot L2 error vs. Np

$$\text{L2 error=}\sqrt{\sum_{m=1}^{N}(u_m^{final} - u_m^{exact})^2 \ / \ N}$$

Final Project

- Repeat the previous problem using a parallel Jacobi method.
- Use at least Np>=500 grid nodes.
- Run more than 5 cases with different Np.
- Compare execution time, L2 error.
- Due date: 12/11/2017
- Prepare PPT for presentation & final report, and upload them to portal.hanyang.ac.kr after the presentation

Parallel Jacobi Method

- $x^{(k+1)}$ uses only the values at the same row.
- It does not require communication between ranks.

$$x_m^{(k+1)} = \frac{1}{a_{mm}}\left(b_m - \sum_{m \neq n} a_{mn} x_n^{(k)}\right)$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Steps

1. Make a node numbering.
2. Block-row partitioning.

Local vector

$$
\left(\begin{array}{cccccccc}
d & & & & & & & \\
\times & d & & \times & \times & & \times & \\
\hline
\times & & d & & \times & \times & \times & \\
 & & & \cdots & & \times & \times & \\
\hline
 & & & & \cdots & & & \times \\
 & & & & & \cdots & & \\
\hline
 & \times & \times & & \times & & d & \times \\
 & \times & \times & & \times & \times & & d
\end{array}\right)
\begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ \\ \\ \\ u_{N-1} \\ u_N \end{bmatrix}
=
\begin{bmatrix} f_1 \\ f_2 \\ \cdots \\ \\ \\ \\ f_{N-1} \\ f_N \end{bmatrix}
$$

3. Define a local vector (size~N/np), global vector (size N)

3. Use the parallel matrix-vector product code (your previous assignment) to get

$$x_m^{(k+1)} = \frac{1}{a_{mm}}\left(b_m - \boxed{\sum_{m \neq n} a_{mn} x_n^{(k)}}\right)$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

4. Update x^{k+1} in all local vectors.
5. Gather local values to global vectors, and broadcast. (or simply call MPI_AllGather)
6. Stop (if $\max | u_m^{k+1} - u_m^k | < 10^{-4}$) or go to 1.