Racing to Convergence

Comparing Parallel Speedup of Damped Jacobi and Gauss-Seidel Methods

Dennys Huber

March 27, 2024

Numerical Training

Table of contents

Iterative Methods

Goal

How to solve a linear system numerically …

- Quickly
- Efficiently
- Cost-Effective
- with Flexibility

$$Ax = f$$

Stationary Iterative Methods

Let $A \in \mathbb{R}^{n \times n}$ be s.p.d, $f \in \mathbb{R}^n$, $x^{(0)} \in \mathbb{R}^n$ an initial guess and the sequence of iterates $x^{(m)}$ for $m = 1, 2, \ldots$.

$$x^{(m+1)} = x^{(m)} - Mr^{(m)}$$

with the residual r

$$r^{(m)} = \left(Ax^{(m)} - f\right).$$

Construction of M

$$A = D + L + U, \quad \left\{ \begin{array}{l} D \text{ diagonal matrix,} \\ L \text{ strictly lower triangular matrix} \\ U \text{ strictly upper triangular matrix.} \end{array} \right\}$$

Damped Jacobi

$$M_{DJ} = \omega D^{-1}$$

Gauss-Seidel

$$M_{GS} = (D + L)^{-1}$$

Competitors

Damped Jacobi

$$x^{(m+1)} = x^{(m)} - \omega D^{-1}\left(Ax^{(m)} - f\right),$$

Gauss-Seidel

$$x^{(m+1)} = x^{(m)} - (D + L)^{-1}\left(Ax^{(m)} - f\right)$$

Stopping Criteria

Stopping paramter $\epsilon > 0$

$$\frac{\|r^{(m)}\|}{\|x^{(m)}\|} < \epsilon.$$

Convergence

- Condition number

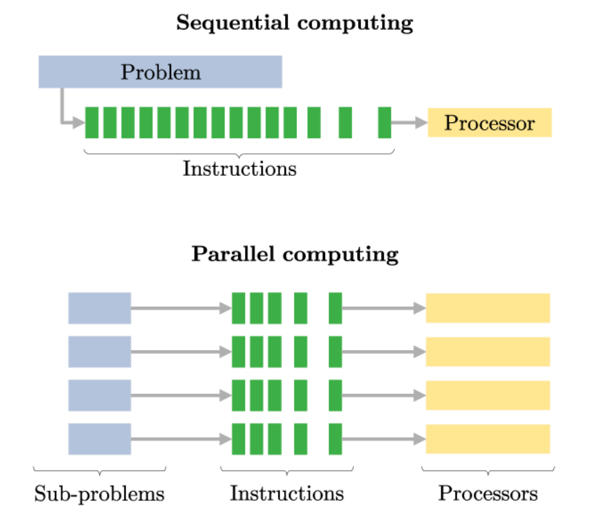
$$\kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$$

- Convergence rate

$$\|C\| \leq \frac{\kappa(A) - 1}{\kappa(A) + 1} = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}$$

Parallel Computing

What is parallel computing?



When is parallel programming useful?

Benefits

- Increased Performance
- Improved Efficiency
- Solving Big Data Problems
- Simulation and Modeling
- Real-Time Applications

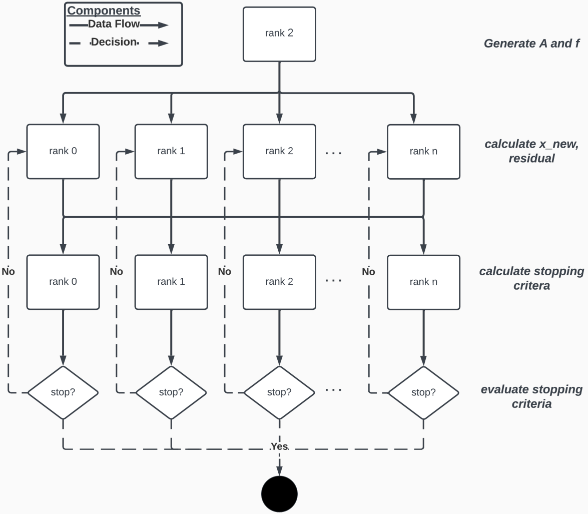
Challenges

- Load Balancing
- Communication Overhead
- Data Dependencies
- Scalability

Element-based Formula Damped Jacobi

$$x_i^{(m+1)} = x_i^{(m)} - \frac{\omega}{a_{ii}} \left[\sum_{j=1}^{n} a_{ij} x_j^{(m)} - f_i \right]$$

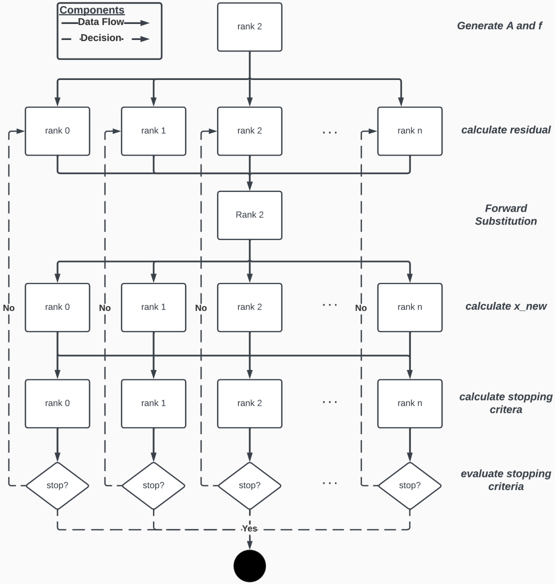
Parallelization of Damped Jacobi



Element-based Formula Gauss-Seidel

$$x_i^{(m+1)} := x_i^{(m)} - \frac{1}{a_{ii}} \left[\sum_{j=1}^{i-1} a_{ij} x_j^{(m+1)} + \sum_{j=i}^{n} a_{ij} x_j^{(m)} - f_i \right]$$

Parallelization of Gauss-Seidel



Model Problem

Model Problem

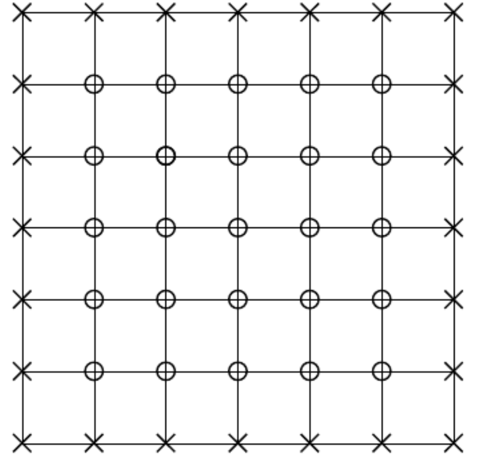
A differential equation is defined on the unit square $\Omega = (0, 1) \times (0, 1)$ as

$$-\Delta u(x, y) = f(x, y) \quad \text{for } (x, y) \in \Omega,$$

$$u(x, y) = \varphi(x, y) \quad \text{on } \Gamma = \partial\Omega$$

Δ is the two-dimensional Laplace operator given by $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$.

Grid



- Grid Ω_h with inner grid points ($\circ$) and boundary points ($\times$)
- Grid dimension N
- Step size $h = 1/N$

Five-Point Formula

$$
\begin{aligned}
-\Delta u(ih, jh) &= \frac{-u_{i-1,j} + 2u_{i,j} - u_{i+1,j}}{h^2} + \frac{-u_{i,j-1} + 2u_{i,j} - u_{i,j+1}}{h^2} \\
&= \frac{1}{h^2}[4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}] \\
&= f_{ij}
\end{aligned}
$$

$$
\begin{aligned}
&u_{i,j} := u(x, y) = u(ih, jh), \\
&f_{i,j} := f(ih, jh)
\end{aligned}
$$

Lexograhic Ordering

Transforming unknowns u_{ij} into a one dimensional vector of size $n = (N-1)^2$

	21	22	23	24	25
	16	17	18	19	20
	11	12	13	14	15
	6	7	8	9	10
	1	2	3	4	5

Matrix A

$$A = h^{-2} \begin{bmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{bmatrix}, \quad T = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix}$$

- $A \in \mathbb{R}^{n \times n}$
- $T, I \in \mathbb{R}^{(N-1) \times (N-1)}$

Five-Point

- Functions

$$u(x, y) = x(1 - x)y(1 - y)$$

$$f(x, y) = 2(x(1 - x) + y(1 - y))$$

- Condition Number

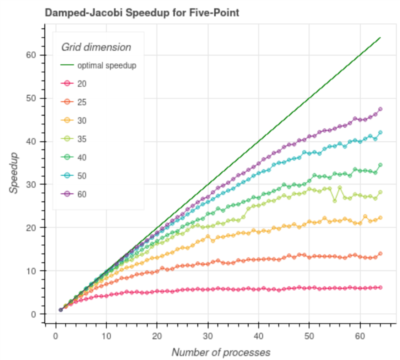
$$\kappa(A) = Ch^{-2}$$

Experiments

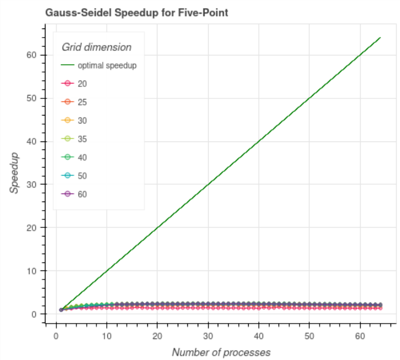
Experiment Remarks

- As low level as possible
- MPI
- One process per core
- Load-balancing
- Shared Server

Speedup Five-Point

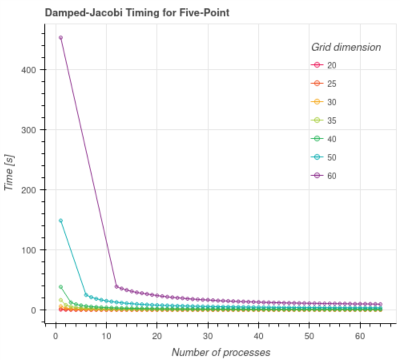


(a) Damped Jacobi

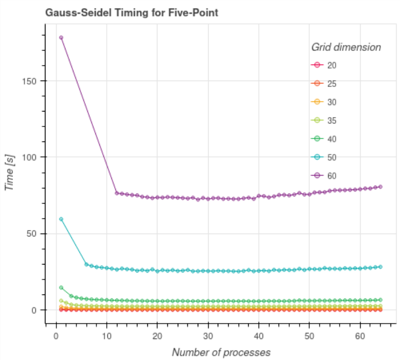


(b) Gauss-Seidel

Timing Five-Point

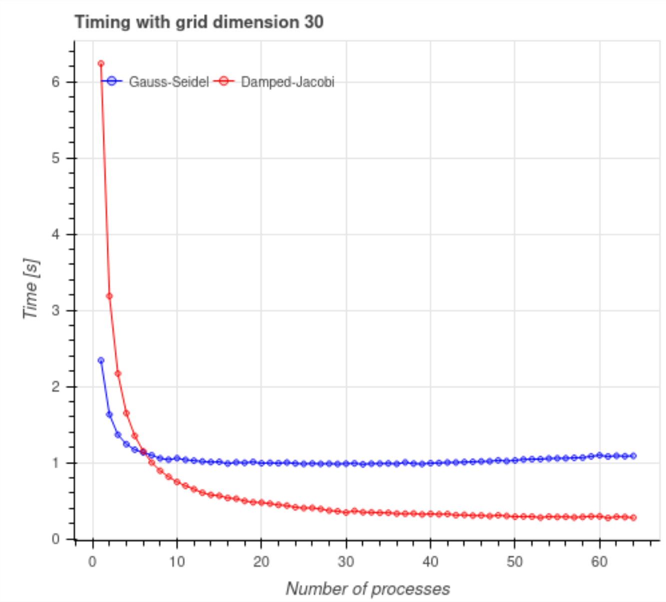


(a) Damped Jacobi

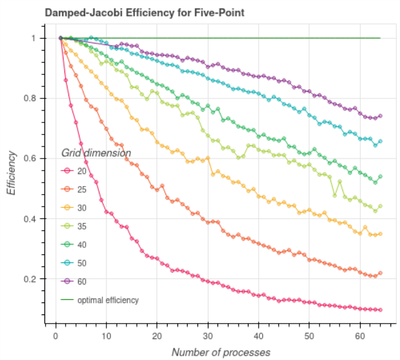


(b) Gauss-Seidel

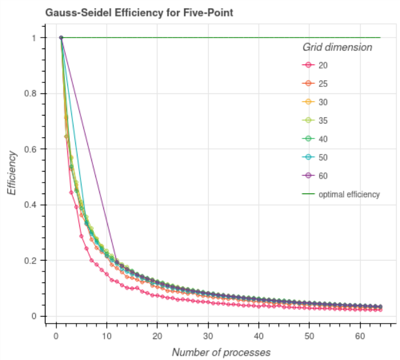
Head to Head Five-Point



Efficency Five-Point



(a) Damped Jacobi



(b) Gauss-Seidel

L2-Projection

$$M_{L2} = \begin{bmatrix} S & L & & & \\ L & S & L & & \\ & \ddots & \ddots & \ddots & \\ & & L & S & L \\ & & & L & S \end{bmatrix}$$

$$S = \begin{bmatrix} \frac{4}{9} & \frac{1}{9} & & & \\ \frac{1}{9} & \frac{4}{9} & \frac{1}{9} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{9} & \frac{4}{9} & \frac{1}{9} \\ & & & \frac{1}{9} & \frac{4}{9} \end{bmatrix}, \quad L = \begin{bmatrix} \frac{1}{9} & \frac{1}{36} & & & \\ \frac{1}{36} & \frac{1}{9} & \frac{1}{36} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{36} & \frac{1}{9} & \frac{1}{36} \\ & & & \frac{1}{36} & \frac{1}{9} \end{bmatrix}$$

- $M_{L2} \in \mathbb{R}^{n \times n}$
- $S, L \in \mathbb{R}^{(N-1) \times (N-1)}$

L2 Projection

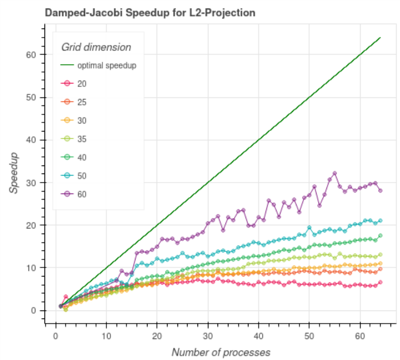
- Functions

$$u(x,y) = (1-x)^2(1-y)^2$$
$$\Delta u(x,y) = f(x,y) = 2((1-x)^2 + (1-y)^2)$$

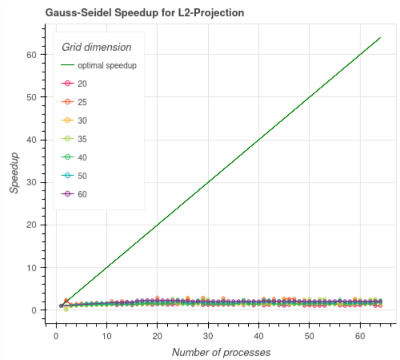
- Condition Number

$$\kappa(M_{L2}) = C, \quad C > 1$$

Speedup L2-Projection

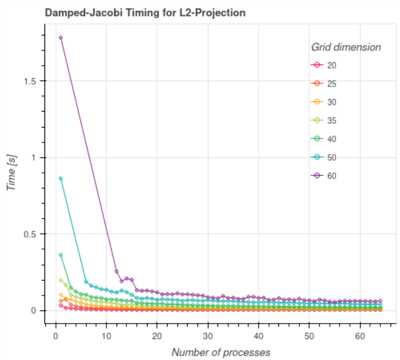


(a) Damped Jacobi

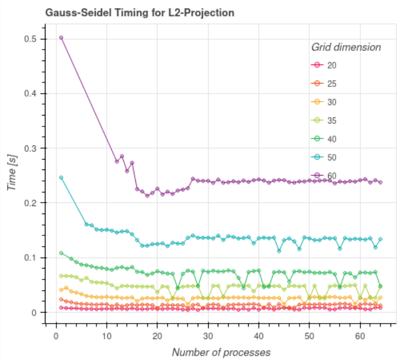


(b) Gauss-Seidel

Timing L2-Projection

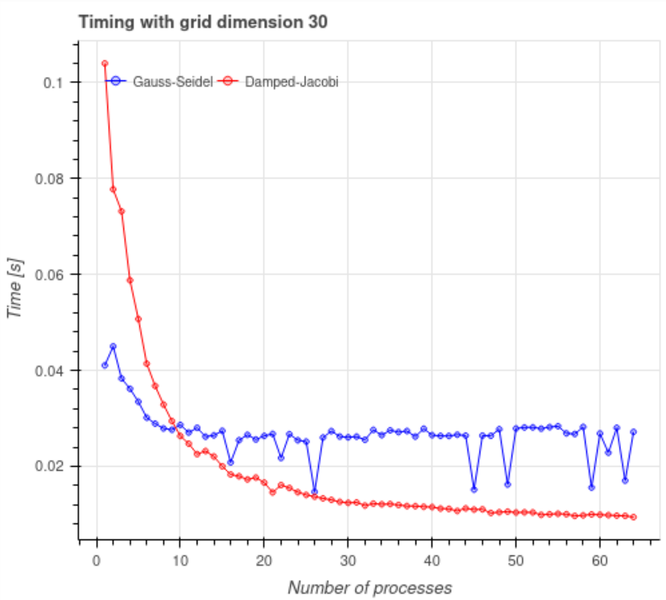


(a) Damped Jacobi

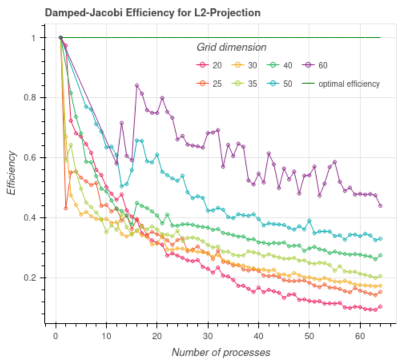


(b) Gauss-Seidel

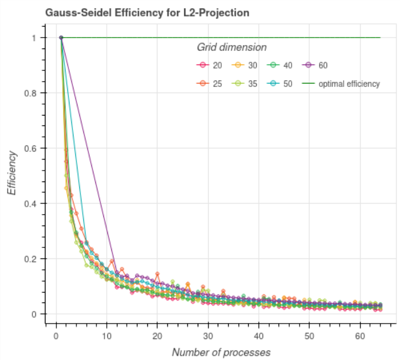
Head to Head L2-Projection



Efficiency L2-Projection



(a) Damped Jacobi



(b) Gauss-Seidel

Conclusion

- Success
- Damped Jacobi better suited for parallel programming
- Still performance to gain
- Compare with other iterative methods

Questions?