

Gauss–Seidel Iteration for Linear Systems

Direct vs. Iterative Solvers

MMAE 350: Computational Mechanics

Motivation

We have already seen the *Thomas Algorithm*:

- ▶ Direct
- ▶ Exact
- ▶ $\mathcal{O}(n)$ for tridiagonal systems

What if:

- ▶ The matrix is not tridiagonal?
- ▶ The system is very large?
- ▶ We want an approximate solution that improves iteratively?

Problem Statement

We seek to solve the linear system

$$\mathbf{Ax} = \mathbf{b},$$

where:

▶ $\mathbf{A} \in \mathbb{R}^{n \times n}$

▶ $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$

The Gauss–Seidel method constructs a sequence

$$\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots$$

that converges to the solution.

Component-Wise Update Formula

For each equation $i = 1, \dots, n$:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j < i} a_{ij} x_j^{(k+1)} - \sum_{j > i} a_{ij} x_j^{(k)} \right).$$

Key observation:

- ▶ New values are used immediately
- ▶ Old values are used where necessary

Algorithm Outline

1. Choose an initial guess $\mathbf{x}^{(0)}$
2. For $k = 0, 1, 2, \dots$:
 - ▶ Loop through equations $i = 1$ to n
 - ▶ Update $x_i^{(k+1)}$ using latest values
3. Check convergence

One full sweep through the equations constitutes one iteration.

Convergence Behavior

Gauss–Seidel converges if:

- ▶ \mathbf{A} is strictly diagonally dominant, or
- ▶ \mathbf{A} is symmetric positive definite

Physical interpretation:

- ▶ Diagonal dominance means local effects outweigh coupling
- ▶ Common in diffusion and elasticity problems

Gauss–Seidel vs. Jacobi

Feature	Jacobi	Gauss–Seidel
Uses updated values	No	Yes
Convergence speed	Slower	Faster
Parallel friendly	Yes	Less so

Gauss–Seidel can be viewed as *Jacobi with memory*.

Gauss–Seidel vs. Thomas Algorithm

Feature	Thomas	Gauss–Seidel
Method type	Direct	Iterative
Accuracy	Exact	Approximate
Matrix type	Tridiagonal	General sparse
Cost	$\mathcal{O}(n)$	$\mathcal{O}(n k)$

Thomas is unbeatable when applicable; Gauss–Seidel works when Thomas cannot.

Role in Computational Mechanics

Gauss–Seidel is foundational to:



- ▶ Successive Over-Relaxation (SOR)
- ▶ Multigrid smoothers
- ▶ Nonlinear Newton solvers


Iterative thinking is essential for large-scale simulations.





Key Takeaways

- ▶ Gauss–Seidel is an *iterative* solver
- ▶ Updates occur *in-place*
- ▶ Convergence depends on matrix structure
- ▶ Complements direct methods like Thomas

Gauss–Seidel Sweep: What is “new” vs “old”?

 already updated ($x^{(k+1)}$)  not yet updated ($x^{(k)}$)

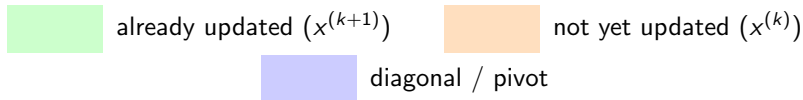
 diagonal / pivot





 a_{11}	 a_{12}	 a_{13}	 a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

$$x_1^{(k+1)} = \frac{1}{a_{11}} \left(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - a_{14}x_4^{(k)} \right)$$

Updating x_1 : everything to the right uses old values.

Gauss–Seidel Sweep: What is “new” vs “old”?

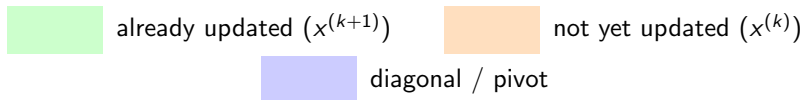






a_{11}	a_{12}	a_{13}	a_{14}
 a_{21}	 a_{22}	 a_{23}	 a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

$$x_2^{(k+1)} = \frac{1}{a_{22}} \left(b_2 - \textcolor{green}{a_{21}}x_1^{(k+1)} - \textcolor{brown}{a_{23}}x_3^{(k)} - \textcolor{brown}{a_{24}}x_4^{(k)} \right)$$

Updating x_2 : left is new (green), right is old (orange).

Gauss–Seidel Sweep: What is “new” vs “old”?










a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
 a_{31}	 a_{32}	 a_{33}	 a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

$$x_3^{(k+1)} = \frac{1}{a_{33}} \left(b_3 - a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)} - a_{34}x_4^{(k)} \right)$$

Updating x_3 : x_1, x_2 are new; x_4 is old.

Gauss–Seidel Sweep: What is “new” vs “old”?

 already updated ($x^{(k+1)}$)  not yet updated ($x^{(k)}$)
 diagonal / pivot

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
 a_{41}	 a_{42}	 a_{43}	 a_{44}

$$x_4^{(k+1)} = \frac{1}{a_{44}} \left(b_4 - a_{41}x_1^{(k+1)} - a_{42}x_2^{(k+1)} - a_{43}x_3^{(k+1)} \right)$$

Updating x_4 : all previous are new; diagonal is the pivot.

Convergence of Gauss–Seidel

Gauss–Seidel produces a sequence

$$\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots$$

that (hopefully) converges to the solution of

$$\mathbf{Ax} = \mathbf{b}.$$

Residual-Based Convergence (Preferred)

Define the residual

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{Ax}^{(k)}.$$

Stop the iteration when

$$\frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{b}\|} < \varepsilon.$$

Update-Based Convergence (Common in Practice)

Alternatively, stop when the solution stops changing: