Thomas Algorithm for Tridiagonal Systems

MMAE 350

Illinois Institute of Technology

January 21, 2026

Why the Thomas Algorithm?

- Many engineering problems lead to **tridiagonal systems**
- Examples:
  - 1D heat conduction (finite differences)
  - 1D Poisson equation
  - Beam and bar discretizations
- Standard Gaussian elimination costs $\mathcal{O}(n^3)$
- Tridiagonal structure allows a solver in $\mathcal{O}(n)$

Tridiagonal Linear System

We solve

$$Ax = d \tag{1}$$

where $\mathbf{A}$ is tridiagonal:

$$\mathbf{A} = \begin{bmatrix} b_0 & c_0 & & & 0 \\ a_1 & b_1 & c_1 & & \\ & a_2 & b_2 & c_2 & \\ & & \ddots & \ddots & c_{n-2} \\ 0 & & & a_{n-1} & b_{n-1} \end{bmatrix} \tag{2}$$

Key Idea

- Thomas algorithm is Gaussian elimination **without fill-in**
- We never store the full matrix — only the three diagonals

$$\{a_i\}, \quad \{b_i\}, \quad \{c_i\}$$

- Algorithm steps:
    - Forward sweep (eliminate subdiagonal)
    - Back substitution (solve from bottom to top)

5×5 Hand Calculation: A Concrete Example

Consider the tridiagonal system $Ax = d$ with

$$A = \begin{bmatrix} 4 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 4 \end{bmatrix}, \qquad d = \begin{bmatrix} 3 \\ 2 \\ 2 \\ 2 \\ 3 \end{bmatrix}. \tag{3}$$

- This is a common finite-difference matrix (Poisson / diffusion).
- We will eliminate the subdiagonal entries one row at a time.

5×5 Hand Calculation: Eliminate a_1

Row 1 (index 0) is

$$4x_0 - x_1 = 3.$$

Row 2 (index 1) is

$$-x_0 + 4x_1 - x_2 = 2.$$

To eliminate x_0 from Row 2, multiply Row 1 by

$$m_1 = \frac{a_1}{b_0} = \frac{-1}{4},$$

and subtract:

$$\text{Row } 2 \leftarrow \text{Row } 2 - m_1(\text{Row } 1).$$

The updated Row 2 becomes

$$\underbrace{(4 - m_1(-1))}_{b_1'} x_1 - x_2 = \underbrace{(2 - m_1(3))}_{d_1'}.$$

5×5 Hand Calculation: The Pattern Emerges

At the next step, we eliminate a_2 using the *updated* Row 2. This repeats with the same structure:

- a multiplier m_i (one number)
- an updated diagonal entry b'_i
- an updated right-hand side entry d'_i

This leads directly to the forward-sweep recurrences on the next slide.

Forward Sweep

Initialization:

$$b'_0 = b_0, \qquad d'_0 = d_0 \tag{4}$$

For $i = 1, \dots, n-1$:

$$m_i = \frac{a_i}{b'_{i-1}}, \qquad b'_i = b_i - m_i c_{i-1}, \qquad d'_i = d_i - m_i d'_{i-1}. \tag{5}$$

Back Substitution

After the forward sweep, the system is upper triangular.

$$x_{n-1} = \frac{d'_{n-1}}{b'_{n-1}} \tag{6}$$

For $i = n-2, \ldots, 0$:

$$x_i = \frac{d'_i - c_i x_{i+1}}{b'_i} \tag{7}$$

Summary

- Specialized solver for tridiagonal systems
- Linear complexity $\mathcal{O}(n)$ and storage $\mathcal{O}(n)$
- Forward sweep updates b_i' and d_i' using one multiplier m_i
- Back substitution recovers x from bottom to top

Why Order n^3 vs. Order n Matters

Computational cost scaling

Problem size n	n^3	n
10	10^3	10
10^2	10^6	10^2
10^3	10^9	10^3
10^6	10^{18}	10^6

Interpretation

- Dense Gaussian elimination:

$$\mathcal{O}(n^3)$$

- Thomas algorithm:

$$\mathcal{O}(n)$$

- Doubling n:
    - $n^3 \to 8\times$ more work
    - $n \to 2\times$ more work

Key takeaway

Thomas is fast because it exploits structure, not because it changes the physics.