Tridiagonal Linear Systems

Zeyad Waleed 202201767

Introduction: What Is a Tridiagonal System?

A **tridiagonal system** is a type of linear system where the coefficient matrix has non-zero entries only on the *main diagonal*, the *diagonal above it*, and the *diagonal below it*. All other elements are zero, which makes the system simpler and more efficient to solve.

Where Are Tridiagonal Systems Used?

Tridiagonal systems frequently occur in:

- Finite difference methods
- Thermal simulations (e.g., heat conduction)
- Modeling fluid flow
- Computer graphics and image processing

The Thomas Algorithm for Solving Tridiagonal Systems

The **Thomas Algorithm** is a modified form of Gaussian elimination specifically optimized for tridiagonal matrices. It consists of two key steps:

Phase 1: Forward Sweep

- Eliminates the lower diagonal elements.
- Transforms the system into upper-triangular form.

Phase 2: Back Substitution

- Solves the system by calculating the unknowns from bottom to top.

This algorithm has a linear time complexity of $\mathcal{O}(n)$, which is far more efficient than the $\mathcal{O}(n^3)$ complexity of standard Gaussian elimination.

Example: Solving a 4x4 Tridiagonal System

Consider the following system:

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

Corresponding vectors:

- $a = [-1, -1, -1]$ (lower diagonal)
- $b = [2, 2, 2, 2]$ (main diagonal)
- $c = [-1, -1, -1]$ (upper diagonal)
- $d = [1, 2, 2, 1]$ (right-hand side)

Python Code Implementation

```
def thomas_algorithm(a, b, c, d):
    n = len(b)
    cp = [0] * (n - 1)
    dp = [0] * n

    cp[0] = c[0] / b[0]
    dp[0] = d[0] / b[0]

    for i in range(1, n - 1):
        denom = b[i] - a[i - 1] * cp[i - 1]
        cp[i] = c[i] / denom
        dp[i] = (d[i] - a[i - 1] * dp[i - 1]) / denom

    dp[n - 1] = (d[n - 1] - a[n - 2] * dp[n - 2]) / (b[n - 1] - a[n -
       2] * cp[n - 2])

    x = [0] * n
    x[-1] = dp[-1]

    for i in range(n - 2, -1, -1):
        x[i] = dp[i] - cp[i] * x[i + 1]

    return x

# Example usage
solution = thomas_algorithm([-1, -1, -1], [2, 2, 2, 2], [-1, -1, -1],
   [1, 2, 2, 1])
print(solution)
```

Expected Output

$$[2.0,\ 3.0,\ 3.0,\ 2.0]$$

Performance Analysis

Efficiency

- The Thomas Algorithm handles this system in linear time $\mathcal{O}(n)$.
- For large-scale systems (e.g., 1000x1000), it is far more efficient and memory-friendly than full Gaussian elimination.

Accuracy

- The algorithm is numerically stable for well-conditioned systems like the one in this example.
- It avoids unnecessary operations and minimizes round-off error by working only on the three diagonals.

Conclusion

The Thomas Algorithm is a powerful and optimized approach to solving tridiagonal systems. Its linear time complexity, accuracy, and simplicity make it a vital tool in simulations involving physics, engineering, and computational science.