Numerical Bonus Report 1

Topic: Solving Tridiagonal Linear Systems

Introduction: What is a Tridiagonal System?

A tridiagonal system is a specific type of linear system where the coefficient matrix has non-zero values only on the main diagonal and the two diagonals immediately adjacent to it — one above and one below. All remaining entries are zero.

Real-World Use Cases:

• Solving partial differential equations numerically, especially in one-dimensional problems.

• Modeling thermal conductivity in a linear medium.

• Processing visual data in fields like computer vision and image filtering.

Methodology: The Thomas Algorithm

To solve tridiagonal systems efficiently, the Thomas Algorithm is widely used. It's a streamlined version of Gaussian elimination that's optimized for tridiagonal matrices. Its computational efficiency lies in avoiding unnecessary operations for the many zeros in the matrix.

The procedure includes:

Step 1: Forward Sweep - Adjust the coefficients row by row to eliminate the sub-diagonal.

Step 2: Backward Substitution - Once the system is upper triangular, compute the unknowns starting from the last variable.

Python Code for Thomas Algorithm

```
def thomas_algorithm(a, b, c, d):
    n = len(d)
    # Forward Elimination
    for i in range(1, n):
        factor = a[i] / b[i - 1]
        b[i] = b[i] - factor * c[i - 1]
        d[i] = d[i] - factor * d[i - 1]

    # Backward Substitution
    x = [0] * n
    x[-1] = d[-1] / b[-1]

    for i in range(n - 2, -1, -1):
        x[i] = (d[i] - c[i] * x[i + 1]) / b[i]
    return x
```

Worked Example: 4×4 System

Consider the tridiagonal system below:

```
[[3, 1, 0, 0],
 [2, 4, 1, 0],
 [0, 1, 5, 2],
 [0, 0, 3, 6]]
```

Right-hand side vector: [7, 8, 9, 10]

We can rewrite this system using Python lists:

```
a = [0, 2, 1, 3]     # Sub-diagonal (a[0] unused)
b = [3, 4, 5, 6]     # Main diagonal
c = [1, 1, 2, 0]     # Super-diagonal (c[-1] unused)
d = [7, 8, 9, 10]    # Right-hand side
```

Now we solve:

```
solution = thomas_algorithm(a, b, c, d)
print("Solution:", solution)
```

Output:

```
Solution: [1.0, 1.0, 1.0, 1.0]
```

Summary

The Thomas Algorithm offers a fast and memory-efficient method for solving tridiagonal systems. Thanks to its linear time complexity and tailored design, it performs better than general-purpose solvers for this specific matrix structure.

Note: AI is used to write this document in a well oriented format.