Numerical Practice 2

Aakash Patil

aakash.patil@master.centralelille.fr

Objectives

1. Study of Tridiagonal matrix Method.
2. Study of error as a function of number of discretization points.

---

1. Tridiagonal Matrix

A tridiagonal matrix is a sparse matrix that has nonzero elements only on the main diagonal, the first diagonal below this, and the first diagonal above the main diagonal. This means that such matrices all have 0s at the same known positions. For instance, the only nonzero elements are either on the diagonal or on subdiagonals just below or above the diagonal, and all other elements are guaranteed to be 0, such as for a $n \times n$ tridiagonal matrix

$$A = \begin{bmatrix} a & b & 0 & \cdots & 0 \\ c & a & b & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & c & a & b \\ 0 & \cdots & 0 & c & a \end{bmatrix}$$

2. Tridiagonal Matrix Algorithm

In numerical linear algebra, the tridiagonal matrix algorithm, also known as the Thomas algorithm (named after Llewellyn Thomas), is a simplified form of Gaussian elimination that can be used to solve tridiagonal systems of equations. A tridiagonal system for n unknowns may be written as

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i,$$

where $a_1 = 0$ and $c_n = 0$.

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}.$$

For such systems, the solution can be obtained in $O(n)$ operations instead of $O(n^3)$ required by Gaussian elimination. A first sweep eliminates the a_i's, and then an (abbreviated) backward substitution produces the solution. Examples of such matrices commonly arise from the discretization of 1D Poisson equation; similar systems of matrices also arise in the nearest neighbor effect models.

Thomas' algorithm is not numerically stable in general, but is so in several special cases, such as when the matrix is diagonally dominant(either by rows or columns) or symmetric positive definite. If stability is required in the general case, Gaussian elimination with partial pivoting is recommended instead.

2.1. Method

The forward sweep consists of modifying the coefficients as follows, denoting the new coefficients with primes:

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & ; \quad i = 1 \\ \dfrac{c_i}{b_i - a_i c'_{i-1}} & ; \quad i = 2, 3, \ldots, n-1 \end{cases}$$

and

$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & ; \quad i = 1 \\ \dfrac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} & ; \quad i = 2, 3, \ldots, n. \end{cases}$$

The solution is then obtained by back substitution:

$$x_n = d'_n$$

$$x_i = d'_i - c'_i x_{i+1} \qquad ; \; i = n-1, n-2, \ldots, 1.$$

2.2. Derivation

The derivation of the tridiagonal matrix algorithm is a special case of Gaussian elimination.

Suppose that the unknowns are $x_1, \ldots, x_n$, and that the equations to be solved are:

$$\begin{aligned} b_1x_1 + c_1x_2 &= d_1; \quad i = 1 \\ a_ix_{i-1} + b_ix_i + c_ix_{i+1} &= d_i; \quad i = 2, \ldots, n-1 \\ a_nx_{n-1} + b_nx_n &= d_n; \quad i = n. \end{aligned}$$

Consider modifying the second ($i = 2$) equation with the first equation as follows:

$$(\text{equation } 2) \cdot b_1 - (\text{equation } 1) \cdot a_2$$

which would give:

$$(a_2x_1 + b_2x_2 + c_2x_3)b_1 - (b_1x_1 + c_1x_2)a_2 = d_2b_1 - d_1a_2$$

$$(b_2b_1 - c_1a_2)x_2 + c_2b_1x_3 = d_2b_1 - d_1a_2$$

where the second equation immediately above is a simplified version of the equation immediately preceding it. The effect is that x_1 has been eliminated from the second equation. Using a similar tactic with the modified second equation on the third equation yields:

$$(a_3x_2+b_3x_3+c_3x_4)(b_2b_1-c_1a_2)-((b_2b_1-c_1a_2)x_2+c_2b_1x_3)a_3 = d_3(b_2b_1-c_1a_2)-(d_2b_1-d_1a_2)a_3$$

$$(b_3(b_2b_1 - c_1a_2) - c_2b_1a_3)x_3 + c_3(b_2b_1 - c_1a_2)x_4 = d_3(b_2b_1 - c_1a_2) - (d_2b_1 - d_1a_2)a_3.$$

This time x_2 was eliminated. If this procedure is repeated until the n^{th} row; the (modified) n^{th} equation will involve only one unknown, x_n. This may be solved for and then used to solve the $(n-1)^{th}$ equation, and so on until all of the unknowns are solved for.

Clearly, the coefficients on the modified equations get more and more complicated if stated explicitly. By examining the procedure, the modified coefficients (notated with tildes) may instead be defined recursively:

$$\tilde{a}_i = 0$$

$$\tilde{b}_1 = b_1$$

$$\tilde{b}_i = b_i\tilde{b}_{i-1} - \tilde{c}_{i-1}a_i$$

$$\tilde{c}_1 = c_1$$

$$\tilde{c}_i = c_i\tilde{b}_{i-1}$$

$$\tilde{d}_1 = d_1$$

$$\tilde{d}_i = d_i \tilde{b}_{i-1} - \tilde{d}_{i-1} a_i.$$

To further hasten the solution process, $\tilde{b}_i$ may be divided out (if there's no division by zero risk), the newer modified coefficients, each notated with a prime, will be:

$$a'_i = 0$$

$$b'_i = 1$$

$$c'_1 = \frac{c_1}{b_1}$$

$$c'_i = \frac{c_i}{b_i - c'_{i-1} a_i}$$

$$d'_1 = \frac{d_1}{b_1}$$

$$d'_i = \frac{d_i - d'_{i-1} a_i}{b_i - c'_{i-1} a_i}.$$

This gives the following system with the same unknowns and coefficients defined in terms of the original ones above:

$$\begin{array}{lll} x_i + c'_i x_{i+1} = d'_i & ; & i = 1, \ldots, n-1 \\ x_n = d'_n & ; & i = n. \end{array}$$

The last equation involves only one unknown. Solving it in turn reduces the next last equation to one unknown, so that this backward substitution can be used to find all of the unknowns:

$$x_n = d'_n$$

$$x_i = d'_i - c'_i x_{i+1} \qquad ; \; i = n-1, n-2, \ldots, 1.$$

3. Code

Random numbers were used to find the error. Firstly, $xRand$ of size $n \times 1$ was generated using random numbers. Then, d was computed as $d = matmul(A, xRand)$ using the given A of size $n \times n$ and the $xRand$ of size $n \times 1$ generated using random numbers. Later, the function to solve tridiagonal matrix was called by supplying A and the previous d to obtain the numerical value of x.

The code has been uploaded to
https://github.com/aakash30jan/ClassWork-TurbulenceMaster/tree/master/11-11-2017/src/

4. Results

4.1. Plots

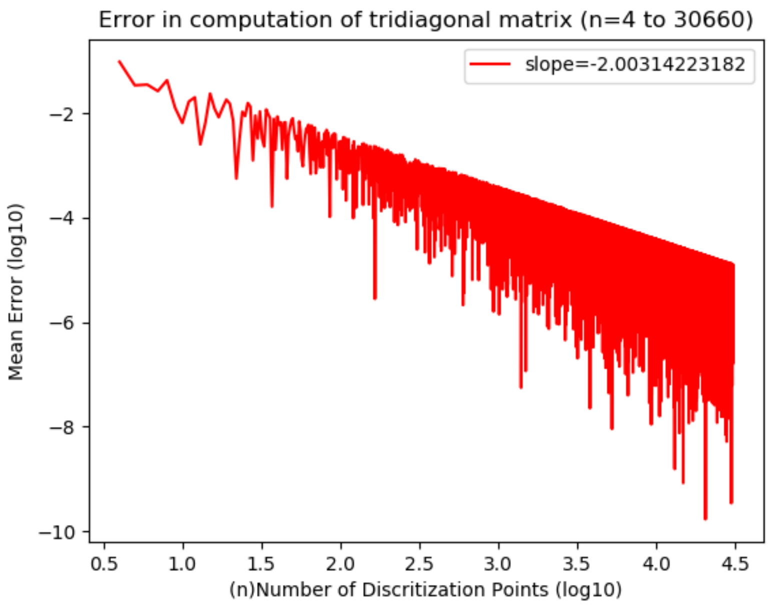

Figure 1: Normal scale with log computed plot of error in numerical computation of tridiagonal matrix (N=4 to 30660 with step of 1)

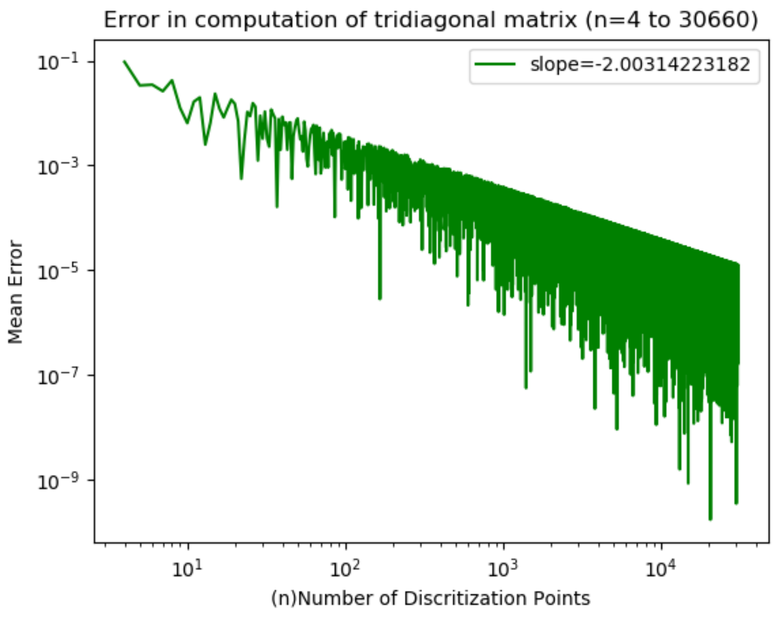

Figure 2: Log scale plot of error in numerical computation of tridiagonal matrix (N=4 to 30660 step of 1)

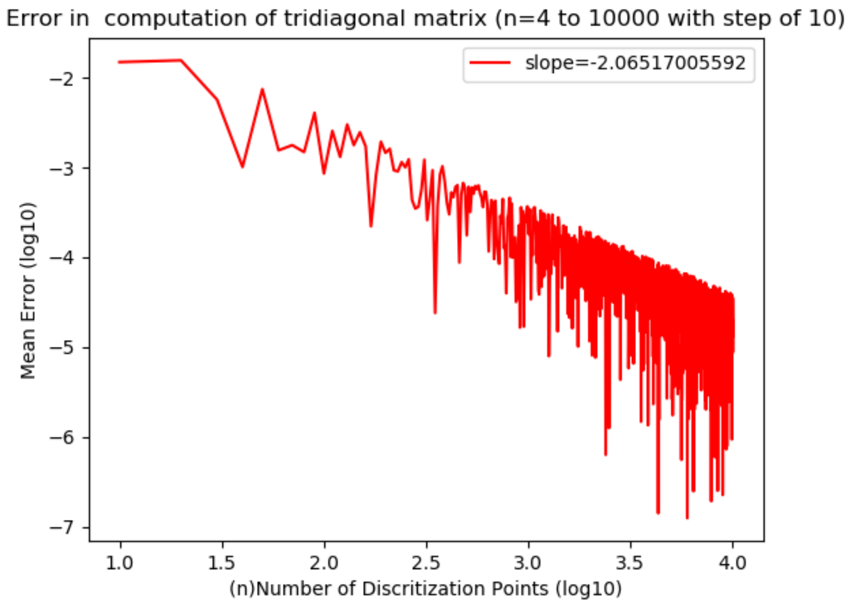

Figure 3: Normal scale with log computed plot of error in numerical computation of tridiagonal matrix (N=4 to 10000 with step of 10)

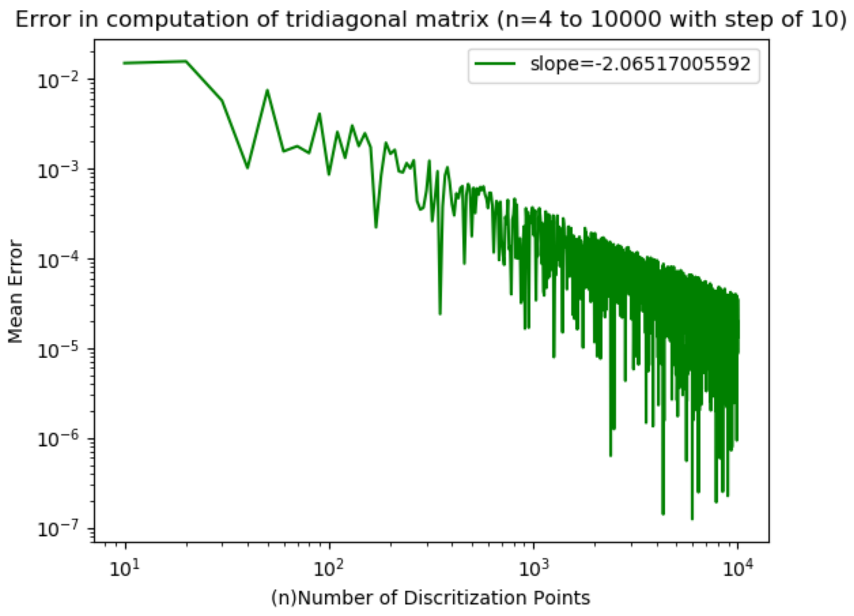

Figure 4: Log scale plot of error in numerical computation of tridiagonal matrix (N=4 to 10000 with step of 10)