4/3/25

ME5107 - Assignment - 3

Name - Jay Prajapati Roll No: ME21B143

Hand Calculations

1) a) A tridiagonal system of the form:

$$Ax = d.$$

$$A = \begin{bmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ 0 & a_2 & b_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & a_{n-1} & b_{n-1} \\ 0 & 0 & \cdots & a_n & a_n \end{bmatrix}$$

where $a_i \rightarrow$ main diagonal

$b_i \rightarrow$ super diagonal

$d \rightarrow$ RHS vector

Step 1: Forward Elimination.

$$a_i' = a_i \qquad a_{i+1}' = a_{i+1}$$

$$a_{i+1}' = a_{i+1} - \frac{b_i}{a_i} \times b_i$$

$$d_{i+1}' = d_{i+1} - \frac{b_i}{a_i} \times d_i'$$

Step 2: Back substitution

$$x_n = \frac{d_n'}{a_n'}$$

$$x_i = \frac{d_i'}{a_i'} \qquad \forall \; i \in \{0, 1, \ldots, n-1\}$$

Pseudo code:

```
for (int i=0; i<n-1; i++) {
        factor = b[i]/a[i];
        a[i+1] -= factor * b[i];
        d[i+1] -= factor * d[i];
}
x[n-1] = d[n-1]/a[n-1];
for (int i=n-2; i>=0; i--){
        x[i] = (d[i] - b[i] * x[i+1])/a[i];
}
```

Time complexity → O(n).

b) Step 1 :

No. of operations = 1 (division) + 2 (multiplications)

= 3

Total operation = 3(n-1).

Step 2 :

No. of operation = 1 (division) + 1 (multiplication)

= 2

Total operation = 2(n-1) + 1 (x[n-1])

∴ Total operation = 3(n-1) + 2(n-1) + 1 = 5n - 4

In thomas algo for tridiagonal system,

Forward elimination = $3(n-1)$

Back substitution = $2(n-1)+1$.

Total = $5n-4$

∴ Thus, Flop count is exactly same for our modified Gaussian elimination and Thomas algorithm.

c) Step 1 :- Forward Elimination

For each row i, eliminate elements below it but only within Bandwidth B.

- Pivot current row.
- Eliminate dependent rows within B.
- Modify RHS vector accordingly.

Step 2 :- Back Substitution

After elimination, system becomes upper triangular, allowing an efficient solution.

- Solve last unknown.
- work backwards, computing each unknown.

Time Complexity → $O(nB^2)$

better than $O(n^3)$ in Gaussian Elimination.

Pseudo code:

```
for (int i=0 ; i<n-1; i++) {
    for (int j=i+1 ; j<min (i+B, n); j++) {
        factor = A[j][i] / A[i][i];
        for (int k=i; k<min (i+B, n); k++) {
            A[j][k] -= factor * A[i][k];
        }
        d[j] -= factor * d[i];
    }
}
x[n-1] = d[n-1] / A[n-1][n-1];
for (int i=n-2 ; i>=0 ; i--) {
    sum=0
    for (int j= i+1 ; j< min (i+B, n); j++) {
        sum += A[i][j] * x[j];
    }
    x[i] = (d[i] - sum) A[i][i];
}
```

2) $A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}$

$R_2 \to R_2 + R_1$ $\quad R_4 \to R_4 + R_1$

$R_3 \to R_3 + R_1$ $\quad R_5 \to R_5 + R_1$

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & -1 & 1 & 0 & 2 \\ 0 & -1 & -1 & 1 & 2 \\ 0 & -1 & -1 & -1 & 2 \end{bmatrix}$$

row = 2 has max element (partial pivoting)

$R_3 \to R_3 + R_2$ $\quad R_4 \to R_4 + R_2$

$R_5 \to R_5 + R_2$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & -1 & 1 & 4 \\ 0 & 0 & -1 & -1 & 4 \end{bmatrix}$$

$R_4 \to R_4 + R_3$ $\quad R_5 \to R_5 + R_3$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & -1 & 8 \end{bmatrix}$$

$R_5 \to R_5 + R_4$

$$U = A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 16 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\text{Growth factor} = \frac{\max |u_{ij}|}{\max |a_{ij}|}$$

$$= \frac{16}{1} = 2^4.$$