

Foundation of Computational Fluid Dynamics
Dr. S. Vengadesan
Department of Applied Mechanics
Indian Institute of Technology, Madras

Lecture - 34

It is my pleasure to welcome you again to this course on CFD. Last class, we started discussing about matrix inversion procedure; we listed three direct inversion procedure gauss elimination, TDMA and L U decomposition. In last class, we did in detail gauss elimination, and today's class we see in particular tridiagonal matrix algorithm otherwise TDMA.

(Refer Slide Time: 00:48)


Tridiagonal System of Equations - TDMA

- In Gauss elimination, bulk effort is spent on the forward elimination.
- For some combination of discretization procedure, coefficient matrix results in tri-diagonal form.
- Simplified form of Gauss elimination procedure is Thomas TDMA.
- Only one element needs to be eliminated from each row during the forward elimination process.
- When the algorithm reaches the last row, the last unknown becomes known.

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

Super-diagonal

Sub-diagonal diagonal



We know in Gauss elimination, there are two steps forward elimination and backward substitution, bulk of the time consume in forward elimination because in the backward substitution and it becomes completely upper triangular and we start from the last row in substitute and get the unknown. Hence most of the time is appear to be spend in the forward elimination process. We also learned different discretization procedure for example, central differencing, forward differencing, backward differencing pure up ending, quick and combination. And whatever the method you follow for some combination, the coefficient matrix will result in what is known as a tri-diagonal form. And we have already seen in previous class, diagonal matrix, tridiagonal matrix and penta diagonal matrix. The tridiagonal matrix structure is displayed here again. So, this

will have one main diagonal, which is marked here in red colour, and immediately above is a super diagonal and immediately below is a sub diagonal.

So, you see in this matrix we have values restricted only to limited region of the matrix and you have for most of the places zeros, so obviously, matrix of this nature applying Gauss elimination is not economical. So, we need to find some alternative way where you do not need to handling zeros, hence reduce computational time substantially. So, you follow what is known as a simplified form of Gauss elimination procedure, it was proposed by Thomas, just called Thomas algorithm for tridiagonal matrix; in short form, it is called TDMA. Only one element needs to be eliminated in the forward elimination process then when the algorithm reaches the last row, you have an equation with only one unknown on the left side and the known on the right side. So, you are immediately able to get the unknown, then we follow what is known as a backward substitution to get all the unknowns.


(Refer Slide Time: 03:15)

Tridiagonal System of Equations

- When ordinary differential equations are finite differenced, for example with CDS approximation, the resulting algebraic equations have an especially simple structure. Each equation contains only the variables at its own node and its immediate left and right neighbors.
- Consider a system of N linear, simultaneous algebraic equations with N unknowns $u_1, u_2, u_3, \dots, u_N$ given in the form below:

$$\begin{aligned}
 b_1 u_1 + c_1 u_2 &= d_1 \\
 a_2 u_1 + b_2 u_2 + c_2 u_3 &= d_2 \\
 a_3 u_2 + b_3 u_3 + c_3 u_4 &= d_3 \\
 &\dots \\
 a_{i-1} u_{i-1} + b_i u_i + c_i u_{i+1} &= d_i \\
 &\dots \\
 a_{N-1} u_{N-2} + b_{N-1} u_{N-1} + c_{N-1} u_N &= d_{N-1} \\
 a_N u_{N-1} + b_N u_N &= d_N
 \end{aligned}$$

$$\begin{bmatrix}
 b_1 & c_1 & 0 & 0 & 0 & 0 \\
 a_2 & b_2 & c_2 & 0 & 0 & 0 \\
 0 & a_3 & b_3 & c_3 & 0 & 0 \\
 & & & & & \\
 & & & & & \\
 0 & 0 & 0 & a_{N-1} & b_{N-1} & c_{N-1} \\
 0 & 0 & 0 & 0 & a_N & b_N
 \end{bmatrix}
 \begin{bmatrix}
 u_1 \\
 u_2 \\
 u_3 \\
 \dots \\
 u_{N-1} \\
 u_N
 \end{bmatrix}
 =
 \begin{bmatrix}
 d_1 \\
 d_2 \\
 d_3 \\
 \dots \\
 d_{N-1} \\
 d_N
 \end{bmatrix}$$



We will see TDMA little more detail now. So, when ordinary differential equations or partial differential equation or finite difference say in this case central different scheme, we know in central differences scheme we have node of interest and one more on either side left as well as right. And if you apply CDS scheme for the full domain, then the resulting algebraic equation will have a simple structure and each equation will have variables at its own nodes and immediate left as well as right, such a system linear

equations with n rows is given here. So, $b_1 u_1 + c_1 u_2 = d_1$; and we go to the next row $a_2 u_1 + b_2 u_2 + c_2 u_3 = d_2$, and all the way up to the n th row, $a_{n-1} u_{n-1} + b_n u_n = d_n$. Now in the first row, we have only two contributions that is because you have on the left side boundary condition and boundary condition is accounted as a source term and that is added and you get d_1 .

Similarly, the last row again has only two contribution, there was for the last row, the right side is again boundary condition and that is appearing as a source term and added to know value d_N . Now the same algebraic question in the form of matrix it is shown here. So, $b_1 c_1$, remaining all zeros; similarly, all the way up to last row $a_n b_n$ this is a coefficient matrix multiplying the unknown column vector, you want to u_1 to u_N on both on left side then on the right side you have a known column vector.


(Refer Slide Time: 05:23)

Tridiagonal System of Equations

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & 0 & 0 & 0 & a_N & b_N \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \dots \\ d_{N-1} \\ d_N \end{bmatrix}$$

Solution step involved are forward elimination and backward substitution as in Gauss elimination.

Recursive relations are

$$\begin{aligned}
 u_N &= q_N & p_1 &= b_1; \quad q_1 = \frac{d_1}{p_1} \\
 u_i &= q_i - \frac{c_i u_{i+1}}{p_i}, \quad i = N-1, N-2, \dots, 1 & p_i &= b_i - \frac{a_i c_{i+1}}{p_{i+1}}, \quad i = 2, 3, \dots, N \\
 & & q_i &= \frac{d_i - a_i q_{i+1}}{p_i}, \quad i = 2, 3, \dots, N
 \end{aligned}$$


We repeat that again here and to solve this, we follow same procedure as Gauss elimination procedure that is you have a forward elimination and backward substitution; only thing it is slightly modify for this matrix structure. And after working out, you are able to set what is known as a recursive relationship and that is what is shown here. So, u_n which is the last unknown is equated to q_n , and you go in the reverse, because now it is a backward substitution; $u_i = q_i - \frac{c_i u_{i+1}}{p_i}$, where i goes from $n-1$ in the reverse order all the way to 1, so this is what is a backward substitution. Now we define q and p as given here; p_1 is equal to b_1 , q_1 is equal to d_1 by p_1 and then this is

for the first for the remaining from 2 to N, q and p are defined as shown here. Now this relationship is already worked out, what I am showing here is only final expression, and this is in the form of a recursive relationship.

(Refer Slide Time: 06:42)

```

TDMA – Sample code listing

The equations are numbered from 1 though N.      C** Compute the solution vector X**
A - Sub-diagonal; B-Main diagonal; C-Super-    X(N) = Q(N)
diagonal;                                       N1 = N - 1
D - RHS vector; X-Computed solution           DO 20 K = 1, N1
                                              J = N-K
Subroutine TDMA (I, N, A, B, C, D, X)          X(J) = Q (J) - (C(J)*X(J+1))/ P(J)
Dimension A(I), B(I), C(I), D(I), X(I), P(100), Q(100)
C** Compute intermediate arrays P and Q **    RETURN
P (I) = B(I)                                  END
Q (I) = D(I) / Q(I)
I1 = I + 1
DO 10 J = I1, N
P (J) = B (J) - A(J)*C(J-1)/P(J-1)
Q (J) = (D(J) - A(J) *Q (J-1))/P (J)

```

What is shown here is a sample FORTRAN code listing to achieve what is known as a TDMA. So, equations are numbered from 1 to N; one special point to be noted here is matrix has values only along three diagonals - main diagonal, sub diagonal and super diagonal, remaining elements are all zero. So, we do not store the matrix in full form; we only store main diagonal, super diagonal and sub diagonal as independent matrix and that is what is A, B and C. So, N, A for example, we will have all elements belonging to the sub diagonal; B will have element belong to the main diagonal, and C will have the element belong to the super diagonal. So, they are independently store that way you do not store full Matrix because the full matrix are zeros and it is expensive in terms of memory as well as handling those zeros that is the specialty of this TDMA procedure.

And we also noticed we have one more column vector on the right side, the known column vector and that is stored in D and X is unknown column vector. So, we call subroutine TDMA with arguments and you define A, B, C, D and we defined two new variables P and Q related to the recursive relationship. So, we follow the formula and write this FORTRAN listing, this will result in some form of forward elimination. And the next step is the backward substitution, there you get solution better itself. So, the last

row is one that we can find first, so X of N is equal to Q of N then we do in the reverse N-1 equal to N minus I, we go in the reverse and find complete column vector.

(Refer Slide Time: 08:48)

```

int main()
{
    double *a, *b, *c, *d, *x;    //a, b, c are the diagonals
    int n, i;                    // n is the number of rows

    //modify the first-row coefficients
    c[0]=c[0]/b[0];
    d[0]=d[0]/b[0];

    for(i=1; i<n-1; i++)
    {
        temp = b[i]-a[i]*c[i-1];
        c[i] = c[i]/temp;
        d[i] = (d[i] - a[i]*d[i-1])/temp;
    }
    d[n] = (d[n]-a[n]*d[n-1])/(b[n]-a[n]*c[n-1]);

    //Back substitution
    x[n] = d[n];
    for(i=n-2; i>=-1; i--)
    {
        x[i] = d[i] - c[i]*x[i+1];
    }
    return 0;
}

```

We also have the similar simple code listing, but written in C language as shown here. So, double star a, star b, star c as we did before a, b, c are diagonals corresponding to main diagonal sub diagonal and super diagonal; and n is a number of rows that is required to be solved. Then you modify so in this instead of P and Q in the previous slide, here we defined some other quantity, you calculate d then substitute in the backward substitution form, you get the answer for unknown column vector.

(Refer Slide Time: 09:32)

TDMA - Example

Problem statement: Solve one dimensional diffusion equation in the domain $0 \leq x \leq 1$. BC: $u(0,t) = 0$; $u(1,t) = 0$ for all $t \geq 0$; $u(x,0) = \sin \pi x$. Use CN scheme.

Consider 1-D unsteady diffusion equation $\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0$

FTCS
$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{(u_{i-1}^n - 2u_i^n + u_{i+1}^n)}{\Delta x^2}$$

Crank-Nicholson Method

$$\frac{1}{\Delta t} (u_i^{n+1} - u_i^n) = \frac{\alpha}{2(\Delta x)^2} [(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}) + (u_{i+1}^n - 2u_i^n + u_{i-1}^n)]$$

$$-ru_{i-1}^{n+1} + 2(1+r)u_i^{n+1} - ru_{i+1}^{n+1} = -ru_{i-1}^n + 2(1+r)u_i^n - ru_{i+1}^n$$

where $r = \frac{\alpha \Delta t}{(\Delta x)^2}$

We will try to take an example problem and explain TDMA procedure. Problem statement - solve one-dimensional diffusion equation in the domain x going from 0 to 1 with the boundary condition $u(x,0) = \sin \pi x$ and $u(0,t) = 0$ and $u(1,t) = 0$ for all time greater than or equal to 0. And at initial condition the time is equal to zero, we have boundary condition $u(x,0) = \sin \pi x$. So, this is a initial condition, and these are all boundary condition applied at x equal to 0, at x is equal to 1 for t greater than or equal to 0. And specifically it is given follow Crank Nicholson scheme C-N scheme. We already learned this before; we will repeat it here again. So, consider one dimensional unsteady diffusion equation $\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0$. We know what is FTCS, that is forward in time central in space. So, we have forward in time for the time derivative and central in space for second order special derivative.

And this is explicit scheme in the sense all the variables are known from the previous time level that is what is given in the superscript as n . The current value to be determine is given the superscript $n + 1$ and that is only one quantity appearing for entire equations; remaining all from previous time level, and in terms of schematic that is shown here. So, $n + 1$ level value required at $n + 1$ level is determined from value is already found at n th level. In this problem specifically it is Crank-Nicholson scheme. So, for the same governing equation, we write down Crank-Nicholson scheme and that is shown here. So, the left hand side you have the time derivative term that is same forward

in time; the difference is only for the special dou squared you by dou x square that is written in the form of Crank-Nicholson scheme. We know Crank Nicholson scheme is a semi implicit scheme that is it is half explicit and half implicit and you are able to identify that here.

So, we have one set of term with superscript n plus 1, we have another set of term for the superscript n and there is weightage of that is what is in the denominator here two, corresponding molecule is schematics is given here. So, n plus 1 value is determined with the values from n plus 1 as well as values known from nth level. Now we can rewrite this specifically for the sake of writing code r is a new variable defined as alpha delta t by delta x square and all this n plus 1 values are to be determined. So, we take all of them on the left hand side and all quantity with the superscript n is known from the previous time level, and they are taken to the right hand side. This equation needs to be solved, when you apply Crank-Nicholson scheme.

(Refer Slide Time: 13:16)

TDMA - Example

Computational domain

Assume $\Delta x = 0.25$; $r = \frac{\alpha \Delta t}{\Delta x^2}$

For the sake of simplicity, we assume Δt and α in such a way $r = 1.0$.

$$\frac{1}{\Delta t} (u_i^{n+1} - u_i^n) = \frac{\alpha}{2(\Delta x)^2} [(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}) + (u_{i+1}^n - 2u_i^n + u_{i-1}^n)]$$

$$4u_2 - u_1 = 1$$

$$-u_1 + 4u_2 - u_3 = 1.42$$

$$-u_2 + 4u_3 = 1$$

$$\begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1.42 \\ 1 \end{bmatrix}$$

Now, if put this form of a matrix, so in the last slide, for the example problem, one-dimensional diffusion equation, we defined discretized equation; now we define computational domain as shown here. It is going from x is equal to zero on the left side to x is equal to one on the right side. We take four spacing otherwise three grid points. So, at one, two, and three and they are equally space on delta x happened to be 0.25. On the left side, boundary condition is defined as well as on the right side of boundary

condition is defined. For the sake of simplicity, we assume delta t and alpha in such a way of value r equal to one and we already define r to be alpha delta t by delta x square.

We rewrite the discretized equation here again. Now we use the value r and apply the discretized questions at every nodal point, and we get algebraic equation as shown here. As you can observe, the first row has influence of left side boundary; last row has influence of right side boundary. Now the same algebraic question is written in the form of matrix as shown here that is 4 minus 1 0 minus 1 4 minus 1 0 minus 1 4 the coefficient matrix and then one unknown column vector; on the right side, you have a known column vector.

(Refer Slide Time: 14:53)

TDMA - Example

$$p_1 = b_1; \quad q_1 = \frac{d_1}{p_1}$$

$$p_i = b_i - \frac{a_i c_{i-1}}{p_{i-1}}, \quad i = 2, 3, \dots, N$$

$$q_i = \frac{d_i - a_i q_{i-1}}{p_i}, \quad i = 2, 3, \dots, N$$

$$p_1 = b_1 = 4; \quad q_1 = \frac{d_1}{p_1} = \frac{1}{4} = 0.25$$

$$p_2 = b_2 - \frac{a_2 c_1}{p_1} = 4 - \frac{(-1)(-1)}{4} = 4 - \frac{1}{4} = 3.75$$

$$q_2 = \frac{d_2 - a_2 q_1}{p_2} = \frac{1.42 - (-1)(0.25)}{3.75} = \frac{1.67}{3.75} = 0.445$$

$$p_3 = b_3 - \frac{a_3 c_2}{p_2} = (-1) - \frac{0(-1)}{3.75} = -1$$

$$q_3 = \frac{d_3 - a_3 q_2}{p_3} = \frac{1 - 0(0.445)}{(-1)} = \frac{1.67}{-1} = -1$$

$$u_N = q_N$$

$$u_i = q_i - \frac{c_i u_{i+1}}{p_i}, \quad i = N-1, N-2, \dots, 1$$

Here $N = 3, N-1 = 2, \dots$

$$u_3 = q_3 \Rightarrow u_3 = q_3 = -1$$

$$u_2 = q_2 - \frac{c_2 u_3}{p_2} = 0.445 - \frac{(-1)(-1)}{3.75} = 0.445 - 0.266 = 0.179$$

$$u_1 = q_1 - \frac{c_1 u_2}{p_1} = 0.25 - \frac{(-1)(0.179)}{4} = 0.25 + 0.04475 = 0.29475$$

Now we apply the recursive relationship, we define two quantities p and q. So, p 1 q 1 and p i, q i for i going from 2 to N, there were also defined. We try to apply for this example problem. So, we get p 1 equal to b 1 equal to 4; q 1 is equal to 0.25. And we calculate other values for example, p 2, q 2, p 3, q 3; once you find this can we do the backward substitution and that is also defined in terms of recursive relationship u N is equal to q N, and u i is equal to q i c i u i plus one p i. And it is backward substitution, so i is going from n minus 1, all the way to one. We substitute n is equal to 3 that is the last row and then n minus 1 is equal to 2. So, we get u N equal to q N which is otherwise u 3 equal to q 3 equal to minus 1, and we calculate for the remaining u 2 u 1 using this

relationship, and substituting appropriate values, and the values are shown here it is so easy in TDMA.


(Refer Slide Time: 16:10)

Tridiagonal System of Equations – Periodic BC

$$\begin{bmatrix}
 b_1 & c_1 & 0 & 0 & 0 & a_1 \\
 a_2 & b_2 & c_2 & 0 & 0 & 0 \\
 0 & a_1 & b_1 & c_1 & 0 & 0 \\
 \dots & \dots & \dots & \dots & \dots & \dots \\
 0 & 0 & 0 & a_{N-1} & b_{N-1} & c_{N-1} \\
 c_N & 0 & 0 & 0 & a_N & b_N
 \end{bmatrix}
 \begin{bmatrix}
 u_1 \\
 u_2 \\
 u_3 \\
 \dots \\
 u_{N-1} \\
 u_N
 \end{bmatrix}
 =
 \begin{bmatrix}
 d_1 \\
 d_2 \\
 d_3 \\
 \dots \\
 d_{N-1} \\
 d_N
 \end{bmatrix}$$

When periodic boundary conditions are imposed, there is a change in the matrix structure.

Sherman-Morrison formula – modified TDMA is used.




Now we have the TDMA for a simple system, but in some problem, you may have to force what is known as a periodic boundary condition. Periodic boundary condition means values are repeated from one boundary to the next boundaries and the domain you supposed to extend in the periodic way indefinitely. When such situation is there, we pose periodic boundary condition, the matrix tridiagonal matrix gets slightly modified as shown here. The main diagonal, sub diagonal, super diagonal they are same. Now in the first row, we have one more element or value added; and the last column as shown here a 1. Similarly, last value in the first column again gets alter by the periodic boundary condition and it is shown here as c N. The TDMA procedure with just now explained is suitable for other boundary condition and it is slightly modified as Sherman-Morrison formula, we are not going to discuss that here and still we can use it TDMA procedure we have explained.

(Refer Slide Time: 17:32)

Tridiagonal System of Equations - TDMA

- It is easily programmable.
- In practice each diagonal is stored as column. Hence the full matrix which contains zero element is not stored.
- Number of operations is proportional to N – the number of unknowns.
- Saves lot of computational time (no handling of zero) and memory requirement is less.
- Successful algorithm.
- Penta- (For ex. ADI, Operator splitting), Septa- diagonal matrices are reduced to tri-diagonal and then TDMA is applied.



So, overall as you observe through example problem and also through recursive relationship, we understand it is very easy to program. And in practice, you do not actually have full coefficient matrix stored, only diagnosis are stored as a separate column vector. Number of operation because we are not handling with zeros, number of operation when compared to the Gauss elimination procedure is substantially reduced, now it is only of order n . It saves lot of computational time, because we are not handling zeros; memory requirement is also less. It is one of the successful algorithm and it is used by many researches for their own calculation.

Cases where you end up getting penta diagonal matrix, for example, we learned before what is known as alternating direction implicit, alternating direction implicit was a reduced form of the penta diagonal matrix solution procedure, where the original penta diagonal matrix is reduced into two tridiagonal matrix by having a method of explicit in x -direction, implicit in y -direction and then next stage implicit in x -direction, explicit in y -direction; you learned that before we will see that again. So, by that procedure, the original penta diagonal matrix its reduced tridiagonal matrix; similarly, there is another operation called operator splitting that also results in reducing original matrix into tridiagonal matrix. There are procedures available to convert penta diagonal matrix and septa diagonal matrix into tridiagonal matrix once it is reduced then we can apply TDMA procedure to get the solutions.


(Refer Slide Time: 19:33)

LU Decomposition

LU Decomposition: A is a full matrix, i.e. no zero value element.
In LU Decomposition, the matrix A is decomposed into two matrices.
L – Lower Triangular and U – Upper Triangular, matrices.

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & \dots & 0 \\ l_{21} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & \dots & 1 \end{bmatrix} \quad U = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

$A = LU$



We have listed three direct methods one is Gauss elimination, second is a TDMA, third one is L U decomposition. In L U decomposition, the full matrix A is split or decomposed into two as A is equal to L into U, where the L stands for lower triangular matrix and U stands for upper triangular Matrix and that is in the form of matrix that is shown A - full original coefficient matrix is decomposed into L Matrix and U matrix. So, in this class, we have seen in detail tridiagonal matrix algorithm procedure, we also had an example problem. We listed advantages associated with the TDMA procedure. And in the next class, we are going to see details about what is known as L U decomposition again with listing an example problem.

Thank you.