ENGR 695 Advanced Topics in Engineering Mathematics Fall 2024

Lecture Outline for Wednesday, Dec. 4, 2024

1. Crank-Nicholson (often spelled Crank-Nicolson) FD method applied to heat equation (continued)

$$c\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}$$

   a. Implicit update equations:

Interior points: $u_{i-1,j+1} - \alpha u_{i,j+1} + u_{i+1,j+1} = -u_{i-1,j} + \beta u_{i,j} - u_{i+1,j}$

Boundary at $x = a$: $-\alpha u_{2,j+1} + u_{3,j+1} = -u_{a,j+1} - u_{a,j} + \beta u_{2,j} - u_{3,j}$

Boundary at $x = b$: $u_{N_x-2,j+1} - \alpha u_{N_x-1,j+1} = -u_{N_x-2,j} + \beta u_{N_x-1,j} - u_{b,j} - u_{b,j+1}$,

where $\alpha = 2 + \dfrac{2\Delta x^2}{c\Delta t}$ and $\beta = 2 - \dfrac{2\Delta x^2}{c\Delta t}$.

   b. Update equations can be expressed as an $(N_x - 2) \times (N_x - 2)$ system of equations:

$$\begin{bmatrix} -\alpha & 1 & 0 & 0 & \cdots & 0 \\ 1 & -\alpha & 1 & 0 & \cdots & 0 \\ 0 & 1 & -\alpha & 1 & & \vdots \\ \vdots & & & \ddots & & 0 \\ 0 & \cdots & 0 & 1 & -\alpha & 1 \\ 0 & \cdots & 0 & 0 & 1 & -\alpha \end{bmatrix} \begin{bmatrix} u_{2,j+1} \\ u_{3,j+1} \\ u_{4,j+1} \\ \vdots \\ u_{N_x-2,j+1} \\ u_{N_x-1,j+1} \end{bmatrix} = \begin{bmatrix} -u_{a,j+1} - u_{a,j} + \beta u_{2,j} - u_{3,j} \\ -u_{2,j} + \beta u_{3,j} - u_{4,j} \\ -u_{3,j} + \beta u_{4,j} - u_{5,j} \\ \vdots \\ -u_{N_x-3,j} + \beta u_{N_x-2,j} - u_{N_x-1,j} \\ -u_{N_x-2,j} + \beta u_{N_x-1,j} - u_{b,j} - u_{b,j+1} \end{bmatrix}$$

Furthermore, the right-hand side can be expressed in matrix form as

$$\begin{bmatrix} -u_{a,j+1} - u_{a,j} + \beta u_{2,j} - u_{3,j} \\ -u_{2,j} + \beta u_{3,j} - u_{4,j} \\ -u_{3,j} + \beta u_{4,j} - u_{5,j} \\ \vdots \\ -u_{N_x-3,j} + \beta u_{N_x-2,j} - u_{N_x-1,j} \\ -u_{N_x-2,j} + \beta u_{N_x-1,j} - u_{b,j} - u_{b,j+1} \end{bmatrix} = \begin{bmatrix} \beta & -1 & 0 & 0 & \cdots & 0 \\ -1 & \beta & -1 & 0 & \cdots & 0 \\ 0 & -1 & \beta & -1 & & \vdots \\ \vdots & & & \ddots & & 0 \\ 0 & \cdots & 0 & -1 & \beta & -1 \\ 0 & \cdots & 0 & 0 & -1 & \beta \end{bmatrix} \begin{bmatrix} u_{2,j} \\ u_{3,j} \\ u_{4,j} \\ \vdots \\ u_{N_x-2,j} \\ u_{N_x-1,j} \end{bmatrix} + \begin{bmatrix} -u_{a,j+1} - u_{a,j} \\ 0 \\ 0 \\ \vdots \\ 0 \\ -u_{b,j} - u_{b,j+1} \end{bmatrix}.$$

(*continued on next page*)

c. The matrix equation can be expressed in more compact form as

$$A\mathbf{u}_{j+1} = B\mathbf{u}_j + \mathbf{c} \quad \rightarrow \quad \mathbf{u}_{j+1} = A^{-1}B\mathbf{u}_j + A^{-1}\mathbf{c},$$

where
$$A = \begin{bmatrix} -\alpha & 1 & 0 & 0 & \cdots & 0 \\ 1 & -\alpha & 1 & 0 & \cdots & 0 \\ 0 & 1 & -\alpha & 1 & & \vdots \\ \vdots & & & \ddots & & 0 \\ 0 & \cdots & 0 & 1 & -\alpha & 1 \\ 0 & \cdots & 0 & 0 & 1 & -\alpha \end{bmatrix}, \quad \mathbf{u}_{j+1} = \begin{bmatrix} u_{2,j+1} \\ u_{3,j+1} \\ u_{4,j+1} \\ \vdots \\ u_{N_x-2,j+1} \\ u_{N_x-1,j+1} \end{bmatrix}, \quad \mathbf{u}_j = \begin{bmatrix} u_{2,j} \\ u_{3,j} \\ u_{4,j} \\ \vdots \\ u_{N_x-2,j} \\ u_{N_x-1,j} \end{bmatrix},$$

$$B = \begin{bmatrix} \beta & -1 & 0 & 0 & \cdots & 0 \\ -1 & \beta & -1 & 0 & \cdots & 0 \\ 0 & -1 & \beta & -1 & & \vdots \\ \vdots & & & \ddots & & 0 \\ 0 & \cdots & 0 & -1 & \beta & -1 \\ 0 & \cdots & 0 & 0 & -1 & \beta \end{bmatrix}, \quad \text{and} \quad \mathbf{c} = \begin{bmatrix} -u_{a,j+1} - u_{a,j} \\ 0 \\ 0 \\ \vdots \\ 0 \\ -u_{b,j} - u_{b,j+1} \end{bmatrix}.$$

d. At each iteration (time step), evaluate the matrix-vector update equation

$$\mathbf{u}_{j+1} = A^{-1}B\mathbf{u}_j + A^{-1}\mathbf{c}$$

Matrices A and B (and vector $\mathbf{c}$ as well if the boundary conditions are not time varying) do not change with time, so $A^{-1}B$ and $A^{-1}\mathbf{c}$ can be computed once and stored before the algorithm begins. If boundary conditions are time varying, then $A^{-1}\mathbf{c}$ must be evaluated at each time step, but A^{-1} can be precalculated.

e. Computational considerations:

i. Matrix multiplication is time consuming, but at least Gaussian elimination is not required. Parallel processing might speed up computation.

ii. Matrix A is tridiagonal and positive definite (i.e., the scalar result of $\mathbf{x}^T A\mathbf{x}$ is positive for any nonzero real column vector $\mathbf{x}$). Positive definite matrices have some desirable properties; consequently, efficient routines are available to compute their inverses. Parallel processing might speed up computation.

iii. The Crank-Nicholson method is an implicit method → no restriction on size of Δt with regard to stability. The method is unconditionally stable when applied to the heat equation.

iv. Accuracy is second order in space and time, which means that errors are proportional to Δx^2 and Δt^2. Accuracy is improved if Δx, Δt, or both are decreased, but computation time increases.