Chapter 4

Finite Difference Method for Parabolic Equations

Last Session Contents:

1) **Numerical Stability**
2) **Convergence**
3) **Tridiagonal Matrix Algorithm**
4) **Implicit Methods**
5) **Boundary Treatment for Derivative BCs**
6) **Keller-Box Method**

Numerical Stability

- A concept only defined in iterative problems.
- It necessitates:
  Errors, of any type, should not grow in an iterative process.
- Somewhat more difficult than the study of consistency!
- For non-linear problems, the necessary condition for stability is that linear stability analysis of them must be stable.
- We will discuss it in detailed later on!
- Now, let's only take a brief look at "stability of Dufort- Frankel and Explicit scheme"

Numerical Stability–In Practice

1. Recall the discretized equation of heat conduction using Dufort-Frankel:

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

- This scheme is **unconditionally stable**.

2. Explicit Method is stable if:

$$r = [\frac{\Delta t}{(\Delta x)^2}] \le \frac{1}{2}$$ It limits time step size!

3. Central Difference in time:

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

- This scheme is **Unconditionally Unstable**.

Numerical Stability–Physical Interpretation

Sometimes numerical instability can be seen as physically unacceptable results!

Let's consider explicit scheme for discretization of heat equation:

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

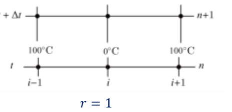
$$r = \frac{\Delta t}{(\Delta x)^2}$$

Assume that at $t = n$ we have: $u_i^n = 0$ and $u_{i+1}^n = u_{i-1}^n = 100℃$

In this case, if $r > \frac{1}{2}$ temperature at point i will exceed the temperature of two nearby points!

UNACCEPTABLE!?

The maximum expected temperature must be 100℃
However, when $r = 1$ it becomes $T_i^{n+1} = 200℃$!



$r = 1$

Convergence

- Generally speaking:
  A Consistent and Stable Scheme will converge!
- **Convergence:**
  Solving discretized equation of a PDE subjected to similar boundary and initial conditions will converge to the exact solution of that PDE provided that grid size is chosen to be infinitely small.
- **Finite Difference Equation** is converging if:

$$\lim_{h,k\to 0} |U_i^n - u_i^n| = 0 \qquad (x_i, t_n) \in \Omega$$

where $u_i^n = u(x_i, t_n)$

- **Lax's Equivalence Theorem**:
  For a linear well-posed problem, with correct boundary condition, and a Finite Difference Approximation of it, Consistency and Stability are necessary and sufficient conditions to provide the convergence!

Tridiagonal Systems of Equations

Using numerical methods, the governing PDEs convert to system of algebraic equations as follow:

Tx=b

Large tridiagonal systems arise naturally in a number of problems, especially in the numerical solution of differential equations by implicit methods.

$$\mathbf{T} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

When a large system of linear algebraic equations has a special pattern, it is usually worthwhile to develop special methods for that unique pattern.

Tridiagonal Systems of Equations

One algorithm that deserves special attention is the algorithm for tridiagonal matrices, often referred to as the Thomas (1949) algorithm.

$$\mathbf{T} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

Row 2: $R_2 - (a_{21}/a_{11})R_1 \qquad [0 \quad a_{22} - (a_{21}/a_{11})a_{12} \quad a_{23} \quad 0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad 0]$

only a_{32} in column 2 must be eliminated from row 3
only a_{43} in column 3 must be eliminated from row 4, etc.
The eliminated element itself does not need to be calculated.
storing the elimination multipliers, $\text{em} = (a_{21}/a_{11})$ etc, in place of the eliminated

Tridiagonal Systems of Equations

$$\mathbf{T} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

Hint:
Only the diagonal element in each row is affected by the elimination.
Elimination in rows 2 to n is accomplished as follows:

$$a_{i,i} = a_{i,i} - (a_{i,i-1}/a_{i-1,i-1})a_{i-1,i} \qquad (i = 2, \ldots, n)$$

Thus, the elimination step involves only **2n** multiplicative operations to place T in upper triangular form.

Tridiagonal Systems of Equations

Subsequent elements of the **b** vector are changed in a similar manner.

$$b_2 = b_2 - (a_{21}/a_{11})b_1$$

$\text{em} = (a_{21}/a_{11})$ is already calculated. Thus, the total process of elimination, including the operation on the b vector, requires only **3n** multiplicative operations.

Hint:
The nxn tridiagonal matrix **T** can be stored as an nx3 matrix **A'** since there is no need to store the zeros.

$$\mathbf{A}' = \begin{bmatrix} - & a'_{1,2} & a'_{1,3} \\ a'_{2,1} & a'_{2,2} & a'_{2,3} \\ a'_{3,1} & a'_{3,2} & a'_{3,3} \\ \cdots & \cdots & \cdots \\ a'_{n-1,1} & a'_{n-1,2} & a'_{n-1,3} \\ a'_{n,1} & a'_{n,2} & - \end{bmatrix}$$

Column1=Sub-diagonal elements of T
Column2=Diagonal elements of T
Column3=Super-diagonal elements of T

Example

$T'' - \alpha^2 T = -\alpha^2 T_a \qquad T(0.0) = 0.0 \qquad T(1.0) = 100.0$

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} + 0(\Delta x^2) - \alpha^2 T_i = -\alpha^2 T_a$$

$$T_{i-1} - (2 + \alpha^2\,\Delta x^2)T_i + T_{i+1} = -\alpha^2\,\Delta x^2\,T_a \qquad \begin{cases} \alpha = 4.0 \\ \Delta x = 0.125 \\ (2 + \alpha^2\,\Delta x^2) = 2.25 \end{cases}$$

$$\mathbf{A}' = \begin{bmatrix} - & -2.25 & 1.0 \\ 1.0 & -2.25 & 1.0 \\ 1.0 & -2.25 & 1.0 \\ 1.0 & -2.25 & 1.0 \\ 1.0 & -2.25 & 1.0 \\ 1.0 & -2.25 & 1.0 \\ 1.0 & -2.25 & - \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ -100.0 \end{bmatrix}$$

Example

$$\mathbf{A}' = \begin{bmatrix} - & -2.250000 & 1.0 \\ (-0.444444) & -1.805556 & 1.0 \\ (-0.553846) & -1.696154 & 1.0 \\ (-0.589569) & -1.660431 & 1.0 \\ (-0.602253) & -1.647747 & 1.0 \\ (-0.606889) & -1.643111 & 1.0 \\ (-0.608602) & -1.641398 & - \end{bmatrix} \quad \text{and} \quad \mathbf{b}' = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ -100.0 \end{bmatrix}$$

$$x_7 = b_7/a'_{7,2} = (-100)/(-1.641398) = 60.923667$$

$$x_6 = (b_6 - a'_{6,3}x_7)/a'_{6,2} = [0 - (1.0)(60.923667)]/(-1.643111)$$
$$= 37.078251$$

$$\mathbf{x} = \begin{bmatrix} 1.966751 \\ 4.425190 \\ 7.989926 \\ 13.552144 \\ 22.502398 \\ 37.078251 \\ 60.923667 \end{bmatrix}$$

Implicit Finite Difference Approximations

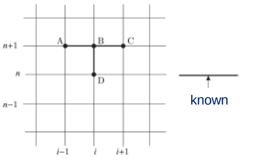
- **Backward Difference Scheme**

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

Considering $r = \frac{k}{h^2}$ we have:

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

or,



Grid Stencil

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

How to solve it?!

Implicit Finite Difference Approximations

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

- Assume that boundary values are zero at both ends.
- This tri-diagonal system can be solved by Thomas Algorithm.

$$\begin{bmatrix} (1+2r) & -r & & & \\ -r & (1+2r) & -r & & \\ & \ddots & \ddots & \ddots & \\ & & -r & (1+2r) & -r \\ & & & -r & (1+2r) \end{bmatrix} \begin{bmatrix} u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ \vdots \\ u_{imax-1}^{n+1} \end{bmatrix} = \begin{bmatrix} u_2^n \\ u_3^n \\ \vdots \\ \vdots \\ u_{imax-1}^n \end{bmatrix}$$

Note that:

- BTCS is unconditionally stable.
- Second order in space but first order in time!

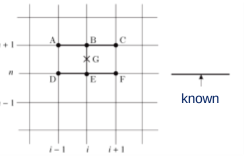
Implicit Finite Difference Approximations

- **Crank-Nikolson Scheme**

$$\frac{1}{\Delta t}\delta_t u_i^{n+\frac{1}{2}} = \frac{1}{2(\Delta x)^2}(\delta_x^2 u_i^{n+1} + \delta_x^2 u_i^n)$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{2(\Delta x)^2} \times$$

$$(u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1} + u_{i-1}^n - 2u_i^n + u_{i+1}^n)$$



Grid Stencil

Considering $r = \frac{k}{\Delta x^2}$ we have:

$$u_i^{n+1} - u_i^n = \frac{r}{2}(u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}) + \frac{r}{2}(u_{i-1}^n - 2u_i^n + u_{i+1}^n)$$

or,

$$\boxed{ru_{i-1}^{n+1} - 2(1+r)u_i^{n+1} + ru_{i+1}^{n+1} = -ru_{i-1}^n + 2(-1+r)u_i^n - ru_{i+1}^n \quad \textbf{Crank-Nikolson}}$$

Implicit Finite Difference Approximations

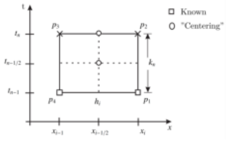
- **Keller-Box Scheme**

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

We can re-write this equation as:

$$\frac{\partial u}{\partial x} = P$$

$$\frac{\partial u}{\partial t} = \frac{\partial P}{\partial x}$$



Grid Stencil

Implicit Finite Difference Approximations

- **Keller-Box Scheme**

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

We can re-write this equation as:

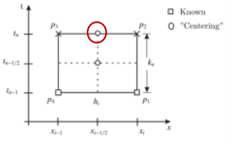
$$\boxed{\frac{\partial u}{\partial x} = P}$$

$$\frac{\partial u}{\partial t} = \frac{\partial P}{\partial x}$$

$$\frac{u_i^n - u_{i-1}^n}{h_i} = P_{i-\frac{1}{2}}^n$$

or,

$$\boxed{u_i^n - u_{i-1}^n - \frac{h_i}{2}(P_i^n + P_{i-1}^n) = 0}$$



Grid Stencil

Implicit Finite Difference Approximations

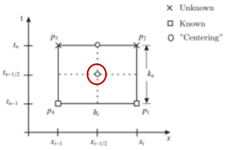
- **Keller-Box Scheme**

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

We can re-write this equation as:

$$\frac{\partial u}{\partial x} = P$$

$$\boxed{\frac{\partial u}{\partial t} = \frac{\partial P}{\partial x}} \rightarrow \frac{P_i^{n-\frac{1}{2}} - P_{i-1}^{n-\frac{1}{2}}}{h_i} = \frac{u_{i-\frac{1}{2}}^n - u_{i-\frac{1}{2}}^{n-1}}{k_n}$$



Grid Stencil

$$\frac{1}{2}\left[\frac{P_i^n - P_{i-1}^n}{h_i} + \frac{P_i^{n-1} - P_{i-1}^{n-1}}{h_i}\right] = \frac{1}{2}\left[\frac{u_i^n + u_{i-1}^n}{k_n} - \frac{u_i^{n-1} + u_{i-1}^{n-1}}{k_n}\right]$$

or,

$$\boxed{P_i^n - P_{i-1}^n - \frac{h_i}{k_n}(u_i^n + u_{i-1}^n) = \underbrace{P_{i-1}^{n-1} - P_i^{n-1} - \frac{h_i}{k_n}(u_i^{n-1} + u_{i-1}^{n-1})}_{R_{i-\frac{1}{2}}^{n-1}}}$$

Implicit Finite Difference Approximations

$$\begin{cases} u_i^n - u_{i-1}^n - \dfrac{h_i}{2}(P_i^n + P_{i-1}^n) = 0 \\ P_i^n - P_{i-1}^n - r_i^n(u_i^n + u_{i-1}^n) = R_{i-\frac{1}{2}}^{n-1} \end{cases}$$

$$\begin{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & -\frac{h_1}{2} \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 1 & -\frac{h_1}{2} \end{bmatrix} & & \\ \begin{bmatrix} -r_i^n & -1 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} -r_i^n & 1 \\ -1 & -\frac{h_2}{2} \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 1 & -\frac{h_2}{2} \end{bmatrix} & \\ & \ddots & & \ddots \\ & & \ddots & & \ddots \end{bmatrix} \begin{bmatrix} \begin{bmatrix} u_0 \\ P_0 \end{bmatrix} \\ \begin{bmatrix} u_1 \\ P_1 \end{bmatrix} \\ \begin{bmatrix} u_2 \\ P_2 \end{bmatrix} \\ \vdots \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} R_2 \\ 0 \end{bmatrix} \\ \vdots \end{bmatrix}$$

Implicit Finite Difference Approximations

- We can re-write the previous matrix as below where the elements are blocks itself.
- This matrix can be solved using block Thomas algorithm.
- Please note that this matrix should be constructed so that: $\det(B_0) \neq 0$

$$\begin{bmatrix} B_0 & C_0 & & & & \\ A_1 & B_1 & C_1 & & & \\ & A_2 & B_2 & C_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & & A_n & B_n \end{bmatrix} \begin{bmatrix} \vec{u}_0 \\ \vec{u}_1 \\ \vec{u}_2 \\ \vdots \\ \vdots \\ \vec{u}_n \end{bmatrix} = \begin{bmatrix} \vec{R}_0 \\ \vec{R}_1 \\ \vec{R}_2 \\ \vdots \\ \vdots \\ \vec{R}_n \end{bmatrix}$$

Implicit Finite Difference Approximations

- The main features of Keller Box Scheme

1. Only slightly more arithmetic to solve than the Crank-Nikolson method
2. Second order accurate with arbitrary (uniform) x and y spacing
3. Allows very rapid x variation
4. Allows easy programming of the solution of large numbers of coupled equation.

- Steps:

1. Reduce the Equations to a 1st – order system
2. Write difference equations using central differencing.
3. Linearize the resulting algebraic equation and write them in matrix-vector form
4. Solve the linear system by the block-tridiagonal elimination method

Implementation of Boundary Condition

- Implicit schemes mostly end up to this form:

$$A_i u_{i-1}^{n+1} + B_i u_i^{n+1} + C_i u_{i+1}^{n+1} = R_i^n = R_i(u_{i-1}^n, u_i^n, u_{i+1}^n)$$

$$u_1 = u_a, \qquad u_{imax} = u_b$$

- **1st Method:**
  Boundary conditions are considered in the equations and matrix-form equation is solved for $i = 2$ to $i = imax - 1$.

$$A_2 u_1^{n+1} + B_2 u_2^{n+1} + C_2 u_3^{n+1} = R_2^n \qquad i = 2$$

$$B_2 u_2^{n+1} + C_2 u_3^{n+1} = R_2^n - A_2 u_a^{n+1} \qquad \text{Considering BC at } a$$

$$A_{imax-1} u_{imax-2}^{n+1} + B_{imax-1} u_{imax-1}^{n+1} + C_{imax-1} u_{imax}^{n+1} = R_{imax-1}^n \qquad i = imax$$

$$A_{imax-1} u_{imax-2}^{n+1} + B_{imax-1} u_{imax-1}^{n+1} = R_{imax-1}^n - C_{imax-1} u_b^{n+1} \qquad \text{Considering BC at } imax$$

Implementation of Boundary Condition

- **1st Method:**
  Boundary conditions are considered in the equations and matrix-form equation is solved for $i = 2$ to $i = imax - 1$.

$$B_2 u_2^{n+1} + C_2 u_3^{n+1} = R_2^n - A_2 u_a^{n+1}$$

$$A_{imax-1} u_{imax-2}^{n+1} + B_{imax-1} u_{imax-1}^{n+1} = R_{imax-1}^n - C_{imax-1} u_b^{n+1}$$

$$\begin{bmatrix} B_2 & C_2 & & & \\ A_3 & B_3 & C_3 & & \\ & \ddots & \ddots & \ddots & \\ & & A_{I-2} & B_{I-2} & C_{I-2} \\ & & & A_{I-1} & B_{I-1} \end{bmatrix} \begin{bmatrix} u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ \vdots \\ u_{I-2}^{n+1} \\ u_{I-1}^{n+1} \end{bmatrix} = \begin{bmatrix} R_2^n - A_2 u_a^{n+1} \\ R_3^n \\ \vdots \\ \vdots \\ R_{I-2}^n \\ R_{I-1}^n - C_{I-1} u_b^{n+1} \end{bmatrix}$$

Implementation of Boundary Condition

- **2nd Method:**
  Let the computer do the calculations!

$$\begin{bmatrix} B_1 & C_1 & & & \\ A_2 & B_2 & C_2 & & \\ & \ddots & \ddots & \ddots & \\ & & A_{I-1} & B_{I-1} & C_{I-1} \\ & & & A_I & B_I \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_{I-1}^{n+1} \\ u_I^{n+1} \end{bmatrix} = \begin{bmatrix} u_a^{n+1} \\ R_2^n \\ \vdots \\ \vdots \\ R_{I-1}^n \\ u_b^{n+1} \end{bmatrix}$$

$$B_1 = B_I = 1 \qquad C_1 = A_I = 0$$

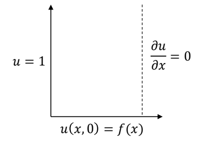
Note: a slight increase in computation cost, however, gives more flexibility in computer code!

Derivative Boundary Condition

- **1st Method:**
  Backward difference at the boundary

$$\frac{\partial u}{\partial x}\Big|_{i=N} = \frac{3u_N - 4u_{N-1} + u_{N-2}}{2\Delta x} = 0$$

$$\boxed{u_N = \frac{4u_{N-1} - u_{N-2}}{3}}$$



- **2st Method:**
  False boundary

$$\frac{\partial u}{\partial x}\Big|_{i=N} = \frac{u_{N+1} - u_{N-1}}{2\Delta x} = 0$$

$$\boxed{u_{N+1} = u_{N-1}}$$

Numerical Solution of Blasius Equation

- **Blasius Equation:**

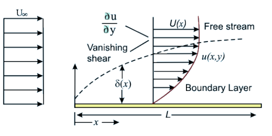
$$f'''(\eta) + \frac{\alpha}{2} f(\eta) f''(\eta) = 0 \qquad \eta = y\sqrt{\frac{u_\infty}{\alpha \nu x}}$$

α an arbitrary parameter

$$f(0) = f'(0) = 0, \qquad f'(\infty) = 1$$

- **Breaking it up to three first order equations:**

$$\frac{df}{d\eta} = u \qquad f(0) = 0$$
$$\frac{du}{d\eta} = v \qquad u(0) = 0 \qquad u(\infty) = 1$$
$$\frac{dv}{d\eta} = -\frac{\alpha}{2} f v$$



Numerical Solution of Blasius Equation

- **We discretize the equations in $\eta_{j-\frac{1}{2}}$**

$$\frac{f_j - f_{j-1}}{h_j} = u_{j-1/2} = \frac{1}{2}(u_j + u_{j-1})$$
$$\frac{u_j - u_{j-1}}{h_j} = v_{j-1/2} = \frac{1}{2}(v_j + v_{j-1})$$
$$\frac{v_j - v_{j-1}}{h_j} = -\frac{\alpha}{2}(fv)_{j-1/2} = -\alpha \frac{f_j v_j + f_{j-1} v_{j-1}}{4}$$

- **Newton Linearization**

These equations are non-linear, so, we have to linearize them.

$$f_k^{n+1} = f_k^n + \delta f_k^n$$
$$u_k^{n+1} = u_k^n + \delta u_k^n$$
$$v_k^{n+1} = v_k^n + \delta v_k^n$$

where n denotes the iteration number.

Note: we call the solution converged if $\delta(.)$ variables approach to zero!

Numerical Solution of Blasius Equation

- **Substituting these parameters into the first equations yields:**

$$f_j^n + \delta f_j^n - f_{j-1}^n - \delta f_{j-1}^n = \frac{h_j}{2}\left\{u_j^n + \delta u_j^n + u_{j-1}^n + \delta u_{j-1}^n\right\}$$
$$u_j^n + \delta u_j^n - u_{j-1}^n - \delta u_{j-1}^n = \frac{h_j}{2}\left\{v_j^n + \delta v_j^n + v_{j-1}^n + \delta v_{j-1}^n\right\}$$
$$v_j^n + \delta v_j^n - v_{j-1}^n - \delta v_{j-1}^n = -\frac{\alpha h_j}{4}\left\{(f_j^n + \delta f_j^n)(v_j^n + \delta v_j^n) + (f_{j-1}^n + \delta f_{j-1}^n)(v_{j-1}^n + \delta v_{j-1}^n)\right\}$$

We can rewrite it as:

$$\delta f_j^n - \delta f_{j-1}^n - \frac{h_j}{2}(\delta u_j^n + \delta u_{j-1}^n) = r_j^n \qquad r_j^n = f_{j-1}^n - f_j^n + h_j u_{j-1/2}^n$$
$$\delta u_j^n - \delta u_{j-1}^n - \frac{h_j}{2}(\delta v_j^n + \delta v_{j-1}^n) = t_j^n \qquad t_j^n = u_{j-1}^n - u_j^n + h_j v_{j-1/2}^n$$
$$(1 + \frac{\alpha h_j}{4} f_j^n)\delta v_j^n + (-1 + \frac{\alpha h_j}{4} f_{j-1}^n)\delta v_{j-1}^n + \frac{\alpha h_j}{4} v_j^n \delta f_j^n + \frac{\alpha h_j}{4} v_{j-1}^n \delta f_{j-1}^n = s_j^n \qquad s_j^n = v_{j-1}^n - v_j^n - \frac{\alpha h_j}{2}(fv)_{j-1/2}^n$$

Numerical Solution of Blasius Equation

- **Finally, it can be written in matrix form as:**

$$A_j^n \vec{\delta}_{j-1}^n + B_j^n \vec{\delta}_j^n + C_j^n \vec{\delta}_{j+1}^n = \vec{R}_j^n \qquad \vec{\delta}_j^n = [\delta f_j^n \quad \delta u_j^n \quad \delta v_j^n]^{\mathrm{T}}$$

$$\begin{bmatrix} B_1^n & C_1^n & & & \\ A_2^n & B_2^n & C_2^n & & \\ & \ddots & \ddots & \ddots & \\ & & A_{jmax-1}^n & B_{jmax-1}^n & C_{jmax-1}^n \\ & & & A_{jmax}^n & B_{jmax}^n \end{bmatrix} \begin{bmatrix} \vec{\delta}_1^n \\ \vec{\delta}_2^n \\ \vdots \\ \vec{\delta}_{jmax-1}^n \\ \vec{\delta}_{jmax}^n \end{bmatrix} = \begin{bmatrix} \vec{R}_1^n \\ \vec{R}_2^n \\ \vdots \\ \vec{R}_{jmax-1}^n \\ \vec{R}_{jmax}^n \end{bmatrix}$$

Note: This block-tridiagonal matrix can be solved using block Thomas elimination

Numerical Solution of Blasius Equation

- A, B, C and R blocks are as following:

$$A_j^n = \begin{bmatrix} -1 & -\frac{h_j}{2} & 0 \\ \frac{\alpha h_j}{4} v_{j-1}^n & 0 & -1 + \frac{\alpha h_j}{4} f_{j-1}^n \\ 0 & 0 & 0 \end{bmatrix} \quad 2 \le j \le jmax$$

$$B_j^n = \begin{bmatrix} 1 & -\frac{h_j}{2} & 0 \\ \frac{\alpha h_j}{4} v_j^n & 0 & 1 + \frac{\alpha h_j}{4} f_j^n \\ 0 & -1 & -\frac{h_{j+1}}{2} \end{bmatrix} \quad 2 \le j \le jmax - 1$$

$$C_j^n = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & -\frac{h_{j+1}}{2} \end{bmatrix} \quad 1 \le j \le jmax - 1$$

$$\vec{R}_j^n = \begin{bmatrix} r_j^n \\ s_j^n \\ t_{j+1}^n \end{bmatrix} \quad 2 \le j \le jmax - 1$$

- Boundary condition implementation also gives:

$$B_1^n = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & -\frac{h_2}{2} \end{bmatrix} \qquad B_{jmax}^n = \begin{bmatrix} 1 & -\frac{h_{jmax}}{2} & 0 \\ \frac{\alpha h_{jmax}}{4} v_{jmax}^n & 0 & 1 + \frac{\alpha h_{jmax}}{4} f_{jmax}^n \\ 0 & 1 & 0 \end{bmatrix}$$

$$\vec{R}_1^n = \begin{bmatrix} 0 \\ 0 \\ t_2^n \end{bmatrix} \qquad \vec{R}_{jmax}^n = \begin{bmatrix} r_{jmax}^n \\ s_{jmax}^n \\ 0 \end{bmatrix}$$