Finite Difference Methods

IPCST
Seoul National University

Modeling by FDM

- FDM for elliptic PDEs
  - Static models: Heat distribution, astronomy, electromagnetism, steady-state flow
- FDM for parabolic PDEs
  - Heat conduction, diffusion, option pricing
- FDM for hyperbolic PDEs
  - Advection, wave
- FDM for mixed PDEs
  - Transonic flows

Steady-state Solution of Parabolic PDE

- Ex.) $\partial_t u = D\triangle u + f(x)$
  - The steady-state solution of this equation is obtained by setting $\partial_t u = 0$.
    - $\triangle u = -f(x)/D$
  - This is an elliptic equation.

FDM for Elliptic PDEs

- **Stencil + linear algebra**
  - Stencil: 3-point (1-D), 5-point (2-D), 7-point (3-D), ……
  - Linear algebra: direct or iterative
- Iterative FDM
  - Unified formulation of 'stencil + linear algebra'
  - Better for huge sparse matrices
    - **Fixed point: Jacobi, Gauss-Seidel, SOR, ……**
    - Krylov solvers: steepest descent, conjugate gradient, Newton-Krylov, ……

FDM for 1-D Elliptic Model

- Poisson eq. $\triangle u = f(x) \;\rightarrow\; u_{xx} = f(x)$

  $\triangle u = f \;\rightarrow\; \boldsymbol{Au} = \boldsymbol{f} \;\; (\boldsymbol{A} = \triangle_h u)$

  – Truncation error at point x_i : $\boldsymbol{\tau}_i \approx (1/12)h^2 u_{xxxx}$
  – Matrix analysis can prove convergence with order 2.

Convergence

- Truncation error: $\boldsymbol{\tau} = \boldsymbol{A}\boldsymbol{u}_{\text{exact}} - \boldsymbol{A}\boldsymbol{u}_{\text{approx}}$
- Actual error: $\boldsymbol{e} = \boldsymbol{u}_{\text{exact}} - \boldsymbol{u}_{\text{approx}}$
- A numerical method is
  - Consistent if $||\boldsymbol{\tau}|| \to 0$ as $h \to 0$
  - Stable if $||\boldsymbol{A}^{-1}|| \leq$ constant ($\because \boldsymbol{A}^{-1}\boldsymbol{\tau} = \boldsymbol{e}$)
  - Convergent if $||\boldsymbol{e}|| \to 0$ as $h \to 0$
    - Global error $\to 0$ as $h \to 0$

✓ Convergence = stability + consistency

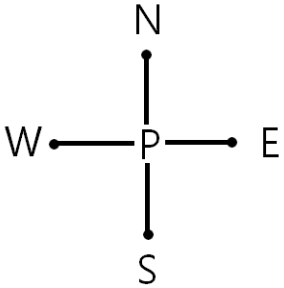
Five-point Stencil (2D)

- 5-point discrete Laplacian (2D)

$$\triangle_h u = (u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j})/h^2$$

  - Local truncation error

    $\boldsymbol{\tau}_{i,j} = (u_{xxxx} + u_{yyyy})h^2/12 + O(h^4)$



  - Convergent with order 2 for $\triangle u = f(x)$

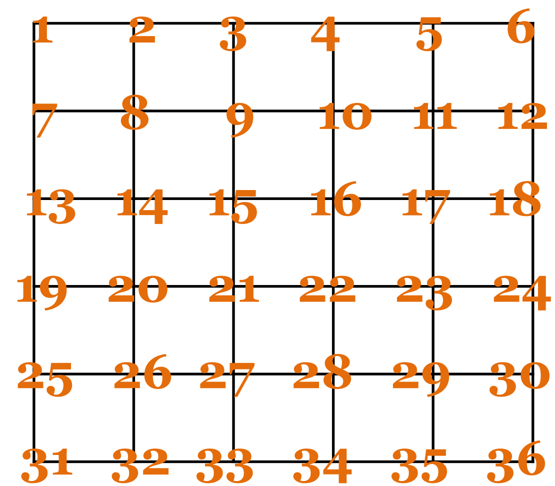
    $(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j})/h^2 = f_{i,j}$

Five-point Stencil (2D)

- Matrix representation
  - Usual ordering →
    - $\boldsymbol{A}$ (= $\triangle_h u$): $m^2 \times m^2$ matrix

$$\frac{1}{h^2}\left[\begin{array}{c|c|c|c} T & I & 0 & 0 \\ \hline I & T & \ddots & 0 \\ \hline 0 & \ddots & \ddots & I \\ \hline 0 & 0 & I & T \end{array}\right]$$

m = 6 case



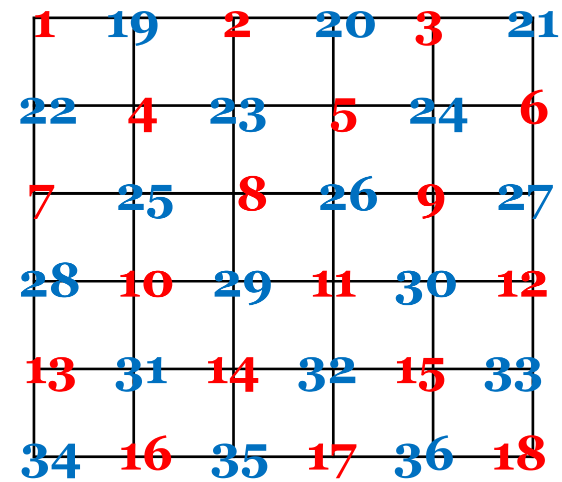
$$I = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix} \quad T = \begin{bmatrix} -4 & 1 & & 0 \\ 1 & -4 & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & 1 & -4 \end{bmatrix}$$

    - $\boldsymbol{T,I}$: $m \times m$ matrices

❖ Ill-conditioned for Krylov linear algebra solvers

Five-point Stencil (2D)

- Matrix representation
  - Alternative ordering →
    - ***A*** (= $\triangle_h u$): $m^2 \times m^2$ matrix



$$\frac{1}{h^2}\left[\begin{array}{c|c} D & H \\ \hline H^T & D \end{array}\right]$$

$$H = \begin{bmatrix} 1 & \cdot & 0 & \cdot & 1 & & & 0 \\ 1 & \ddots & & & & \ddots & & \\ \cdot & & & & & & & 1 \\ 0 & & \ddots & & \ddots & & & \cdot \\ \cdot & & & & & & & 0 \\ 1 & & & & \ddots & & \ddots & \\ & \ddots & & & & & & \cdot \\ 0 & & 1 & & 0 & & 1 & 1 \end{bmatrix}$$

    - ***D*** = -4***I***
    - ***D,H*** : $(m^2/2)\times(m^2/2)$ matrices

Five-point Stencil (2D)

$$u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j} = h^2 f_{i,j}$$

- Applying boundary conditions
  - Dirichlet B.C.: replacement by values
    - Ex.) $u(x,0) = g(x) \rightarrow u_{i,0} = g_i$

    → $u_{i-1,1} + u_{i+1,1} + u_{i,2} - 4u_{i,1} = h^2 f_{i,1} - g_i$
  - Neumann B.C.: replacement by terms
    - Forward FD ex.) $u_x(0,y) = g(y) \rightarrow -3u_{0,j} + 4u_{1,j} - u_{2,j} = 2hg_j$

    → $(2/3)u_{2,j} + u_{1,j-1} + u_{1,j+1} - (8/3)u_{1,j} = h^2 f_{i,j} + (2/3)hg_j$
    - Ghost boundary ex.) $u_x(0,y) = g(y) \rightarrow u_{-1,j} - u_{1,j} = 2hg_j$

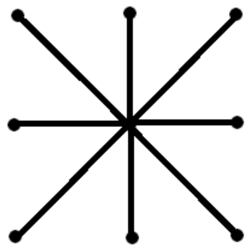
    → $2u_{1,j} + u_{0,j-1} + u_{0,j+1} - 4u_{0,j} = h^2 f_{i,j} + 2hg_j$

    ✓ Matrix elements for $u_{0,j}$ are needed.

參考: Nine-point Stencil (2D)

- 9-point discrete Laplacian (2D)

$\triangle_h u = (4u_{i-1,j} + 4u_{i+1,j} + 4u_{i,j-1} + 4u_{i,j+1}$
$+ u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1}$
$- 20u_{i,j})/(6h^2)$



  - Local truncation error

    $\tau_{i,j} = (h^2/12)\triangle(\triangle u) + O(h^4)$

    - $\triangle(\triangle u) = \triangle f$ if $\triangle u = f$
    - 5-point stencil can be used to calculate $\triangle f$
    - You can obtain a method of order 4 accuracy by setting $f_{i,j} \to f_{i,j} + (h^2/12)\triangle f$

參考: Notes on Linear algebra

- Krylov iterative solvers are most frequently used with matrices from stencils for FDM of $\triangle u = f(x)$
  - Steepest Descent
  - Conjugate Gradient
  - Preconditioned Conjugate Gradient

❖ $\boldsymbol{A}$ (= $\triangle_h u$) must be symmetric positive definite or negative definite for these methods.

❖ For nonlinear PDEs, Newton-Krylov methods are used.

Iterative Methods for Elliptic PDEs

- For the 2-D equation $\triangle u = f(x)$
  - Jacobi

$$u_{i,j}^{k+1} = (u_{i-1,j}^{k} + u_{i+1,j}^{k} + u_{i,j-1}^{k} + u_{i,j+1}^{k} - h^2 f_{i,j})/4$$

    - This can be derived from the 5-point stencil.
  - Gauss-Seidel

$$u_{i,j}^{k+1} = (u_{i-1,j}^{k+1} + u_{i+1,j}^{k} + u_{i,j-1}^{k+1} + u_{i,j+1}^{k} - h^2 f_{i,j})/4$$

    - Twice faster than Jacobi

❖ Computational time ~ $O(m^4 \log m)$

- Grid size = m^2

➢ $O(m^2)$ per each iteration × $O(m^2 \log m)$ iterations

Iterative Methods for Elliptic PDEs

- For the 2-D equation $\triangle u = f(x)$
  - SOR method

$$u_{i,j}^{k+1} = u_{i,j}^{k} + \omega\,(u_{i,j\,[\text{Gauss-Seidel}]} - u_{i,j}^{k})$$
$$= \omega(u_{i-1,j}^{k+1} + u_{i+1,j}^{k} + u_{i,j-1}^{k+1} + u_{i,j+1}^{k} - h^2 f_{i,j})\,/4$$
$$+ (1 - \omega)\, u_{i,j}^{k}$$

- $\omega = 1 \rightarrow$ Gauss-Seidel
- $\omega > 1 \rightarrow$ Successive over-relaxation (SOR)
  - It converges more rapidly than Jacobi or Gauss-Seidel
  - Rapidest when $\omega \approx 2 - 2\pi h$
  - It won't converge when $\omega \geq 2$

Iterative Methods for Elliptic PDEs

$$u_{i,j}^{k+1} = (u_{i-1,j}^{k} + u_{i+1,j}^{k} + u_{i,j-1}^{k} + u_{i,j+1}^{k} - h^2 f_{i,j})/4$$

- Applying boundary conditions (in Jacobi)
  - Dirichlet B.C.: replacement by values
    - Ex.) $u(x,0) = g(x) \rightarrow u_{i,0} = g_i$

    $\rightarrow u_{i,j}^{k+1} = (u_{i-1,1}^{k} + u_{i+1,1}^{k} + u_{i,2}^{k} - h^2 f_{i,1} + g_i)/4$
  - Neumann B.C.: replacement by terms
    - Forward FD ex.) $u_x(0,y) = g(y) \rightarrow -3u_{0,j} + 4u_{1,j} - u_{2,j} = 2hg_j$

    $\rightarrow u_{1,j}^{k+1} = (2u_{2,j}^{k} + 3u_{1,j-1}^{k} + 3u_{1,j+1}^{k} - 3h^2 f_{i,j} - 2hg_j)/8$
    - Ghost boundary ex.) $u_x(0,y) = g(y) \rightarrow u_{-1,j} - u_{1,j} = 2hg_j$

    $\rightarrow u_{0,j}^{k+1} = (2u_{1,j}^{k} + u_{0,j-1}^{k} + u_{0,j+1}^{k} - h^2 f_{i,j} - 2hg_j)/4$

Do It Yourself

- Edit your Jacobi, Gauss-Seidel, or SOR code to solve this PDE problem.

$$u_{xx} + u_{yy} = 1, \qquad 0 < x < 1, 0 < y < 1$$

  - Boundary conditions

$$u(x,0) = 0, u(x,1) = x, \qquad 0 \le x \le 1$$
$$u(0,y) = 0, u(1,y) = y, \qquad 0 \le y \le 1$$

    - Wen Shen 11.5-1

Try a small number of grid intervals and check out your result.

- [After this class]: Try finer grids.

參考: Multi-Grid Technique

- Drawbacks of iterative methods
  - Smooth discrete error modes decay hard while highly oscillating error modes decay fast.
- Damped Jacobi method
  - Jacobi: $\boldsymbol{u}^{k+1} = \boldsymbol{G}\,\boldsymbol{u}^{k}$
  - Damped Jacobi: $\boldsymbol{u}^{k+1} = (1 - \omega)\,\boldsymbol{u}^{k} + \omega\,\boldsymbol{G}\,\boldsymbol{u}^{k}$
  - Usually, $\omega = 2/3$ ($\mu = 1/3$)
  - The damped one is better for multi-grid approach than the original Jacobi.

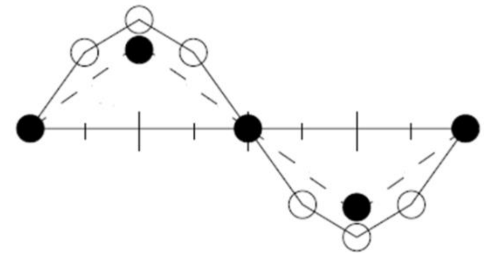
參考: Multi-Grid Technique

- 2-grid V-cycle for Gauss-Seidel (or Jacobi)
  - Case of h and $2h$ grids ($\mu = 0.5$)

1. Iterate 3 steps for $\boldsymbol{Au} = \boldsymbol{f}$
2. Restrict the residual $\boldsymbol{r}^k = \boldsymbol{Au}^k - \boldsymbol{f}$ by $\boldsymbol{r}_{2h} = \boldsymbol{R}_h^{2h}\boldsymbol{r}_h$
   - $\boldsymbol{R}_h^{2h}$: restriction matrix ($h \rightarrow 2h$) $\boldsymbol{R}_h^{2h} = \mu\,(\boldsymbol{I}_{2h}^{h})^{\mathrm{T}}$
3. Solve $\boldsymbol{A}_{2h}\boldsymbol{e}_{2h} = \boldsymbol{r}_{2h}$ or iterate for it on the coarse grid
4. Interpolate $\boldsymbol{e}_{2h}$ by $\boldsymbol{e}_h = \boldsymbol{I}_{2h}^{h}\boldsymbol{e}_{2h}$
   - $\boldsymbol{I}_{2h}^{h}$: (linear) interpolation matrix ($2h \rightarrow h$)
5. Iterate 3 more steps on the fine grid for $\boldsymbol{Au} = \boldsymbol{f}$

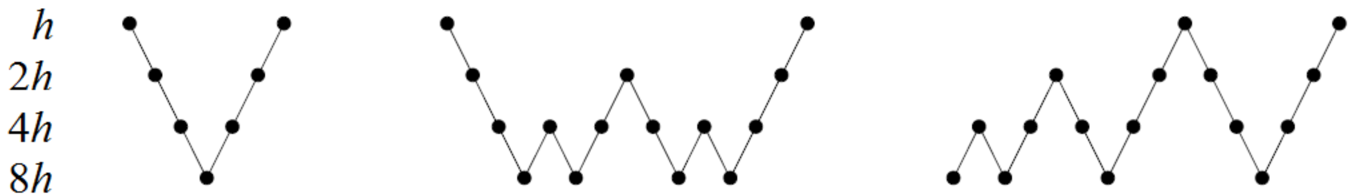
參考: Multi-Grid Technique

- Restriction example



$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & & 1 & 2 & 1 \end{bmatrix}$$

- Other types of multigrid cycles (μ = 0.5)



4-grid V-cycle   W-cycle   Full multigrid cycle

❖ Computational time ~ $O(m^d)$

- $O(1)$ iterations. d : space dimension

Pictures from Strang

FDM for Parabolic PDE

- Method of lines
- Forward Euler
- Backward Euler
- Crank-Nicolson method
- Alternate direction implicit method

Backward Euler Method

- 1-D diffusion equation

$$u_t = Du_{xx}$$

  - Let $U_i^k = u(x_i, t_k)$ where $x_i = ih + x_0, \; t_k = k\delta + t_0$ (uniform grids)

$$u_t \rightarrow \frac{U_i^k - U_i^{k-1}}{\delta}$$

$$u_{xx} \rightarrow \frac{U_{i+1}^k - 2U_i^k + U_{i-1}^k}{h^2}$$

$$u_t = Du_{xx} \rightarrow \frac{U_i^k - U_i^{k-1}}{\delta} = D\frac{U_{i+1}^k - 2U_i^k + U_{i-1}^k}{h^2}$$

  - Let $\gamma = D\delta/h^2$, then

$$-\gamma U_{i+1}^k + (1+2\gamma)U_i^k - \gamma U_{i-1}^k = U_i^{k-1}$$

Backward Euler Method

- 1-D diffusion equation

$$-\gamma U_{i+1}^{k} + (1+2\gamma)U_i^k - \gamma U_{i-1}^{k} = U_i^{k-1}$$

  - Matrix-vector form: $\mathbf{M}\mathbf{u}^k = \mathbf{u}^{k-1}$
    - In case of Dirichlet B. C., 0 at the ends,

$$\mathbf{M} = \begin{pmatrix} 1+2\gamma & -\gamma & 0 & 0 & \cdots & 0 \\ -\gamma & 1+2\gamma & -\gamma & 0 & \cdots & 0 \\ 0 & -\gamma & 1+2\gamma & -\gamma & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & -\gamma & 1+2\gamma & -\gamma & 0 \\ 0 & \cdots & 0 & -\gamma & 1+2\gamma & -\gamma \\ 0 & \cdots & 0 & 0 & -\gamma & 1+2\gamma \end{pmatrix}$$

Backward Euler Method

- 1-D diffusion equation

$$-\gamma U_{i+1}^{k} + (1 + 2\gamma) U_i^k - \gamma U_{i-1}^k = U_i^{k-1}$$

  – The discrete maximum principle always holds.

$$(1 + 2\gamma)\left|U_i^k\right| \le \left|U_i^{k-1}\right| + \gamma\left|U_{i+1}^k\right| + \gamma\left|U_{i-1}^k\right|$$

$$\rightarrow\ (1 + 2\gamma) \max_i\left|U_i^k\right| \le \max_i\left|U_i^{k-1}\right| + 2\gamma \max_i\left|U_i^k\right|$$

$$\therefore \max_i\left|U_i^k\right| \le \max_i\left|U_i^{k-1}\right|$$

  – Accuracy
    - Time: 1st order
    - Space: 2nd order

Crank-Nicolson Method

- Based on the trapezoidal rule (implicit method)

- If a PDE has the form of

$$\partial_t u = f(u, x, y, t, \partial_x u, \partial_y u, \partial^2{}_x u, \partial^2{}_y u)$$

- By discretization, Crank-Nicolson method gives

$$(U_i^{n+1} - U_i^n)/\delta = (F_i^n + F_i^{n+1})/2$$

where $U_i^n = u(x_i, t_n)$ and F_i^n : value of f at t_n and x_i

- Time: 2-point. Space: any finite difference

- Unconditional stability and 2nd order accuracy
- Used for
  - Parabolic PDEs and advection equations

Crank-Nicolson Method

- Ex.) Heat equation $u_t = \triangle u$

$$\rightarrow \frac{u(\vec{x}, t+\delta) - u(\vec{x}, t)}{\delta} = \frac{\triangle_h u(\vec{x}, t+\delta) + \triangle_h u(\vec{x}, t)}{2}$$

  - For the 2-D case,

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{1}{2}\frac{\delta}{h^2}\left[\left(u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1} + u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1} - 4u_{i,j}^{n+1}\right) + \left(u_{i+1,j}^{n} + u_{i-1,j}^{n} + u_{i,j+1}^{n} + u_{i,j-1}^{n} - 4u_{i,j}^{n}\right)\right]$$

$$\rightarrow (1+2\mu)u_{i,j}^{n+1} - \frac{\mu}{2}\left(u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1} + u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1}\right) = (1-2\mu)u_{i,j}^{n} + \frac{\mu}{2}\left(u_{i+1,j}^{n} + u_{i-1,j}^{n} + u_{i,j+1}^{n} + u_{i,j-1}^{n}\right).$$

where $\mu = \delta/h^2$

    - matrix form: $(\mathbf{I} + \mathbf{C})\,\mathbf{u}^{n+1} = (\mathbf{I} - \mathbf{C})\,\mathbf{u}^{n}$

參考: ADI Method

- **Alternate Direction Implicit Method**

- Solving a 2-D or 3-D diffusion equation with Crank-Nicolson needs large cost due to the matrix with a large band width.

- Splitting the matrix into tridiagonal matrices is possible by implicit differentiation in one direction and another.
  - Two stages for 2-D
  - Three stages for 3-D

參考: ADI Method

- Ex.) $u_t = u_{xx} + u_{yy}$

$$u_{i,j}^{n+1/2} = u_{i,j}^{n} + \frac{1}{2}\frac{\delta}{h^2}\left[(u_{i+1,j}^{n+1/2} - 2u_{i,j}^{n+1/2} + u_{i-1,j}^{n+1/2}) + (u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n})\right]$$

$$u_{i,j}^{n+1} = u_{i,j}^{n+1/2} + \frac{1}{2}\frac{\delta}{h^2}\left[(u_{i+1,j}^{n+1/2} - 2u_{i,j}^{n+1/2} + u_{i-1,j}^{n+1/2}) + (u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1})\right]$$

  - Two tridiagonal matrix equation steps

- Cf.) Crank-Nicolson

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{1}{2}\frac{\delta}{h^2}\left[(u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1} + u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1} - 4u_{i,j}^{n+1}) + (u_{i+1,j}^{n} + u_{i-1,j}^{n} + u_{i,j+1}^{n} + u_{i,j-1}^{n} - 4u_{i,j}^{n})\right]$$

參考: ADI Method

- 2^{nd} order accuracy
  - First order accuracy for each step
  - But the 1^{st} order errors cancel each other.
- Unconditionally stable
- For Dirichlet boundary conditions
  - ➢ $u^{n+1/2}$ values at boundaries can be calculated by combining two equations.

❖ There are advanced versions of ADI methods.

Advection Equations

$$\frac{\partial \psi}{\partial t} + \nabla \cdot (\psi \mathbf{u}) = 0$$

- For incompressible flows ($\nabla \cdot \mathrm{u} = 0$),

$$\frac{\partial \psi}{\partial t} + \mathbf{u} \cdot \nabla \psi = 0.$$

- The simplest 1-D case

$$\frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = 0$$

a: constant

✓ Cf.) $\left(\frac{\partial}{\partial t} - a\frac{\partial}{\partial x}\right)\left(\frac{\partial}{\partial t} + a\frac{\partial}{\partial x}\right) u = \left(\frac{\partial^2}{\partial t^2} - a^2 \frac{\partial^2}{\partial x^2}\right) u = 0$

FDM for Advection Equations

- Implicit methods
  - Backward central
  - Crank-Nicolson
- Explicit methods
  - Upwind methods
  - Lax-Friedrichs
  - Leapfrog
  - Lax-Wendroff

FDM for Advection Equations

- FDM for $u_t + au_x = 0$
  - Crank-Nicolson

$$u_i^{k+1} - u_i^{k} + a\sigma(u_{i+1}^{k+1} - u_{i-1}^{k+1} + u_{i+1}^{k} - u_{i-1}^{k})/4 = 0$$

✓ $\sigma = \delta/h$

- Cf.) Backward central
  (BTCS: backward time central space)

$$u_i^{k+1} - u_i^{k} + a\sigma(u_{i+1}^{k+1} - u_{i-1}^{k+1})/2 = 0$$

Cf.) Forward central (FTCS)

$$u_i^{k+1} - u_i^{k} + a\sigma(u_{i+1}^{k} - u_{i-1}^{k})/2 = 0$$

FDM for Advection Equations

- FDM for $u_t + au_x = 0$
  - Forward upwind (if $a > 0$)

    $u_i^{k+1} - u_i^k + a\sigma(u_i^k - u_{i-1}^k) = 0$

    $\rightarrow u_i^{k+1} = u_i^k - a\sigma(u_i^k - u_{i-1}^k)$
  - Backward upwind (if $a < 0$)

    $u_i^{k+1} = u_i^k - a\sigma(u_{i+1}^k - u_i^k)$

  ✓ $\sigma = \delta/h$

  - Cf.) Forward central

    $u_i^{k+1} - u_i^k + a\sigma(u_{i+1}^k - u_{i-1}^k)/2 = 0$

FDM for Advection Equations

- Stable condition for explicit methods
  - CFL condition

A numerical domain of dependence must contain the true domain of dependence.

$\delta \leq h/|a|$ (1-D)

$|a_x|/h_x + |a_y|/h_y \leq 1/\delta$ (2-D)

➢ This means that information outside the region limited by the speed is meaningless.

參考: FDM for Advection Equations

- FDM for $u_t + au_x = 0$
  - Lax-Friedrichs

$u_i^{k+1} = (u_{i-1}^k + u_{i+1}^k)/2 - a\sigma(u_{i+1}^k - u_{i-1}^k)/2$

✓ $\sigma = \delta/h$

- Cf.) Leapfrog

$u_i^{k+1} = u_i^{k-1} - a\sigma(u_{i+1}^k - u_{i-1}^k)$

Cf.) Forward central

$u_i^{k+1} = u_i^k - a\sigma(u_{i+1}^k - u_{i-1}^k)/2$

參考: FDM for Advection Equations

- FDM for $u_t + au_x = 0$
  - Lax-Wendroff

    $u_i^{k+1} = u_i^k - a\sigma(u_{i+1}^k - u_{i-1}^k)/2$
    $\quad + a^2 \cdot \sigma^2 \cdot (u_{i+1}^k - 2u_i^k + u_{i-1}^k)/2$

  ✓ $\sigma = \delta/h$

  ❖ Combinations of the Lax-Wendroff and the upwind methods are possible and called the 'Beam-Warming' methods.

  - Cf.) Forward central

    $u_i^{k+1} = u_i^k - a\sigma(u_{i+1}^k - u_{i-1}^k)/2$

參考: FDM for Advection Equations

- Stability
  - Forward central: unstable
  - Backward central: unconditionally stable
  - Crank-Nicolson: unconditionally stable
  - Upwind: conditionally stable
  - Lax-Friedrichs: conditionally stable
  - Leapfrog: conditionally stable
  - Lax-Wendroff: conditionally stable

參考: FDM for Advection Equations

- Accuracy

Method	Time	Space
Backward central	1st order	2nd order
Crank-Nicolson	2nd order	2nd order
Upwind	1st order	2nd order
Lax-Friedrichs	1st order	2nd order
Leapfrog	2nd order	2nd order
Lax-Wendroff	2nd order	2nd order

參考: FDM for Advection Equations

- Dissipation-dispersion
  - Try wavelike solutions $Ae^{i(kx-\omega t)}$ to

    $u_t + au_x = Du_{xx}$ (1)

    $u_t + au_x = \mu u_{xxx}$ (2)

  → (1) $\omega = ak - iDk^2$ (2) $\omega = ak - \mu k^3$

  → (1) $u \sim \exp(-Dk^2 t)\exp[ik(x - at)]$ (dissipative)

  (2) $u \sim \exp[ik(x - at + \mu k^2 t)]$ (dispersive)
  - Solution of (1) decays as time.
  - Only waves of single *k* hold their form in (2).

參考: FDM for Advection Equations

- Dissipation-dispersion
  - Taylor expansion of a FDM formula can give the 2nd or 3rd derivative terms which have the same form with its truncation error.
  - Ex.) Upwind
    - $\ldots\ldots + (a/2)(h - a\delta)u_{xx} + O(h^2) + O(\delta^2) + O(h\delta) + \ldots\ldots$
  - Ex.) Lax-Wendroff
    - $\ldots\ldots + (a/6)(a^2\delta^2 - h^2)u_{xxx} + O(h^3) + O(\delta^3) + \ldots\ldots$

參考: FDM for Advection Equations

- Dissipation-dispersion

Method	Dissipation	Dispersion
Backward central	Large	Large
Crank-Nicolson	None	Large
Upwind	Large	Small
Lax-Friedrichs	Large	Small
Leapfrog	None	Small
Lax-Wendroff	Small	Small

參考: Fourier Stability Analysis

– Von Neumann stability analysis

- If we apply the discrete Fourier transform to a kind of FDM and get

  $$U_j^n = u(x_j, t_n) = \Xi_n(k)\exp(ikj\Delta x)$$

  Then the method is unstable if

  $$|\Xi_{n+1}(k)/\,\Xi_n(k)| > 1$$

- The CFL condition can be derived from this.

參考: Fourier Stability Analysis

- Ex.) Forward central (FTCS) for $u_t + au_x = 0$

$$u_j^{\,n+1} - u_j^{\,n} + a\sigma(u_{j+1}^{\,n} - u_{j-1}^{\,n})/2 = 0$$

→ $\Xi_{n+1}(k)\exp(ikj\Delta x) - \Xi_n(k)\exp(ikj\Delta x) + a\sigma\{\Xi_n(k)\exp[ik(j+1)\Delta x] - \Xi_n(k)\exp[ik(j\text{-}1)\Delta x]\}/2 = 0$

→ $\Xi_{n+1}(k)/\,\Xi_n(k) = 1 - a\sigma\{\exp(ik\Delta x) - \exp(-ik\Delta x)\}/2 = 1 - ia\sigma\cdot\sin(k\Delta x)$

→ $|\Xi_{n+1}(k)/\,\Xi_n(k)| = [1 + a^2\sigma^2\sin^2(k\Delta x)]^{1/2} > 1$

➢ FTCS is unstable for $u_t + au_x = 0$

參考: FDM for Wave Equations

- A PDE like $\partial^2{}_t u = a^2 \triangle u$ can become a system of 1st order PDEs with auxiliary variables.

  → Apply FDM for advection equations
- System of equations
  - For example, $q = au_x$, $r = au_y$ & $s = u_t$

$$\partial^2{}_t u = a^2 \triangle u \quad \rightarrow \quad \begin{aligned} q_t &= as_x \\ r_t &= as_y \\ s_t &= a(q_x + r_y) \end{aligned}$$

參考: FDM for Wave Equations

- Alternative way
  - Centered second order time difference

$$\frac{\partial^2 u}{\partial t^2} \to \frac{u(\vec{x}, t+\delta) - 2u(\vec{x}, t) + u(\vec{x}, t-\delta)}{\delta^2}$$

Ex.) $\partial^2{}_t u = a^2 \triangle u$

$$\to u(\vec{x}, t+\delta) = 2u(\vec{x}, t) - u(\vec{x}, t-\delta) + a^2 \delta^2 \triangle_h u(\vec{x}, t)$$

參考: Convection-Diffusion Equation

- Also known as *advection-diffusion equation*

- $u_t + au_x = Du_{xx}$
  - Even forward time central space (FTCS) FDM is possible. (Same stable condition with forward Euler for diffusion equation)
  - $a\delta \le h^2$ is safe if you apply Crank-Nicolson.
  - You can also apply the method of lines.

參考: FDM for the Black-Scholes Eq.

$$V_t + \tfrac{1}{2}\sigma^2 s^2 V_{ss} + rsV_s - rV = 0$$

- ✓ A backward equation: A final condition is necessary instead of an initial condition.
- ✓ Boundary conditions
  - – Depending on the final condition, but
    - $s = 0$: $V(0, t) = \text{constant}$
    - $s \to \infty$: calculation from the final option price
- Discretization
  - – Time: $t_k = t_f - k\delta \ (\delta = \Delta t)$
  - – Stock price (s): $s_i = ih \ \ (h = \Delta s)$
  - – Option price (V): $V(s_i, t_k) \rightarrow V_{i,k}$

參考: FDM for the Black-Scholes Eq.

$$V_t + \tfrac{1}{2}\sigma^2 s^2 V_{ss} + rsV_s - rV = 0$$

- Method 1 – Euler method
  - Time derivative
    - Forward finite difference: explicit method
      - Stable condition: $\delta \le \sigma^{-2} N^{-2}$ where N is the grid number for S
    - Backward finite difference: implicit method
  - Stock price derivative
    - Central finite difference (1st & 2nd order derivs.)
      - Stable condition: $\delta \le \sigma^2 r^{-2}$
    - Forward or backward finite difference (1st order deriv.)
      - Forward for $rS \ge 0$ and backward for $rS \le 0$

參考: FDM for the Black-Scholes Eq.

$$V_t + \tfrac{1}{2}\sigma^2 s^2 V_{ss} + rsV_s - rV = 0$$

- Method 2 – Crank-Nicholson
  - Matrix-vector equation

$$A_{i,k+1}V_{i-1,k+1} + \left(1 + B_{i,k+1}\right)V_{i,k+1} + C_{i,k+1}V_{i+1,k+1}$$
$$= -A_{i,k}V_{i-1,k} + \left(1 - B_{i,k}\right)V_{i,k} - C_{i,k}V_{i+1,k}$$

    - where

$$A_{i,k} = \tfrac{1}{4}\delta\left(\sigma_k^2 i^2 + r_k i\right)$$
$$B_{i,k} = -\tfrac{1}{2}\delta(\sigma_k^2 i^2 + r_k)$$
$$C_{i,k} = \tfrac{1}{4}\delta(\sigma_k^2 i^2 - r_k i)$$

參考: FDM for the Navier-Stokes Eq.

- 2-D incompressible Navier-Stokes equation

$$\rho_0 u_t + \rho_0(\mathbf{u}\cdot\nabla)u = -p_x + \mu(u_{xx} + u_{yy})$$

$$\rho_0 v_t + \rho_0(\mathbf{u}\cdot\nabla)v = -p_y + \mu(v_{xx} + v_{yy})$$

$\mathbf{u} = (u, v)$

  - With the continuity equation: $\rho_0(u_x + v_y) = 0$
  - In some cases, $p = \rho_0 U^2 =$ const.

U: characteristic velocity
L: characteristic length

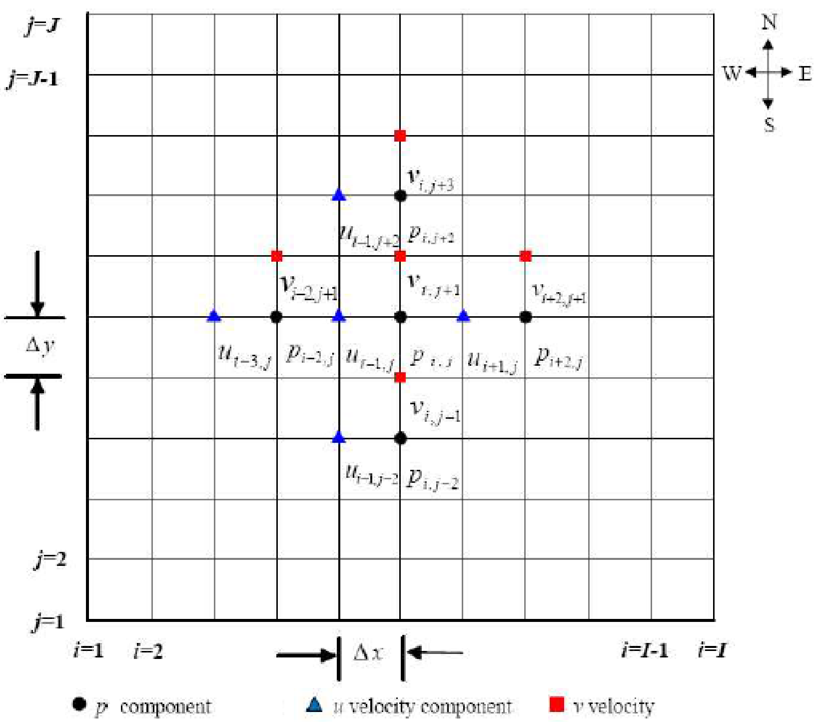
- Reynolds number $Re = \rho_0 UL/\mu$
  - Used in equations of dimensionless variables
- Discretization
  - Space derivatives: usually central FD
  - Time derivatives: usually forward FD

參考: FDM for the Navier-Stokes Eq.

- Time-step δ limit

  $\mu\delta/h^2 \le 1/4$ & $(|u| + |v|)\delta/\mu \le 2$ $\quad (h = \Delta x = \Delta y)$

- Grid choice
  - Staggerd grid →
  - Or non-uniform
    - See A. B. Usov, Comput. Math. Math. Phys. 48, 464 (2008)



N. Rusli *et al.*, Matematika 27, 1 (2011)

參考: FDM for the Navier-Stokes Eq.

- Algorithm
  - N. Rusli *et al.*, Matematika 27, 1 (2011)

1. Initialization: intermediate pressure field (close to average pressure)
2. Find the intermediate velocity fields (close to average velocities) by solving the main PDEs
3. Pressure correction
   - From a deformed continuity equation
4. Editing velocity fields
5. Repeat until convergence

參考: FDM for Euler-Tricomi Eq.

- Euler-Tricomi equation

  $$yu_{xx} + u_{yy} = 0$$

  - Analytically solvable, usually
    - Exact polynomial solutions
    - Mostly used to test a numerical method.
  - FDM formulation
    - u_{yy} : the central 2nd order finite difference
    - u_{xx} : the central 2nd order finite difference for y>0 and the backward finite difference for y<0

      $(U_{i,j} - 2U_{i-1,j} + U_{i-2,j})/h^2$ (1st order)

      or $(2U_{i,j} - 5U_{i-1,j} + 4U_{i-2,j} - U_{i-3,j})/h^2$ (2nd order)
    - Mostly, the multi-grid technique is necessary.

References

- Wen Shen,
  An Introduction to Numerical Computation

- G. D. Smith,
  Numerical Solution of Partial Differential Equations: Finite Difference Methods

- J. C. Strikwerda,
  Finite Difference Schemes and Partial Differential Equations

References

- C. Moler,

  Numerical Computing with MATLAB

- D. M. Causon & C. G. Mingham,

  Introductory Finite Difference Methods for PDEs

- R. J. LeVeque,

  Finite Difference Methods for Differential Equations

References

- H. De Sterck & P. Ullrich,
  Introduction to Computational PDEs

- R. J. LeVeque,
  Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-state and Time-dependent Problems

- G. Strang, “Multigrid Methods”