Numerical Flow Simulation over Gateway Arch of St. Louis
Plan for Today

1) REVIEW
   - Smolarkiewicz, diagonal flow, splitting

2) NUMERICAL METHODS:
   - The origin of Lax-Wendroff
   - Fully implicit schemes

3) NUMERICAL METHODS:
   - Introduction to nesting
Numerical methods:
Lax-Wendroff

References:
• C001 (Lax-Wendroff)
• C006 (Finite differences)
• C007 (Taylor series)
• C052 (Advection)
Where is 1-D Lax-Wendroff from?

- Lax-Wendroff a.k.a. 2\textsuperscript{nd}-order Crowley
  - Taylor series expansion for $\phi^{n+1}$
    - where $\phi(t+\Delta t)$ leads … with extra term on RHS

$$
\phi^{n+1} = \phi^n + \Delta t \phi_t + \frac{\Delta t^2}{2} \phi_{tt} + \ldots

Substitute: \( \phi_t = -c\phi_x, \ \phi_{tt} = c^2\phi_{xx} \)

$$
\begin{align*}
\phi^{n+1} &= \phi^n + \Delta t (-c\phi_x) + \frac{\Delta t^2}{2} (c^2\phi_{xx}) \\
&= \phi^n - c\Delta t \phi_x + \frac{c^2\Delta t^2}{2} \phi_{xx}
\end{align*}

Using centered differences works here - this is what we are using in program #1.
**Review: Lax-Wendroff**

- **Lax-Wendroff: Taylor series, 1-way wave eqn.**

\[
\phi(t + \Delta t) = \phi(t) + \Delta t \frac{\partial \phi}{\partial t} + \frac{(\Delta t)^2}{2!} \frac{\partial^2 \phi}{\partial t^2} + \ldots \text{ (change notation...)}
\]

must approximate these derivatives

= \phi(t) + \Delta t \phi_t + \frac{(\Delta t)^2}{2!} \phi_{tt} + \ldots \text{ ; neglect terms past } \phi_{tt} \text{ for Lax-W.}

1-way wave equation: \( \phi_t = -c \phi_x \) takes care of first time derivative.

For \( \phi_{tt} \), \( \phi_t = -c \phi_x \) \( \Rightarrow \) \( \phi_{tt} = -c \phi_{xt} \) and:

\( \phi_t = -c \phi_x \) \( \Rightarrow \) \( \phi_{xt} = -c \phi_{xx} \) so \( \phi_{tt} = -c (-c \phi_{xx}) = c^2 \phi_{xx} \); Thus,

\[
\phi(t + \Delta t) = \phi(t) + \Delta t (-c \phi_x) + \frac{(\Delta t)^2}{2!} (c^2 \phi_{xx}); \text{ use centered derivatives to finish:}
\]

\[
\phi(t + \Delta t) = \phi(t) - c \Delta t \left( \frac{\phi_{j+1} - \phi_{j-1}}{2\Delta x} \right) + c^2 \frac{(\Delta t)^2}{2!} \left( \frac{\phi_{j+1} - 2\phi_j + \phi_{j-1}}{(\Delta x)^2} \right)
\]
Fully Implicit Schemes

Reference pages for this section:

- C002 – explicit numerical methods
- C070 – implicit numerical methods
Explicit vs. Implicit: advection

At right:

- Group velocities for 1-D advection for explicit (dashed) and implicit (solid)
- Explicit always more accurate for waves longer than $4\Delta x$
Explicit vs. Implicit: advection

**At right:**
- Group velocities for 1-D advection for explicit (dashed) and implicit (solid)
- **Explicit** always more accurate for waves longer than $4\Delta x$

Grotjahn and O'Brien, MWR 104 (1976), Fig. 6
Explicit vs. Implicit: advection

At right:

- Group velocities for 1-D advection for explicit (dashed) and implicit (solid)
- Explicit always more accurate for waves longer than $4\Delta x$

Groffahn and O'Brien, MWR 104 (1976), Fig. 6
Explicit vs. Implicit: advection

At right:

- Group velocities for 1-D advection for explicit (dashed) and implicit (solid)
- Explicit always more accurate for waves longer than $4\Delta x$

Grotjahn and O'Brien, MWR 104 (1976), Fig. 6
Explicit vs. Implicit: advection

At right:

- **Group velocities** for 1-D advection for **explicit** (dashed) and **implicit** (solid)
- **Explicit** more accurate for waves longer than $4\Delta x$
- **Implicit** allows large time step but has large errors, as seen here for $v=10$
- **Group velocities** always negative for $2\Delta x$
Explicit vs. Implicit: advection

- **Relative phase velocity** shown in 3D for implicit and explicit schemes
- $\log_{10}(\mu)$ on abscissa; $\beta = k\Delta x$ on ordinate.
- **Implicit schemes** maintain stability by *slowing down the waves*. 
Explicit advection

\[ \log_{10} \mu \]
Implicit advection

\[ \beta \]

\[ \log_{10} \mu \]

Grotjahn and O'Brien, MWR 104 (1976), Fig. 3
Implicit Schemes: FTCS

- Recall the forward time, centered space (FTCS) scheme applied to the linear advection equation.

\[
\frac{u_j^{n+1} - u_j^n}{\Delta t} = -c \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta t}
\]

- ... has amplification factor: \( \lambda = 1 - i\mu \sin \beta \)
- ... and is thus unstable.

- Stabilize this scheme using \( \frac{du}{dx} \) at \( (n+1) \):

\[
\frac{u_j^{n+1} - u_j^n}{\Delta t} = -c \frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x}
\]
Implicit Schemes: FTCS

- Explicit
  \[
  \frac{u_j^{n+1} - u_j^n}{\Delta t} = -c \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}
  \]
  - Computational molecule
  - Amplification factor
    \[\lambda = 1 - i\mu \sin \beta\]
    (UNSTABLE)

- Fully implicit
  \[
  \frac{u_j^{n+1} - u_j^n}{\Delta t} = -c \frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x}
  \]
  - Computational molecule
  - Amplification factor
    \[\lambda = \frac{1 - i\mu \sin \beta}{1 + \mu^2 \sin^2 \beta}\]
    (STABLE)

Both are \(O(\Delta t, \Delta x^2)\) accurate. Explicit is unstable; Implicit is unconditionally stable but has large errors for large time steps, and requires matrix inversion.
Implicit Schemes: FTCS

- Euler scheme:
  \[ u_{j}^{n+1} - u_{j}^{n} = -(\mu/2)(u_{j+1}^{n+1} - u_{j-1}^{n+1}) \]

- Coefficients of each term:
  \[
  \begin{align*}
  (-\frac{\mu}{2}) u_{j-1}^{n+1} + (1) u_{j}^{n+1} + (\frac{\mu}{2}) u_{j+1}^{n+1} &= u_{j}^{n} \\
  a u_{j-1}^{n+1} + b u_{j}^{n+1} + c u_{j+1}^{n+1} &= D
  \end{align*}
  \]

- Solve set of equations
  - For all \( j \), simultaneously
  - Matrix \([A]\) is tridiagonal
    - rapid solvers exist

- Solution
  \[
  \begin{bmatrix} b_1 & c_1 & 0 & \cdots & \cdots & 0 \\
  a_2 & b_2 & c_2 & & & \\
  0 & a_3 & b_3 & c_3 & & \\
  & & \ddots & \ddots & \ddots & \ddots \\
  & & & \ddots & \ddots & \ddots \\
  0 & & & & a_{M-1} & b_{M-1} \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\
  u_2^{n+1} \\
  \vdots \\
  \vdots \\
  \vdots \\
  u_M^{n+1} \end{bmatrix} = \begin{bmatrix} D_1 \\
  D_2 \\
  \vdots \\
  \vdots \\
  \vdots \\
  D_{M-1} \\
  D_M \end{bmatrix}
  \]

- \( M \) actual grid points: \( j=1, \ldots, M \)
- Boundary values are specified at \( j=0 \) and \( j=M \)
- \( M+2 \) points (total)
- Initial conditions known at \( n=0 \)
Implicit schemes

- There are differencing schemes requiring more points - either for higher order schemes, or more dimensions.

Ferziger p. 55
Implicit schemes

- The matrix layout changes appropriately.

Fig. 3.5. Structure of the matrix for a five-point computational molecule (non-zero entries in the coefficient matrix on five diagonals are shaded; each horizontal set of boxes corresponds to one grid line)
Nesting: introduction
After 1 rotation

No nest

Nest
Why are we nesting

- **To minimize errors** -
  - Amplitude
  - Phase speed
  - Group velocity
  - Truncation error

- **To better resolve small-scale features**
  - ... which are damped out if represented by too few grid points (e.g. less than 4-6Δx)

- **To save computation time**
  - Nested grids allow higher resolution only where needed
Why are we nesting

- To minimize errors -
  - Amplitude
  - Phase speed
  - Group velocity
  - Truncation error

- To better resolve small-scale features

- To save computation time

- Nesting: “move” solution to lower wavenumbers.
Irregular grids

- Instead of regular, structured grids -

“Method of lines” - Grid points reposition themselves during integration

“Dynamic grid adaptation for computational magnetohydrodynamics” - Keppens et al., 2000

Fig. 1. The grid history in a 1D MHD simulation of an oscillating plasma sheet embedded in a vacuum. Starting with an equidistant grid of 250 grid points, the sheet boundaries are automatically recognized as regions where grid points need to be clustered. After this rapid initial adjustment (prior to times $T < 0.05$), the mesh clearly follows the oscillation.
Irregular grids

- Instead of regular, structured grids -

Grid distortion in response to evolving solution

“Dynamic grid adaptation for computational magnetohydrodynamics” - Keppens et al., 2000

Fig. 2. A 2D kinematic flux expulsion. The left panel shows the initial cartesian mesh and the shading corresponds to the magnetic vector potential. Right panel: an imposed four-cell convection pattern causes the initially straight, uniform field to distort, which is recognized and followed by the 2D grid cell movements.
2D nesting

Evolving 2D grid

“Adaptive mesh refinement routines for Overture” - Brown and Henshaw, 2003

Figure 1: Results from hypeAmr, solving a convection diffusion equation with adaptive mesh refinement.
Irregular grids

2D gridpoint redistribution

“Dynamic grid adaptation using the MPDATA scheme” - Iselin et al., 2002
(Time) stepping forward when nesting

Keppens et al., 2000
Nesting: Implementation

COMPUTER PROBLEM #4
Program #4: main routine

IC

T=0

Stats

Stop

Integrate

grid1: n to n+1

Don't do update yet!

Nest Update

BC

for grid1

BC

for nest

Integrate

nest - small \( \Delta t \!

Update

nest

Feedback

nest to coarse grid

Update

grid 1

Plot

Stats

stop done!

until simulation done
(coarse grid loop)