LECTURE 2-2
PDE - GENERAL

Min Sup Hur

Mar. 9 2023

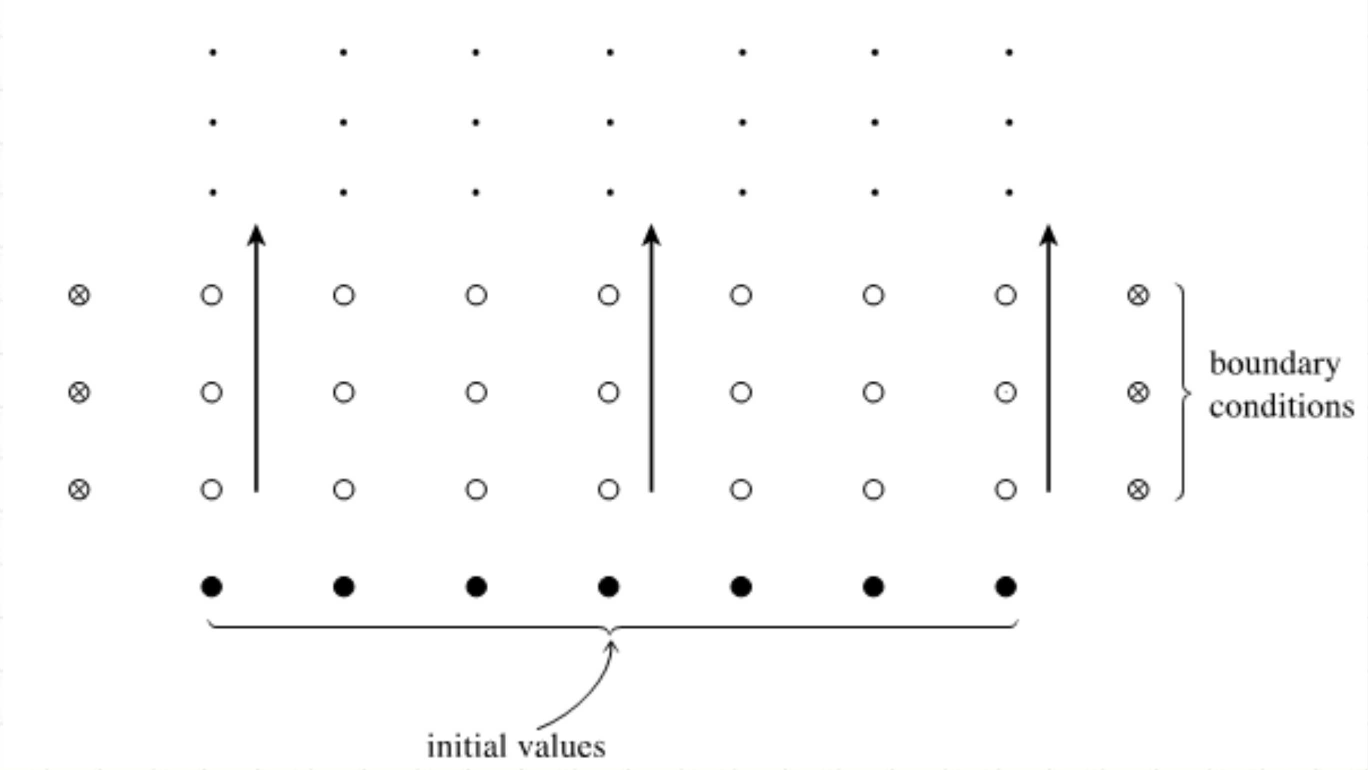
Outline

- Categories of PDE
- Initial Value Problems
- Stability
- Various Stable Schemes for the Continuity Equation

Categories of PDE

Kinds	PDEs	Physical Systems	Dispersion Relation (Fourier Transf.)
Hyperbolic	$\frac{\partial^2\psi}{\partial t^2} - v^2\frac{\partial^2\psi}{\partial x^2} = -A\psi$	Wave etc.	$\omega^2 - v^2k^2 = A$
Parabolic	$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial u}{\partial x}\right)$	Diffusion etc.	$\omega = Dk^2$
Elliptic	$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = -A\phi$	Electro- and magneto-statics etc.	$k_x^2 + k_y^2 = A$

Initial Value Problems



- Any time-dependent equation, such as Conservation Equation, Wave equation, Diffusion Equation etc. needs initial values.
- Any position-dependent equation requires boundary conditions

Continuity Equation

- General form $$\frac{\partial \mathbf{u}}{\partial t} = -\frac{\partial \mathbf{F}(\mathbf{u})}{\partial x}$$

  * Example: the charge conservation equation $$\frac{\partial \rho}{\partial t} + \nabla \cdot \vec{J} = 0$$

- A simple case with a constant velocity $$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x}$$

- Note that in this form, it is also an equation of right-going waves;

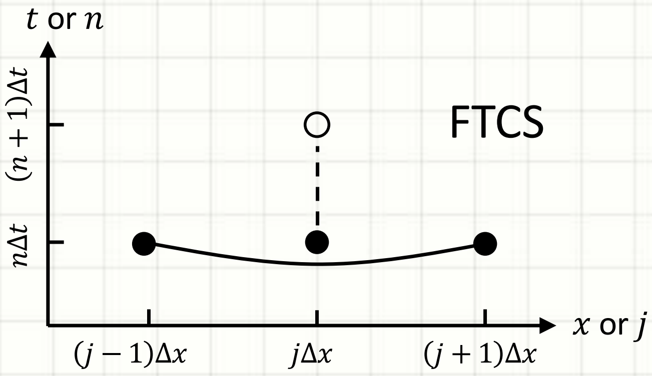
$$\frac{\partial^2 u}{\partial t^2} - \frac{1}{v^2}\frac{\partial^2 u}{\partial x^2} = 0 \quad \rightarrow \quad \left(\frac{\partial}{\partial t} - \frac{1}{v}\frac{\partial}{\partial x}\right)\boxed{\left(\frac{\partial}{\partial t} + \frac{1}{v}\frac{\partial}{\partial x}\right)u} = 0$$

- Discrete form by Forward Time, Centered Space (**FTCS**) scheme

$$\left.\frac{\partial u}{\partial t}\right|_{j,n} = \frac{u_j^{n+1} - u_j^n}{\Delta t} + O(\Delta t) \qquad \left.\frac{\partial u}{\partial x}\right|_{j,n} = \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + O(\Delta x^2)$$

$$\rightarrow \quad \frac{u_j^{n+1} - u_j^n}{\Delta t} = -v\left(\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}\right) + O(\Delta t, \Delta x^2)$$

Solving Continuity Equation by FTCS



- The equation of iteration is

$$u_j^{n+1} = u_j^n - \frac{\alpha}{2}\left(u_{j+1}^n - u_{j-1}^n\right)$$

where $\alpha \equiv \frac{\Delta t}{\Delta x} v$

- When u^{n+1} (future step) at each mesh can be determined solely by u^n's (past step), the method is called *explicit*.

- The matrix form; $\mathbf{u}^{n+1} = \mathbf{F} \cdot \mathbf{u}^n$

  where $\mathbf{u}^n$ is a column vector of u_j^n. Here $j = 1,2,\cdots,N-1$. u_0^n and u_N^n are given as boundary conditions.

- The matrix $\mathbf{F}$ is tridiagonal;

$$\mathbf{F} = \begin{pmatrix}
1 & -\alpha/2 & & & & & \\
\alpha/2 & \ddots & & & & \text{O} & \\
& & 1 & -\alpha/2 & & & \\
& & \alpha/2 & 1 & -\alpha/2 & & \\
& & & \alpha/2 & 1 & & \\
& \text{O} & & & & \ddots & -\alpha/2 \\
& & & & & \alpha/2 & 1
\end{pmatrix}$$

FTCS in Python

Example 1. Create a code to solve the continuity equation (or the right-going wave equation) with v=1 by the FTCS method. Set the domain length to 1, with the boundary values at both ends be zero. Build a pulse of Gaussian-shape, centered at x=0.2 and the half-width at 1/e to be 0.02. Run the code with mesh size=0.001, time step=0.0005, up to 100 steps.

Answer:
```
import numpy as np;  import matplotlib.pyplot as plt
dx=0.001;  dt=0.0005;
x=np.arange(0,1+dx,dx);  a=dt/dx     # simulation domain x and alpha parameter (a).
u=np.exp(-(x-0.2)**2/0.02**2)         # initial Gaussian pulse centered at 0.2.
u0=np.exp(-(x-0.2)**2/0.02**2)        # backup a copy of initial signal for comparison
u[0]=u[len(u)-1]=0.0      # Boundary condition
s=0                       # Loop from here
while s<=100:   # s represents num. steps
        u[1:-1]=u[1:-1] – 0.5*a*(u[2:] –  u[0:-2]) # Calculate from j=1 thru n-2.
        s+=1         # next step
plt.plot(x,u0,x,u); plt.show()
```

Stability Analysis

- From the matrix form of the FTCS method, it is expected that the method is potentially unstable when the norm of the matrix is larger than unity.

$$\mathbf{u}^{n+1} = \mathbf{F} \cdot \mathbf{u}^{n} \quad \rightarrow \quad |\mathbf{u}^{n+1}| = |\mathbf{F} \cdot \mathbf{u}^{n}| \lesssim \ \|\mathbf{F}\| |\mathbf{u}^{n}|$$

where $\|\mathbf{F}\|$ is the Frobenius norm.

$$\|\mathbf{F}\| = \left(\sum_{i,j} |a_{ij}|^2\right)^{1/2}$$

$*$ $|\mathbf{u}^{n}|$ can unstably grow when $\|\mathbf{F}\| > 1$

- The calculation of matrix norm can give a rough estimation of the 'degree' of numerical instability.
- The Neumann stability analysis give more detailed information; stability depending on the mode number. See the next page.

Von Neumann Stability Analysis

- Substitute a fluctuation of an arbitrary wavenumber into the difference equation, and check the ‘growing’ or ‘diminishing’ of that as the time advances.

- To help understand, consider a discretized version of the ODE, $\frac{df}{dt} = Af$.

$$\frac{f^{n+1} - f^n}{\Delta t} = Af^n \quad \rightarrow \quad f^{n+1} = (1 + A\Delta t)f^n$$

- The solution can be written by $f^n = \xi^n f^0$, where $\xi \equiv 1 + A\Delta t$.
- That is, the temporal growing is represented by ξ^n.
- In the PDE, similarly, conceive a temporally growing mode with ξ^n, which is but a spatially eigenmode with an arbitrary wave number.

$$f_j^n = \underline{\xi^n}\,\underline{e^{ik(j\Delta x)}}$$

At the j’th mesh, $x = j\Delta x$

Growing factor (can be complex)

eigenmode of wave number k

If $|\xi| > 1$, the method is unstable. Otherwise, it is stable.

Stability Analysis on the FTCS Method

- Substitute $u_j^n = \xi^n e^{ik(j\Delta x)}$ into the discretized conservation equation;

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -v\left(\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}\right)$$

- Then you obtain,

$$\frac{\xi^{n+1} e^{ik(j\Delta x)} - \xi^n e^{ik(j\Delta x)}}{\Delta t} = -v\left(\frac{\xi^n e^{ik[(j+1)\Delta x]} - \xi^n e^{ik[(j-1)\Delta x]}}{2\Delta x}\right)$$

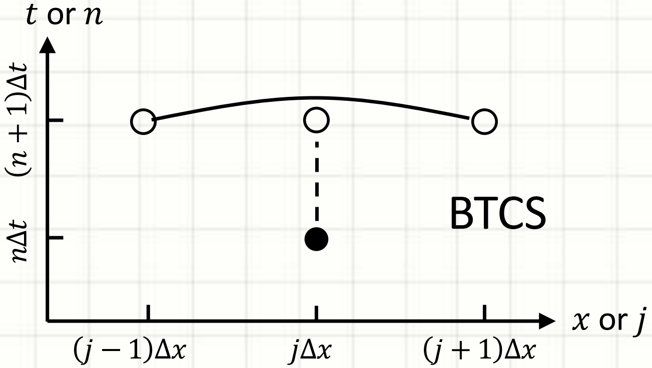
- From a little bit algebra, with $\alpha \equiv \frac{v\Delta t}{\Delta x}$

$$\xi - 1 = -\frac{v\Delta t}{2\Delta x}\left(e^{ik\Delta x} - e^{-ik\Delta x}\right) \quad \rightarrow \quad \xi(k) = 1 - i\,\alpha \sin k\Delta x$$

- The magnitude of the growing factor is larger than unity, indicating the FTCS method for the conservation equation is unconditionally unstable.

$$|\xi| = \sqrt{1 + \alpha^2 \sin^2 k\Delta x} > 1$$ Unconditionally Unstable!

BTCS Method and Stability



- Backward Time, Centered Space (FTCS) scheme

$$\left.\frac{\partial u}{\partial t}\right]_{n+1,j} = \frac{u_j^{n+1} - u_j^n}{\Delta t} + O(\Delta t) \qquad \left.\frac{\partial u}{\partial x}\right]_{n+1,j} = \frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x} + O(\Delta x^2)$$

$$\rightarrow \quad \frac{u_j^{n+1} - u_j^n}{\Delta t} = -v\left(\frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x}\right)$$

- The iterative equation is $u_j^{n+1} + \frac{\alpha}{2}\left(u_{j+1}^{n+1} - u_{j-1}^{n+1}\right) = u_j^n$

- The matrix form is $\mathbf{B} \cdot \mathbf{u}^{n+1} = \mathbf{u}^n$

where

$$\mathbf{B} = \begin{pmatrix}
1 & \alpha/2 & & & & \mathrm{O} & \\
-\alpha/2 & \ddots & & & & & \\
& & 1 & \alpha/2 & & & \\
& & -\alpha/2 & 1 & \alpha/2 & & \\
& & & -\alpha/2 & 1 & & \\
& \mathrm{O} & & & & \ddots & \alpha/2 \\
& & & & & -\alpha/2 & 1
\end{pmatrix}$$

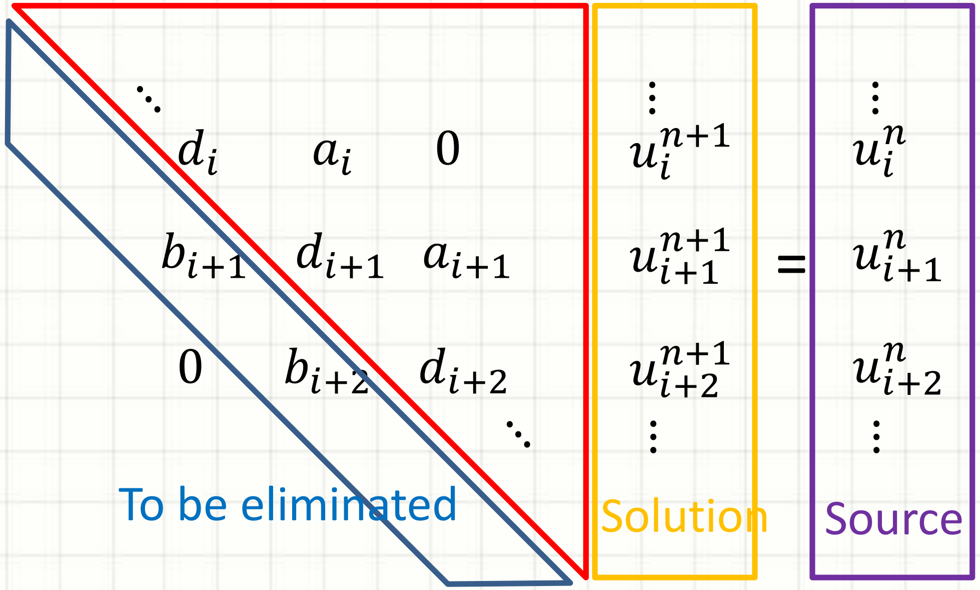
BTCS requires the matrix inversion

This is an *implicit* method

- The growing factor is $\xi = \dfrac{1}{1 + i\alpha \sin k\Delta x} \quad \rightarrow \quad |\xi| = \dfrac{1}{\sqrt{1 + \alpha^2 \sin^2 k\Delta x}} < 1$

BTCS is Unconditionally Stable!

Inverting Matrix B in BTCS



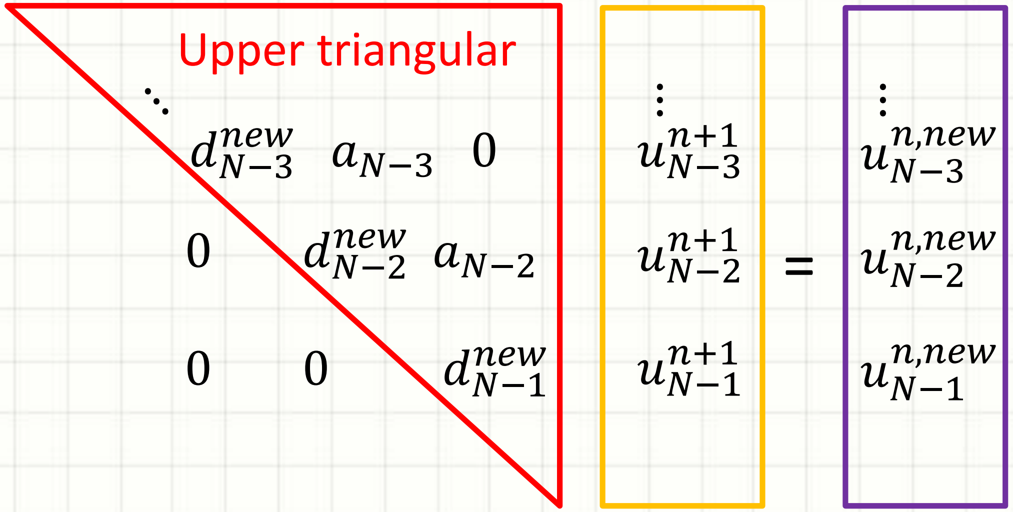
$$\begin{pmatrix} \ddots & & & \\ & d_i & a_i & 0 \\ & b_{i+1} & d_{i+1} & a_{i+1} \\ & 0 & b_{i+2} & d_{i+2} \\ & & & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ u_i^{n+1} \\ u_{i+1}^{n+1} \\ u_{i+2}^{n+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ u_i^{n} \\ u_{i+1}^{n} \\ u_{i+2}^{n} \\ \vdots \end{pmatrix}$$

$$\boxed{Row_{i+1}^{new} = Row_{i+1}^{old} - Row_i^{new}\frac{b_{i+1}^{old}}{d_i^{new}}}$$

Lower $$b_{i+1}^{new} = b_{i+1}^{old} - d_i^{new}\frac{b_{i+1}^{old}}{d_i^{new}} = 0$$

Diagonal $$d_{i+1}^{new} = d_{i+1}^{old} - a_i\frac{b_{i+1}^{old}}{d_i^{new}}$$

Source $$u_{i+1}^{n,new} = u_{i+1}^{n,old} - u_i^{n,new}\frac{b_{i+1}^{old}}{d_i^{new}}$$



$$\begin{pmatrix} \ddots & & & \\ & d_{N-3}^{new} & a_{N-3} & 0 \\ & 0 & d_{N-2}^{new} & a_{N-2} \\ & 0 & 0 & d_{N-1}^{new} \end{pmatrix} \begin{pmatrix} \vdots \\ u_{N-3}^{n+1} \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \end{pmatrix} = \begin{pmatrix} \vdots \\ u_{N-3}^{n,new} \\ u_{N-2}^{n,new} \\ u_{N-1}^{n,new} \end{pmatrix}$$

Backsubstitution

$$u_{N-1}^{n+1} = \frac{u_{N-1}^{n,new}}{d_{N-1}^{new}}$$

$$u_{N-2}^{n+1} = \frac{1}{d_{N-2}^{new}}\left(u_{N-2}^{n,new} - a_{N-2}u_{N-1}^{n+1}\right)$$

$$u_{N-3}^{n+1} = \frac{1}{d_{N-3}^{new}}\left(u_{N-3}^{n,new} - a_{N-3}u_{N-2}^{n+1}\right)$$

BTCS in Python

Example 2. Repeat Exercise 1, but this time, with BTCS.

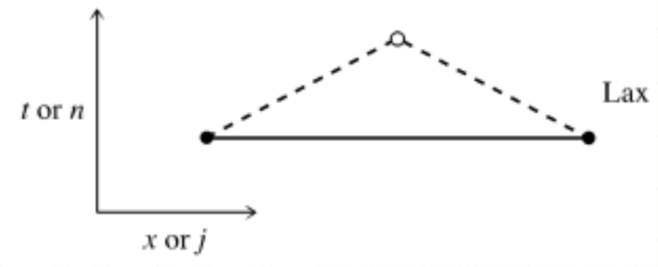
Answer:

```
import numpy as np;  import matplotlib.pyplot as plt
    #  Get the parameters
dx=input(“dx=“);  dt=input(“dt=“);  smax=input(“Max steps=“)
    # Initialization of x, alpha(=dt/dx), number of meshes (N), and initial pulse (u).
x=np.arange(0,1+dx,dx);  a=dt/dx;  N=1/dx
u=np.exp(-(x-0.2)**2/0.02**2);
u0=np.exp(-(x-0.2)**2/0.02**2);      #  Make a copy of initial pulse for later use.
    # Boundary condition
u[0]=u[len(u)-1]=0.0      # Boundary condition
    # Make the matrix triangular.
d=[0,1];  i=1; y=1
while i<N:
              y=1+0.25*a*a/y;  d.append(y);  i+=1     #  $d_{i+1}^{new} = d_{i+1}^{old} - a_i \frac{b_{i+1}^{old}}{d_i^{new}}$
    # Loop
s=0
while s<=smax:
              i=1
              while i<N-1:          # Update the source term    $u_{i+1}^{n,new} = u_{i+1}^{n,old} - u_i^{n,new} \frac{b_{i+1}^{old}}{d_i^{new}}$
                            u[i+1] + = 0.5*u[i]*a/d[i]; i+=1
              while i>=1:           # Backsubstitution    $u_{N-2}^{n+1} = \frac{1}{d_{N-2}^{new}}(u_{N-2}^{n,new} - a_{N-2}u_{N-1}^{n+1})$
                            u[i]=(u[i]-0.5*a*u[i+1])/d[i]; i-=1
              s+=1
```

The Lax Method

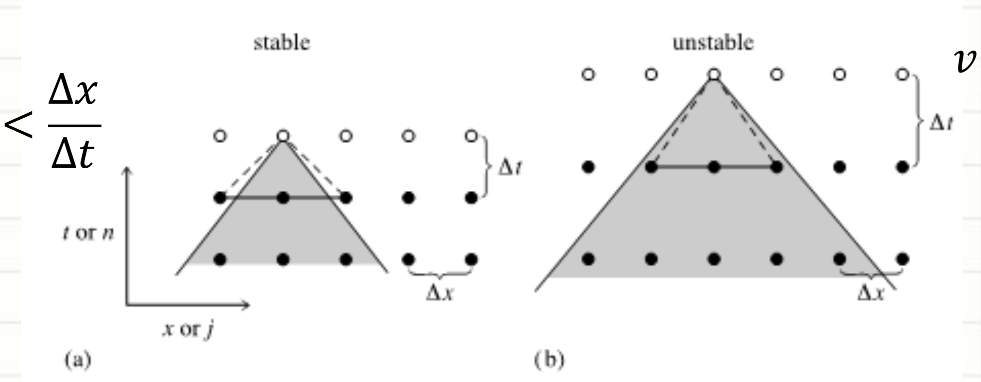
- **Lax method** $u_j^n \rightarrow \frac{1}{2}\left(u_{j+1}^n + u_{j-1}^n\right)$

$$u_j^{n+1} = \frac{1}{2}\left(u_{j+1}^n + u_{j-1}^n\right) - \frac{v\Delta t}{2\Delta x}\left(u_{j+1}^n - u_{j-1}^n\right)$$



- Growing factor $\xi = \cos k\Delta x - i\frac{v\Delta t}{\Delta x}\sin k\Delta x \rightarrow |\xi| = \sqrt{\cos^2 k\Delta x + \alpha^2 \sin^2 k\Delta x}$
- Stability Condition $\boxed{\alpha = \left|\frac{v\Delta t}{\Delta x}\right| \leq 1}$ Courant-Friedrichs-Lewy (CFL) Condition
- Schematic understanding of the CFL condition (for details, see the NR book)

$v < \frac{\Delta x}{\Delta t}$

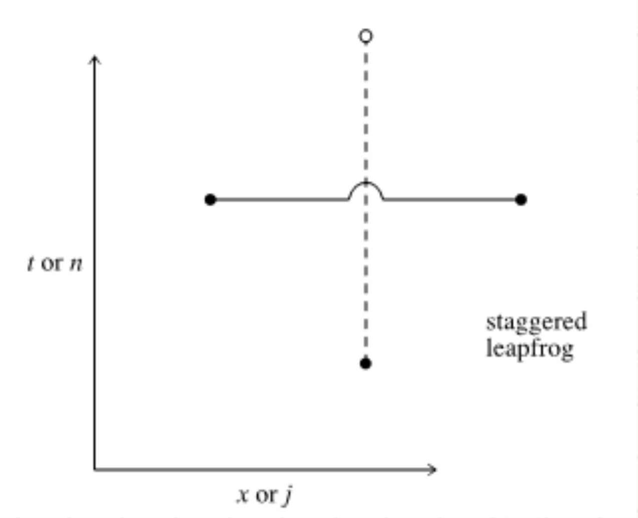


$v > \frac{\Delta x}{\Delta t}$

Second-order Scheme in Time

- **Leap Frog Method**

$$u_j^{n+1} - u_j^{n-1} = -\frac{v\Delta t}{\Delta x}(u_{j+1}^n - u_{j-1}^n)$$



- Stability Condition

Amplification factor

$$\xi^2 - 1 = -2i\xi\frac{v\Delta t}{\Delta x}\sin k\Delta x \longrightarrow \xi = -i\frac{v\Delta t}{\Delta x}\sin k\Delta x \pm \sqrt{1 - \left(\frac{v\Delta t}{\Delta x}\sin k\Delta x\right)^2}$$

$$|\xi|^2 = 1 \text{ for any } v\Delta t \le \Delta x$$

No amplitude dissipation

- Leap Frog Method for Wave equation

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{(\Delta t)^2} = v^2\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2}$$

In this case we have $|\xi|$=1 again as long as CFL is satisfied

Two-step Lax-Wendroff Method

- The leap-frog method is generally not so stable for more complicated equation
- The cure may be the two-step **Lax-Wendroff** scheme, which is low-dissipative, and highly -accurate

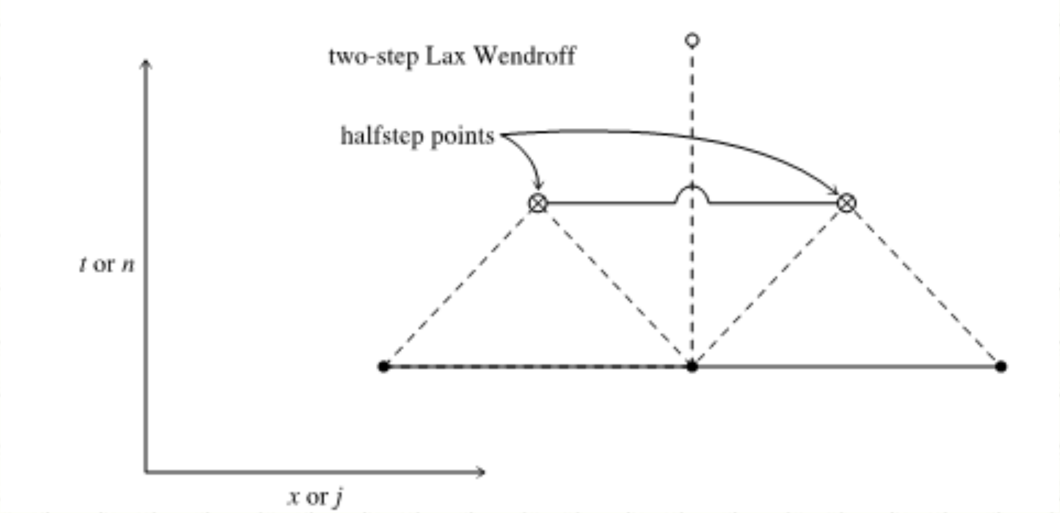
- For a general conservation equation,

$$u_{j+1/2}^{n+1/2} = \frac{1}{2}(u_{j+1}^n + u_j^n) - \frac{\Delta t}{2\Delta x}(F_{j+1}^n - F_j^n)$$

Lax method for a half-step advance

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}\left(F_{j+1/2}^{n+1/2} - F_{j-1/2}^{n+1/2}\right)$$

Another half-step by leapfrog



END OF LECTURE 2-2