These slides are based on the recommended textbook: Culbert B. Laney. “Computational Gas Dynamics,” CAMBRIDGE UNIVERSITY PRESS, ISBN 0-521-62558-0
Outline

1. Conservative Finite Difference Methods in One Dimension
2. Forward, Backward, and Central Time Methods
3. Domain of Dependence and CFL Condition
4. Linear Stability Analysis
5. Formal, Global, and Local Order of Accuracy
6. Upwind Schemes in One Dimension
7. Nonlinear Stability Analysis
8. Multidimensional Extensions
Note: The material covered in this chapter equally applies to scalar conservation laws and to the Euler equations, in one and multiple dimensions. In order to keep things as simple as possible, it is presented in most cases for scalar conservation laws: first in one dimension, then in multiple dimensions.
Conservative Finite Difference Methods in One Dimension

- Recall that scalar conservation laws are simple scalar models of the Euler equations that can be written in strong conservation form as

\[
\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \tag{1}
\]

- Suppose that a 1D space is divided into grid points \(x_i\) and "cells" \([x_{i-1/2}, x_{i+1/2}]\), where \(x_{i+1/2}\) is called a "cell edge"

- Also suppose that time is divided into time-intervals \([t^n, t^{n+1}]\)

- The conservation form of a finite difference method applied to the numerical solution of equation (1) is defined as follows

\[
\Delta t \left( \frac{\partial u}{\partial t} \right)_i^n = -\lambda (\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n) \tag{2}
\]

where the subscript \(i\) designates the point \(x_i\), the superscript \(n\) designates the time \(t^n\), a "hat" designates a time-approximation, and

\[
\lambda = \frac{\Delta t}{\Delta x}, \quad \Delta t = t^{n+1} - t^n, \quad \Delta x = x_{i+1/2} - x_{i-1/2}
\]
One interpretation of the finite difference approach (2) and the conservation form label is the approximation of the following integral form of equation (1)

\[ \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} [u(x, t^{n+1}) - u(x, t^{n})] \, dx \]

\[ = -\frac{1}{\Delta x} \int_{t^n}^{t^{n+1}} [f(u(x_{i+1/2}, t)) - f(u(x_{i-1/2}, t))] \, dt \]

which clearly describes a conservation law.
Not every finite difference method can be written in conservation form: Those which can are called conservative and their associated quantities $\hat{f}_{i+1/2}^n$ are called *conservative numerical fluxes*.

- Finite difference methods derived from the conservation form of the Euler equations or scalar conservation laws tend to be conservative.
- Finite difference methods derived from other differential forms (for example, primitive or characteristic forms) of the aforementioned equations tend not to be conservative.
- Conservative finite differencing implies correct shock and contact locations.

![Conservative and non-conservative finite difference schemes](image)
Like many approximation methods, conservative finite difference methods can be divided into *implicit* and *explicit* methods.

In a typical implicit method

\[
\frac{\partial u}{\partial t}^n_i = \left( \frac{\partial u}{\partial t} \right)_i^n (u_{i-K_1}^n, ..., u_{i+K_2}^n; u_{i-L_1}^{n+1}, ..., u_{i+L_2}^{n+1})
\]

\[
\hat{f}^{n+1/2}_{i+1/2} = \hat{f} (u_{i-K_1+1}^n, ..., u_{i+K_2}^n; u_{i-L_1+1}^{n+1}, ..., u_{i+L_2}^{n+1})
\]  

(3)

so that from (2) one has

\[
u_{i}^{n+1} = u(u_{i-K_1}^n, ..., u_{i+K_2}^n; u_{i-L_1}^{n+1}, ..., u_{i+L_2}^{n+1})
\]

(4)

\[\Rightarrow\] the solution of a system of equations is required at each time-step

Note: if \(u_{i-K_1+1}^n\) in (3) were written as \(u_{i-K_1}^n\), one would get the less convenient notation \(u_{i}^{n+1} = u(u_{i-K_1-1}^n, ..., u_{i+K_2}^n; u_{i-L_1}^{n+1}, ..., u_{i+L_2}^{n+1})\) instead of (4)
Like many approximation methods, conservative finite difference methods can be divided into *implicit* and *explicit* methods (continue)

- in a typical explicit method

\[
\left( \frac{\partial u}{\partial t} \right)_i^n = \left( \frac{\partial u}{\partial t} \right)(u_{i-K_1}^n, \ldots, u_{i+K_2}^n; u_{i+1}^{n+1})
\]
\[
\hat{f}_{i+\frac{1}{2}}^n = \hat{f}(u_{i-K_1+1}^n, \ldots, u_{i+K_2}^n)
\]

so that from (2) one has

\[
u_{i+1}^{n+1} = u(u_{i-K_1}^n, \ldots, u_{i+K_2}^n)
\]

\[\Rightarrow\] only function evaluations are incurred at each time-step

- \((u_{i-K_1}^n, \ldots, u_{i+K_2}^n)\) and \((u_{i-L_1}^{n+1}, \ldots, u_{i+L_2}^{n+1})\) are called the *stencil* or direct numerical domain of dependence of \(u_{i}^{n+1}\)

- \(K_1 + K_2 + 1\) and \(L_1 + L_2 + 1\) are called the *stencil widths*
Conservative Finite Difference Methods in One Dimension

Summary: typical stencil diagram

- Stencil width = 5
  - $L_1 = 2$
  - $L_2 = 2$

- Stencil width = 6
  - $K_1 = 1$
  - $K_2 = 4$
Like any proper numerical approximation, proper finite difference approximation becomes perfect in the limit $\Delta x \to 0$ and $\Delta t \to 0$

- an approximate equation is said to be **consistent** if it equals the true equations in the limit $\Delta x \to 0$ and $\Delta t \to 0$
- a solution to an approximate equation is said to be **convergent** if it equals the true solution of the true equation in the limit $\Delta x \to 0$ and $\Delta t \to 0$

Hence, a conservative approximation is consistent when

$$\hat{f}(u, \ldots, u) = f(u)$$

$\implies$ in this case, the conservative numerical flux $\hat{f}$ is said to be **consistent** with the physical flux

A conservative numerical method — and therefore a conservative finite difference method — automatically **locates** shocks correctly (however, it does not necessarily reproduce the shape of the shock correctly)

A method that explicitly enforces the Rankine-Hugoniot relation is called a **shock-capturing** method
Forward, Backward, and Central Time Methods

Forward Time Methods

- Forward Time (FT) conservative finite difference methods correspond to the choices

\[ \Delta t \left( \frac{\partial u}{\partial t} \right)_i^n = u_i^{n+1} - u_i^n \]  

and  

\[ \hat{f}_{i+1/2}^n = \hat{f}(u_{i-K_1+1}^n, \ldots, u_{i+K_2}^n) \]

- with Forward Space (FS) approximation of the term \( \frac{\partial u}{\partial x}(x_i, t^n) \), this leads to the FTFS scheme

\[ u_i^{n+1} = u_i^n - \lambda(\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n), \quad \text{with} \quad \hat{f}_{i+1/2}^n = f(u_{i+1}^n) \]

- with Backward Space (BS) approximation of the term \( \frac{\partial u}{\partial x}(x_i, t^n) \), this leads to the FTBS scheme

\[ u_i^{n+1} = u_i^n - \lambda(\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n), \quad \text{with} \quad \hat{f}_{i+1/2}^n = f(u_i^n) \]

- with Central Space (CS) approximation of the term \( \frac{\partial u}{\partial x}(x_i, t^n) \), this leads to the FTCS scheme

\[ u_i^{n+1} = u_i^n - \lambda(\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n), \quad \text{with} \quad \hat{f}_{i+1/2}^n = \frac{1}{2} (f(u_{i+1}^n) + f(u_i^n)) \]
Backward Time (BT) conservative finite difference methods correspond to the choices

\[
\Delta t \left( \frac{\partial u}{\partial t} \right)_i^n = u_i^{n+1} - u_i^n \quad \text{and} \quad \hat{f}_{i+1/2}^n = \hat{f}(u_{i-K_1+1}^{n+1}, \ldots, u_{i+K_2}^{n+1})
\]

- with Forward Space (FS) approximation of the term \( \frac{\partial u}{\partial x}(x_i, t^n) \), this leads to the BTFS scheme
  \[
u_i^{n+1} = u_i^n - \lambda(\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n), \quad \text{with} \quad \hat{f}_{i+1/2}^n = f(u_{i+1}^{n+1})
  \]

- with Backward Space (BS) approximation of the term \( \frac{\partial u}{\partial x}(x_i, t^n) \), this leads to the BTBS scheme
  \[
u_i^{n+1} = u_i^n - \lambda(\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n), \quad \text{with} \quad \hat{f}_{i+1/2}^n = f(u_{i}^{n+1})
  \]

- with Central Space (CS) approximation of the term \( \frac{\partial u}{\partial x}(x_i, t^n) \), this leads to the BTCS scheme
  \[
u_i^{n+1} = u_i^n - \lambda(\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n), \quad \text{with} \quad \hat{f}_{i+1/2}^n = \frac{1}{2} \left( f(u_{i+1}^{n+1}) + f(u_{i}^{n+1}) \right)
  \]
Forward, Backward, and Central Time Methods

Central Time (CT) conservative finite difference methods correspond to the choices

$$\Delta t \left( \frac{\partial u}{\partial t} \right)_i^n = \frac{1}{2} (u_{i}^{n+1} - u_{i}^{n-1}) \quad \text{and} \quad \hat{f}_{i+1/2}^n = \hat{f}(u_{i-K_1+1}^n, \ldots, u_{i+K_2}^n)$$

with Forward Space (FS) approximation of the term $\frac{\partial u}{\partial x}(x_i, t^n)$, this leads to the CTFS scheme

$$u_i^{n+1} = u_i^{n-1} - 2\lambda (\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n), \quad \text{with} \quad \hat{f}_{i+1/2}^n = f(u_{i+1}^n)$$

with Backward Space (BS) approximation of the term $\frac{\partial u}{\partial x}(x_i, t^n)$, this leads to the CTBS scheme

$$u_i^{n+1} = u_i^{n-1} - 2\lambda (\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n), \quad \text{with} \quad \hat{f}_{i+1/2}^n = f(u_i^n)$$

with Central Space (CS) approximation of the term $\frac{\partial u}{\partial x}(x_i, t^n)$, this leads to the CTCS scheme

$$u_i^{n+1} = u_i^{n-1} - 2\lambda (\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n), \quad \text{with} \quad \hat{f}_{i+1/2}^n = \frac{1}{2} (f(u_{i+1}^n) + f(u_i^n))$$
Domain of Dependence and CFL Condition

Numerical and Physical Domains of Dependence

- Recall the theory of characteristics: A point in the $x - t$ plane is influenced only by points in a finite domain of dependence and influences only points in a finite range of influence.

- Hence, the physical domain of dependence and physical range of influence are bounded on the right and left by the waves with the highest and lowest speeds.

- In a well-posed problem, the range of influence of the initial and boundary conditions should exactly encompass the entire flow in the $x - t$ plane.
The *direct numerical domain of dependence* of a finite difference method is its stencil: For example, if the solution approximated by an implicit finite difference method can be written as

$$u_{i}^{n+1} = u(u_{i-K_1}^{n}, ..., u_{i+K_2}^{n} ; u_{i-L_1}^{n+1}, ..., u_{i+L_2}^{n+1})$$

its direct numerical domain of dependence is the region of the $x - t$ plane covered by the points $(u_{i-K_1}^{n}, ..., u_{i+K_2}^{n}; u_{i-L_1}^{n+1}, ..., u_{i+L_2}^{n+1})$.

Similarly, if the solution approximated by an explicit finite difference method can be written as

$$u_{i}^{n+1} = u(u_{i-K_1}^{n}, ..., u_{i+K_2}^{n})$$

its direct numerical domain of dependence is the region of the $x - t$ plane covered by the points $(u_{i-K_1}^{n}, ..., u_{i+K_2}^{n})$.

The *full (or complete)* numerical domain of dependence of a finite difference method consists of the union of its direct numerical domain of dependence and the domain covered by the points of the $x - t$ plane upon which the numerical values in the direct numerical domain of dependence depend upon.
The Courant-Friedrichs-Lewy or (CFL) condition

*The full numerical domain of dependence must contain the physical domain of dependence*

Any numerical method that violates the CFL condition misses information affecting the exact solution and may blow up to infinity. For this reason, the CFL condition is *necessary but not sufficient* for numerical stability.
Consider first the linear advection problem

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

\[ u(x, 0) = \begin{cases} 
1 & \text{if } x < 0 \\
0 & \text{if } x \geq 0 
\end{cases} \]

Assume that \( a > 0 \): The exact solution is

\[ u(x, t) = u(x - at, 0) = \begin{cases} 
1 & \text{if } x - at < 0 \\
0 & \text{if } x - at \geq 0 
\end{cases} \]

The FTFS approximation with \( \Delta x = \text{cst} \) is

\[ u_i^{n+1} = (1 + \lambda a)u_i^n - \lambda a u_{i+1}^n \]

\[ u_i^0 = u(i \Delta x, 0) = \begin{cases} 
1 & \text{if } i \leq -1 \\
0 & \text{if } i \geq 0 
\end{cases} \]

where as before, \( \lambda = \frac{\Delta t}{\Delta x} \)
Domain of Dependence and CFL Condition

Scalar Conservation Laws

Then

\[ u_1^1 = \begin{cases} 
1 & \text{if } i \leq -2 \\
1 + \lambda a & \text{if } i = -1 \\
0 & \text{if } i \geq 0 
\end{cases} \]

\[ u_1^2 = \begin{cases} 
1 & \text{if } i \leq -3 \\
(1 + \lambda a)(1 - \lambda a) & \text{if } i = -2 \\
(1 + \lambda a)(1 + \lambda a) & \text{if } i = -1 \\
0 & \text{if } i \geq 0 
\end{cases} \]

and so forth

The first two time-steps reveal that FTFS moves the jump in the wrong direction (left rather than right!) and produces spurious oscillations and overshoots.

Furthermore, the exact solution yields \( u(0, \Delta t) = 1 \), but FTFS yields \( u_0^1 = 0 \)!
This is because FTFS violates the CFL condition

\[ u_{i+1}^{n+1} = (1 + \lambda a)u_i^n - \lambda au_{i+1}^n \]
FTCS satisfies the CFL condition for $-1 \leq \lambda a \leq 1$

However, it almost always blow up (as will be seen in a homework): This illustrates the fact that the CFL condition is a necessary but not sufficient condition for numerical stability.

You can also check that when applied to the solution of any scalar conservation law, the BTCS method always satisfies the CFL condition.
For scalar conservation laws, the CFL condition translates into a simple inequality restricting the wave speed. For the linear advection equation and explicit forward-time method with

\[ u_i^{n+1} = u_i^n - \frac{K_1}{\Delta x} (u_{i-K_1}^n - u_{i+K_2}^n) \]

- in the \(x-t\) plane, the physical domain of dependence is the line of slope \(1/\lambda\)
- in the \(x-t\) plane, the full numerical domain of dependence of \(u_i^{n+1}\) is bounded on the left by a line of slope \(\Delta t / \Delta x = \lambda / K_1\) and on the right by a line of slope \(-\Delta t / \Delta x = -\lambda / K_2\)

hence, the CFL condition is

\[ -\frac{K_2}{\lambda} \leq \frac{\Delta x}{\Delta t} \leq \frac{K_1}{\lambda} \iff -K_2 \leq \lambda a \leq K_1 \]

which requires that waves travel no more than \(K_1\) points to the right or \(K_2\) points to the left during a single time-step.
For scalar conservation laws, the CFL condition translates into a simple inequality restricting the wave speed (continue)

- linear advection equation and explicit forward-time method with
  \[ u_{i}^{n+1} = u(u_{i-K1}, ..., u_{i+K2}) \] (continue)

if \( K_1 = K_2 = K \), the previous CFL condition becomes

\[ \lambda |a| \leq K \] (5)

for this reason, \( \lambda a \) is called the CFL number or the Courant number

keep in mind however that in general, \( a = a(u) \) and therefore the CFL number depends in general on the solution’s range
For scalar conservation laws, the CFL condition translates into a simple inequality restricting the wave speed (continue)

- linear advection equation and implicit backward-time method with
  \[ u_{i}^{n+1} = u(u_{i-K_1}^{n}, \ldots, u_{i+K_2}^{n}; u_{i-L_1}^{n+1}, \ldots, u_{i+L_2}^{n+1}) \]
  - if \( L_1 > 0 \) and \( L_2 = 0 \), the full numerical domain of dependence of \( u_{i}^{n+1} \) includes everything to the left of \( x = x_i \) and beneath \( t = t^{n+1} \) in the \( x - t \) plane
  - if \( L_1 = 0 \) and \( L_2 > 0 \), the full numerical domain of dependence of \( u_{i}^{n+1} \) includes everything to the right of \( x = x_i \) and beneath \( t = t^{n+1} \) in the \( x - t \) plane
  - if \( L_1 > 0 \) and \( L_2 > 0 \), the full numerical domain of dependence of \( u_{i}^{n+1} \) includes everything in the entire \( x - t \) plane beneath \( t = t^{n+1} \)

- conclusion: as long as their stencil includes one point to the left and one to the right, implicit methods avoid CFL restrictions by using the entire computational domain (hence, this includes BTCS but not BTFS or BTBS)
In 1D, the Euler equations have three families of waves that define the physical domain of dependence.

For each family of waves, a CFL condition of a given numerical method can be established as in the case of a scalar conservation law: Then, the overall CFL condition is the most restrictive of all established CFL conditions.

For example, if $K_1 = K_2 = K$, $A$ is the Jacobian matrix of the conservative flux vector, and $\rho(A)$ denotes its spectral radius ($\rho(A) = \max(|v_x - a|, |v_x|, |v_x + a|)$, the CFL condition of an explicit forward-time method becomes (recall (5))

$$\lambda \rho(A) \leq K$$

$\lambda \rho(A)$ is called the CFL number or the Courant number.
Domain of Dependence and CFL Condition

The Euler Equations

\[ \lambda \rho (A) \leq K \]

- For supersonic flows, all waves travel in the same direction, either left or right \( \Rightarrow \) the minimum stencil allowed by the CFL condition contains either \( W_{i-1}^n \) and \( W_i^n \) for right-running supersonic flow, or \( W_i^n \) and \( W_{i+1}^n \) for left-running supersonic flow

- For subsonic flows, waves travel in both directions, and the minimum stencil should always contain \( W_{i-1}^n \), \( W_i^n \), and \( W_{i+1}^n \)

- Hence, a *smart* or *adaptive* stencil can be useful for the case of the Euler equations!
Unstable solutions exhibit *significant* spurious oscillations and/or overshoots.

Unstable solutions of linear problems exhibit *unbounded* spurious oscillations: Their errors grow to infinity as \( t \to \infty \).

Hence the concept of instability discussed here for the solution of linear problems is that of *unbounded growth*.

Since unstable solutions typically oscillate, it makes sense to describe the solution of a linear problem such as a linear advection problem as a Fourier series (sum of oscillatory trigonometric functions).
Linear Stability Analysis

von Neumann Analysis

The Fourier series for the continuous (in space) solution \( u(x, t^n) \) on any spatial domain \([a, b]\) is

\[
  u(x, t^n) = a_0^n + \sum_{m=1}^{\infty} a_m^n \cos \left( 2\pi m \frac{x - a}{b - a} \right) + \sum_{m=1}^{\infty} b_m^n \sin \left( 2\pi m \frac{x - a}{b - a} \right)
\]  
(6)

For the discrete solution \( u_i^n \approx u(x_i, t^n) \), the Fourier series is obtained by sampling (6) as follows

\[
  u_i^n = a_0^n + \sum_{m=1}^{\infty} a_m^n \cos \left( 2\pi m \frac{x_i - a}{b - a} \right) + \sum_{m=1}^{\infty} b_m^n \sin \left( 2\pi m \frac{x_i - a}{b - a} \right)
\]  
(7)

Assume \( x_{i+1} - x_i = \Delta x = \text{cst} \), \( x_0 = a \), and \( x_N = b \) \( \Rightarrow x_i - a = i\Delta x \) and \( b - a = N\Delta x \): This transforms (7) into

\[
  u_i^n = a_0^n + \sum_{m=1}^{\infty} \left( a_m^n \cos \left( 2\pi m \frac{i}{N} \right) + b_m^n \sin \left( 2\pi m \frac{i}{N} \right) \right)
\]  
(8)
Recall that samples can only support wavelengths of $2\Delta x$ or longer (the Nyquist sampling theorem states that samples spaced apart by $\Delta x$ perfectly represent functions whose shortest wavelengths are $4\Delta x$): Hence (8) is truncated as follows

$$u^n_i \approx a^n_0 + \sum_{m=1}^{N/2} \left( a^n_m \cos \left( \frac{2\pi mi}{N} \right) + b^n_m \sin \left( \frac{2\pi mi}{N} \right) \right)$$

An equivalent expression in the complex plane using $I$ as the notation for the pure imaginary number ($I^2 = -1$) is

$$u^n_i \approx \sum_{m=-N/2}^{N/2} C^n_m e^{i \frac{2\pi mi}{N}} = \sum_{m=-N/2}^{N/2} u^n_{i_m}$$

From (8), (9), and Euler’s formula $e^{i\theta} = \cos \theta + I \sin \theta$ it follows that

$$C^n_0 = a^n_0, \quad C^n_m = \frac{a^n_m - Ib^n_m}{2}, \quad C^n_{-m} = \frac{a^n_m + Ib^n_m}{2}$$
Linear Stability Analysis

von Neumann Analysis

Hence, each term of the Fourier series can be written as

\[ u_{i,m}^n = C_m^n e^{i \frac{2\pi m i}{N}} = C_m^n e^{i \phi_{m,i}} \]

where \( \phi_m = \frac{2\pi m}{N} \) and \( m = -N/2, \ldots, N/2 \)

Because of linearity the amplification factor

\[ G_m = \frac{C_{m+1}}{C_m} = G_m(\lambda) \]

does not depend on \( n \): However, it depends on \( \lambda \) (since \( u_{i}^{n+1} \) and \( u_{i}^{n} \) are produced by the numerical scheme being analyzed) which itself depends on \( \Delta t \)

Hence, each term of the Fourier series can be expressed as

\[ u_{i,m}^n = \frac{C_m^n}{C_{m-1}^n} \cdots \frac{C_m^2}{C_{m-1}^1} \frac{C_m^1}{C_{m-1}^0} C_m^0 e^{i \phi_{m,i}} = G_m \cdots G_m C_m^0 e^{i \phi_{m,i}} = G_m^n C_m^0 e^{i \phi_{m,i}} \]

where \( G_m^n = G_m^n(\lambda) = (G_m(\lambda))^n \)
Finally, assume that $C_m^0 = 1$ (for example): This leads to

$$u_{im}^n = G_m^n(\lambda)e^{l\phi_m i}$$

Conclusions

- the linear approximation is linearly stable if $|G_m(\lambda)| < 1$ for all $m$
- it is neutrally linearly stable if $|G_m(\lambda)| \leq 1$ for all $m$ and $|G_m(\lambda)| = 1$ for some $m$
- it is linearly unstable if $|G_m(\lambda)| > 1$ for some $m$

Each of the above conclusion can be re-written in terms of $\lambda = \frac{\Delta t}{\Delta x}$

Application (in class): apply the von Neumann analysis to determine the stability of the FTFS scheme for the linear advection equation
Shortcomings of the von Neumann stability analysis method:
- Requires the solution to be periodic \((u^n_0 = u^n_N)\)
- Requires constant spacing \(\Delta x\)
- Does not account for the boundary conditions

Alternative method: so-called Matrix (eigenvalue analysis) Method based on the fact that for a linear problem and a linear approximation method, one has

\[
  u^{n+1} = M(\lambda)u^n, \quad \text{where} \quad u^n = (u^n_0 \ u^n_1 \cdots \ u^n_N)^T
\]

and \(M\) is an amplification matrix which depends on the approximation scheme and on \(\lambda\)

This implies

\[
  u^n = M^n(\lambda)u^0
\]
Linear Stability Analysis

Matrix Method

\[ u^n = M^n(\lambda)u^0 \]

- Suppose that \( M \) is diagonalizable
  \[ M(\lambda) = Q^{-1}\Lambda(\lambda)Q, \quad \Lambda = \text{diag} (\lambda_1(\lambda), \cdots, \lambda_N(\lambda)) \]

- Then
  \[ M^n(\lambda) = Q^{-1}\Lambda^n Q \Rightarrow (Qu)^n = \Lambda^n(\lambda)(Qu)^0 \]

- Conclusions
  - the linear approximation is linearly stable if \( \rho(M(\lambda)) < 1 \)
  - it is neutrally linearly stable if \( \rho(M(\lambda)) = 1 \)
  - it is linearly unstable if \( \rho(M(\lambda)) > 1 \)
Linear Stability Analysis

Matrix Method

Advantages of the Matrix Method for (linear) stability analysis
- does not require the solution to be periodic
- does not require constant grid spacing
- incorporates the effects of the boundary conditions

Shortcoming: in general, the computation of $\rho(M)$ is not trivial

However, the above shortcoming is not an issue when the objective is to prove the unconditional stability of an (implicit) scheme
- re-write the linear version of equation (2) before time-discretization in matrix form as

$$\frac{du}{dt} + B(\Delta x)u = 0 \quad (10)$$

- suppose that $B$ is diagonalizable and transform equation (10) into the set of independent scalar equations

$$\frac{dv_m}{dt} + \mu_m(\Delta x)v_m = 0, \quad \mu_m > 0, \quad m = 1, \cdots, N$$

- focus on one of the above equations and discretize it in time
- apply the scalar form of the Matrix Method for stability analysis: if the conclusion turns out to be independent of $\mu_m$, then the aforementioned shortcoming is not an issue
- example (in class): apply the Matrix Method to determine the stability of a BT scheme for the linear advection equation
Formal order of accuracy measures the order of accuracy of the individual space and time approximations
- Taylor series expansions
- modified linear equations

However due to instability, formal order of accuracy may not be indicative of the actual performance of a method: For example, recall that a stability condition is \( \lambda \rho(A) \leq K \Rightarrow \Delta t \rho(A) \leq K \Delta x \) and observe that such a stability condition prevents, for example, fixing \( \Delta t \) and studying the order of accuracy of the individual space approximation when \( \Delta x \to 0 \).
Besides formal order of accuracy, one way to measure the order of accuracy is to reduce $\Delta x$ and $\Delta t$ simultaneously while maintaining $\lambda = \frac{\Delta t}{\Delta x}$ constant and fixing the initial and boundary conditions.

In this case, a method is said to have *global* $p$-th order of accuracy (in space and time) if

$$\|e\|_\infty \leq C_x \Delta x^p = C_t \Delta t^p, \quad e_i = u(x_i, t_i^n) - u_i^n$$

for some constant $C_x$ and the related constant $C_t = \frac{C_x}{\lambda^p}$.

Other error measures can be obtained by using the 1-norm, 2-norm, or any vector norm, or if the error is measured pointwise.
Determining analytically the global order of accuracy defined above can be challenging: For this reason, it is usually predicted by comparing two different numerical solutions obtained using the same numerical method but two different values of $\Delta x$

$$p = \frac{\log(\|e_2\|_\infty / \|e_1\|_\infty)}{\log(\Delta x_2 / \Delta x_1)} = \frac{\log(|e_2|_\infty / |e_1|_\infty)}{\log(\Delta t_2 / \Delta t_1)}$$

where $e_{i_l} = |u(x, t^n) - u_i^n|$ is the absolute error for $\Delta x_l$ and $\Delta t_l = \lambda \Delta x_l$, $l = 1, 2$
Another way to measure the order of accuracy is to assume that the solution is perfect at time $t^n$ — that is, $u^n_i = u(x_i, t^n) \forall i$, which is usually true for $n = 0$ — and measure the local (in time) truncation error induced by a single time-step

$$\bar{e}_i = \frac{u(x_i, t^{n+1}) - u^n_i}{\Delta t}$$

Now, let $\Delta t \to 0$ and $\Delta x \to 0$ while maintaining $\lambda = \frac{\Delta t}{\Delta x}$, and the initial and boundary conditions fixed: Then, a method is said to have local $p$-th order of accuracy (in space and time) if

$$\|\bar{e}_i\|_\infty \leq C_x \Delta x^p = C_t \Delta t^p$$

for some constant $C_x$ and the related constant $C_t = \frac{C_x}{\lambda^p}$.

Unlike the global order of accuracy, the local order of accuracy is relatively easy to determine analytically.

Example (in class): determine analytically the local order of accuracy of the FTFS scheme for the linear advection equation.
Consider a nonlinear scalar conservation law

In 1D, there are right-running waves and left-running waves: For right-running waves, right is the *downwind* direction and left is the *upwind* direction

Similarly for left-running waves, left is the *downwind* direction and right is the *upwind* direction

Then every numerical approximation to a scalar conservation law can be described as

- **Centered**: if its stencil contains equal numbers of points in both directions
- **Upwind**: if its stencil contains more points in the upwind direction
- **Downwind**: if its stencil contains more points in the downwind direction

Upwind and downwind stencils are *adjustable* or *adaptive* stencils: Upwind and downwind methods test for wind direction and then, based on the outcome of the tests, select either a right- or a left-biased stencil
Upwind Schemes in One Dimension

physical domain of dependence

\( a > 0 \)  \( \implies \)  upwind  \( \rightarrow \)  downwind

\( (x_i, t^{n+1}) \)
Upwind Schemes in One Dimension

\[ a > 0 \quad \text{upwind} \]

\[ (x_i, t^{n+1}) \quad \text{centered} \]

\[ \text{downwind} \]
Upwinding ensures shock avoidance if the shock reverses the wind, whereas central differencing does not.
Upwinding does not ensure shock avoidance if the shock does not reverse the wind.

\[ a > 0 \implies \quad a > 0 \]

Downwinding on the right above avoids the shock but violates the CFL condition and thus would create larger errors than crossing the shock would.
General remarks

- Upwind methods are popular because of their excellent shock capturing ability.
- Among simple FT or BT methods, upwind methods outdo centered methods. However, higher-order upwind methods often have no special advantages over higher-order centered methods.

Sample techniques for designing methods with upwind and adaptive stencils

- Flux averaging methods
- Flux splitting methods
- Wave speed splitting methods
Upwind Schemes in One Dimension

Introduction to Flux Splitting

- Flux splitting is defined as
  \[ f(u) = f^+(u) + f^-(u) \]
  \[ \frac{df^+}{du} \geq 0, \quad \frac{df^-}{du} \leq 0 \]

- Hence, \( f^+(u) \) is associated with a right-running wave and \( f^-(u) \) is associated with a left-running wave.

- Using flux splitting, the governing conservation law becomes
  \[ \frac{\partial u}{\partial t} + \frac{\partial f^+}{\partial x} + \frac{\partial f^-}{\partial x} = 0 \]
  which is called the *flux split form*.

- Then, \( \frac{\partial f^+}{\partial x} \) can be discretized conservatively using at least one point to the left, and \( \frac{\partial f^-}{\partial x} \) can be discretized conservatively using at least one point to the right, thus obtaining conservation and satisfaction of the CFL condition.
Unfortunately, because in general

$$\frac{\partial f^+}{\partial u} \neq \left(\frac{\partial f}{\partial u}\right)^+ \quad \text{and} \quad \frac{\partial f^-}{\partial u} \neq \left(\frac{\partial f}{\partial u}\right)^-$$

flux splitting cannot describe the true connection between fluxes and waves, unless all waves run in the same direction

- if all waves are right-running, the unique physical flux splitting is $f^+ = f$ and $f^- = 0$
- if all waves are left-running, the unique physical flux splitting is $f^- = f$ and $f^+ = 0$

This is the case only for (nonlinear) scalar conservation laws away from sonic points, and for the Euler equations in the supersonic regime
Upwind Schemes in One Dimension

Introduction to Flux Splitting

- Assume that $\frac{\partial f^+}{\partial x}$ is discretized with a leftward bias so that the approximation at $x = x_i$ is centered or biased towards $x = x_{i-1/2}$

\[
\left( \frac{\partial f^+}{\partial x} \right)_i \approx \frac{\Delta \hat{f}^+_{i-1/2}}{\Delta x} \quad \text{for some } \Delta \hat{f}^+_{i-1/2}
\]

- Assume that $\frac{\partial f^-}{\partial x}$ is discretized with a rightward bias so that the approximation at $x = x_i$ is centered or biased towards $x = x_{i+1/2}$

\[
\left( \frac{\partial f^-}{\partial x} \right)_i \approx \frac{\Delta \hat{f}^-_{i+1/2}}{\Delta x} \quad \text{for some } \Delta \hat{f}^-_{i+1/2}
\]

- Using forward Euler to perform the time-discretization leads to

\[
u_i^{n+1} = u_i^n - \lambda (\Delta \hat{f}^+_{i-1/2} + \Delta \hat{f}^-_{i+1/2})
\]

which is called the *flux split form* of the numerical approximation.
A method in flux split form is conservative if and only if
\[ \Delta \hat{f}_{i+1/2}^+ + \Delta \hat{f}_{i+1/2}^- = g_{i+1}^n - g_i^n \text{ for some } g_i^n \tag{11} \]

Proof

\[ u_i^{n+1} = u_i^n - \lambda (\Delta \hat{f}_{i-1/2}^+ + g_i^n - g_i^n + \Delta \hat{f}_{i+1/2}^-) \]

compare with the conservation form \[ u_i^{n+1} = u_i^n - \lambda (\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n) \]

\[ \implies \hat{f}_{i+1/2}^n = \Delta \hat{f}_{i-1/2}^- + g_i^n, \quad \hat{f}_{i-1/2}^n = -\Delta \hat{f}_{i-1/2}^+ + g_i^n \]

\[ \implies \Delta \hat{f}_{i+1/2}^- = \hat{f}_{i+1/2}^n - g_i^n, \quad \Delta \hat{f}_{i-1/2}^+ = -\hat{f}_{i-1/2}^n + g_i^n \]

require now that
\[ \hat{f}_{i+1/2}^n = \hat{f}_{(i+1)-1/2}^n \implies \Delta \hat{f}_{i+1/2}^- + g_i^n = -\Delta \hat{f}_{i+1/2}^+ + g_{i+1}^n \]

\[ \implies \Delta \hat{f}_{i+1/2}^+ + \Delta \hat{f}_{i+1/2}^- = g_{i+1}^n - g_i^n \]

Since there are no restrictions on \( g_i^n \), every conservative method has infinitely many flux split forms that are useful for nonlinear stability analysis.
Example: Design a first-order upwind method for Burgers' equation using flux splitting then re-write it in conservation form.

For Burgers' equation, the unique physical flux splitting is:

\[ f(u) = \frac{u^2}{2} = \max(0, u) \frac{u}{2} + \min(0, u) \frac{u}{2} \]

\[ f^+(u) \quad f^-(u) \]

\[ f(u) \]

\[ f^+(u) \]

\[ f^-(u) \]
Upwind Schemes in One Dimension

Introduction to Flux Splitting

Example: Design a first-order upwind method for Burgers’ equation using flux splitting then re-write it in conservation form (continue)

- A flux split form of Burgers’ equation is

  \[ \frac{\partial u}{\partial t} + \frac{1}{2} \frac{\partial}{\partial x} (\max(0, u)u) + \frac{1}{2} \frac{\partial}{\partial x} (\min(0, u)u) = 0 \]

- A backward-space approximation of \( \frac{\partial f^+}{\partial x} \) gives

  \[ \left( \frac{\partial}{\partial x} (\max(0, u)u) \right)_i \approx \max(0, u_i^n)u_i^n - \max(0, u_{i-1}^n)u_{i-1}^n \]

- A forward-space approximation of \( \frac{\partial f^-}{\partial x} \) gives

  \[ \left( \frac{\partial}{\partial x} (\min(0, u)u) \right)_i \approx \min(0, u_{i+1}^n)u_{i+1}^n - \min(0, u_i^n)u_i^n \]

- Combining these with an FT approximation yields

  \[ u_i^{n+1} = u_i^n - \frac{\lambda}{2} (\max(0, u_i^n)u_i^n - \max(0, u_{i-1}^n)u_{i-1}^n) \]

  \[ \quad - \frac{\lambda}{2} (\min(0, u_{i+1}^n)u_{i+1}^n - \min(0, u_i^n)u_i^n) \]
Example: Design a first-order upwind method for Burgers’ equation using flux splitting then re-write it in conservation form (continue)

- the reader can check that the first-order upwind method described in the previous page can be re-written in conservation form using

\[ g_i^n = \frac{1}{2} (u_i^n)^2 \]
In contrast with flux splitting, wave speed splitting uses the governing equations in non conservation form and tends to yield non conservative approximations.

Hence in most cases, flux splitting is preferred over wave speed splitting ... except when the flux function has the property

\[ f(u) = \frac{df}{du} u = a(u)u \]

which means that \( f(u) \) is a homogeneous function of degree 1 (recall Euler’s theorem which states that a differentiable function \( f(u) \) is a homogeneous function of degree \( p \) if and only if \( (df/du) \ u = pf(u) \)): This property makes flux splitting and wave speed splitting closely related.
For scalar conservation laws, wave speed splitting can be written as

\[
a(u) = a^+(u) + a^-(u)
\]

\[
a^+(u) \geq 0, \quad a^-(u) \leq 0
\]

Then, the scalar conservation law can be written as

\[
\frac{\partial u}{\partial t} + a^+ \frac{\partial u}{\partial x} + a^- \frac{\partial u}{\partial x} = 0
\]

which is called the wave speed split form

Then, \( a^+ \frac{\partial u}{\partial x} \) can be discretized conservatively using at least one point to the left, and \( a^- \frac{\partial u}{\partial x} \) can be discretized conservatively using at least one point to the right, thus obtaining satisfaction of the CFL condition.
Next, consider vector conservation laws such as the Euler equations.

Split the Jacobian matrix as follows

\[ A(u) = A^+(u) + A^-(u) \]

where the eigenvalues of \( A^+ \) are non negative and those of \( A^- \) are non positive

\[ A^+ \geq 0, \quad A^- \leq 0 \]

Recall that \( A^+ \) and \( A^- \) are obtained by computing and splitting the eigenvalues of \( A \).

The wave speed split form of the Euler equations can then be written as

\[ \frac{\partial u}{\partial t} + A^+ \frac{\partial u}{\partial x} + A^- \frac{\partial u}{\partial x} = 0 \]

Again, \( A^+ \frac{\partial u}{\partial x} \) can then be discretized conservatively using at least one point to the left, and \( A^- \frac{\partial u}{\partial x} \) using at least one point to the right, thus obtaining satisfaction of the CFL condition.
If $f(u)$ is a homogeneous function of degree 1, then from Euler’s theorem it follows that

$$f(u) = a(u)u \Rightarrow f^\pm(u) = a^\pm(u)u$$

However, the above flux vector splitting may or may not satisfy

$$\frac{df^+}{du} \geq 0 \quad \text{and} \quad \frac{df^-}{du} \leq 0$$
Upwind Schemes in One Dimension

Introduction to Wave Speed Splitting

- Assume that $a^+ \frac{\partial u}{\partial x}$ is discretized with a leftward bias so that the approximation at $x = x_i$ is centered or biased towards $x = x_{i-1/2}$

$$
(a^+ \frac{\partial u}{\partial x})_{i}^n \approx a^+_{i-1/2} \frac{u_i^n - u_{i-1}^n}{\Delta x}
$$

- Assume that $a^- \frac{\partial u}{\partial x}$ is discretized with a rightward bias so that the approximation at $x = x_i$ is centered or biased towards $x = x_{i+1/2}$

$$
(a^- \frac{\partial u}{\partial x})_{i}^n \approx a^-_{i+1/2} \frac{u_{i+1}^n - u_i^n}{\Delta x}
$$

- Using forward Euler to perform the time-discretization leads to

$$
u_{i}^{n+1} = u_i^n - \lambda a^-_{i+1/2}(u_{i+1}^n - u_i^n) - \lambda a^+_{i-1/2}(u_i^n - u_{i-1}^n)
$$

which is called the wave speed split form of the numerical approximation.
The flux split form and wave speed form are connected via

\[ \Delta \hat{f}_{i+1/2}^\pm = a_{i+1/2}^\pm (u_{i+1}^n - u_i^n) \]

From the above relation and equation (11), it follows that

\[ (a_{i+1/2}^+ + a_{i+1/2}^-)(u_{i+1}^n - u_i^n) = g_{i+1}^n - g_i^n \text{ for some flux function } g_i^n \]

Hence, the transformation from conservation form to wave speed form and vice versa is

\[ \hat{f}_{i+1/2}^n = a_{i+1/2}^- (u_{i+1}^n - u_i^n) + g_i^n, \quad \hat{f}_{i-1/2}^n = -a_{i-1/2}^+ (u_i^n - u_{i-1}^n) + g_i^n \]
Upwind Schemes in One Dimension

Introduction to Wave Speed Splitting

\[ u_{i}^{n+1} = u_{i}^{n} - \lambda a_{i+1/2}^{-n}(u_{i+1}^{n} - u_{i}^{n}) - \lambda a_{i-1/2}^{+n}(u_{i}^{n} - u_{i-1}^{n}) \]

The above notation for the wave speed split form is the standard notation when wave speed splitting is used to derive new approximation methods.

Wave speed split form is also often used as a preliminary step in nonlinear stability analysis, in which case the standard notation is

\[ u_{i}^{n+1} = u_{i}^{n} + C_{i+1/2}^{+n}(u_{i+1}^{n} - u_{i}^{n}) - C_{i-1/2}^{-n}(u_{i}^{n} - u_{i-1}^{n}) \]

Hence

\[ C_{i+1/2}^{+n} = -\lambda a_{i+1/2}^{-n}, \quad C_{i-1/2}^{-n} = \lambda a_{i-1/2}^{+n} \leftrightarrow C_{i+1/2}^{-n} = \lambda a_{i+1/2}^{+n} \]
From (14), it follows that if a method is derived using wave speed splitting and not just written in wave speed split form, the splitting underlying (12) can also be written as

\[ \lambda a(u) = C^-(u) - C^+(u), \quad C^+(u) \geq 0, \ C^-(u) \geq 0 \]

Then, the conservation condition (12) becomes

\[ (C_{i+1/2}^- - C_{i+1/2}^+) (u_{i+1}^n - u_i^n) = \lambda (g_{i+1}^n - g_i^n) \]

And equations (13) become

\[ \lambda f_{i+1/2}^n = -C_{i+1/2}^+ (u_{i+1}^n - u_i^n) + \lambda g_i^n, \quad \lambda f_{i-1/2}^n = -C_{i-1/2}^- (u_i^n - u_{i-1}^n) + \lambda g_i^n \]
- Focus is set here on explicit FT difference approximations
- Recall that unstable solutions exhibit *significant* spurious oscillations and/or overshoots
- Recall also that linear stability analysis focuses on these oscillations and relies on the Fourier series representation of the numerical solution: It requires only that this solution should not blow up, or more specifically, that each component in its Fourier series representation should not increase to infinity
  - because of linearity, this is equivalent to requiring that each component in the Fourier series should shrink by the same amount or stay constant at each time-step
Nonlinear Stability Analysis

Similarly, nonlinear stability analysis focuses on the spurious oscillations of the numerical solution, but without representing it by a Fourier series.

- It can require that the overall amount of oscillation remains bounded, which is known as the *Total Variation Bounded* (TVB) condition.
- It can also require that the overall amount of oscillation, as measured by the total variation, either shrinks or remains constant at each time-step (this is known as the *Total Variation Diminishing* (TVD) condition).
- However, whereas not blowing up and shrinking are equivalent notions for linear equations, these are different notions for nonlinear equations: In particular, TVD implies TVB but TVB does not necessarily imply TVD.
Monotonicity Preservation

- The solution of a scalar conservation law on an infinite spatial domain is *monotonicity preserving*: If the initial condition is monotone increasing (decreasing), the solution is monotone increasing (decreasing) at all times.

- Suppose that a numerical approximation inherits this monotonicity preservation property: Then, if the initial condition is monotone, the numerical solution cannot exhibit a spurious oscillation.

- Monotonicity preservation was first suggested by the Russian scientist Godunov in 1959: It is a nonlinear stability condition, but not a great one for the following reasons:
  - it does not address the case of nonmonotone solutions
  - it is a too strong condition:
    - it does not allow even an *insignificant* spurious oscillation that does not threaten numerical stability
    - attempting to purge all oscillatory errors, even the small ones, may cause much larger nonoscillatory errors

- Godunov’s theorem: For linear methods (*NOT to be confused with linear problems*), *monotonicity preservation leads to first-order accuracy at best*
Nonlinear Stability Analysis

Total Variation Diminishing

- TVD was first proposed by the American applied mathematician Amiram Harten in 1983 as a nonlinear stability condition.

The total variation of the exact solution may be defined as follows:

\[ TV(u(\cdot, t)) = \sup_{\text{all possible sets of samples } x_i} \sum_{i=\infty}^{\infty} |u(x_{i+1}, t) - u(x_i, t)| \]

- Laney and Caughey (1991):
  - the total variation of a function on an infinite domain is a sum of extrema — maxima counted positively and minima counted negatively — with the two infinite boundaries always treated as extrema and counting each once, and every other extrema counting twice.
Nonlinear Stability Analysis

Total Variation Diminishing

\[
T V = u_{\infty} - u_{\infty}
\]

\[
T V = -2 u_{\min} + u_{\infty} + u_{\infty}
\]

\[
T V = 2 u_{\max} - u_{\infty} - u_{\infty}
\]

\[
T V = 6 u_{\max} - 6 u_{\min} + u_{\infty} - u_{\infty}
\]

\[
= 7 u_{\max} - 7 u_{\min}
\]
Numerical effects that can cause the total variation to increase

- Transforming a monotone region into one with a maximum and a minimum
- Splitting a maximum into two maxima and a minimum
- Amplifying a maximum
- Decreasing a minimum
Nonlinear Stability Analysis

Total Variation Diminishing

- The exact solution of a scalar conservation law is TVD

\[ TV (u(\cdot, t_2)) \leq TV (u(\cdot, t_1)), \quad \forall t_2 \geq t_1 \]

- What about the numerical solution of a scalar conservation law?
- The total variation of a numerical approximation at time \( t^n \) may be equally defined as

\[ TV(u^n) = \sum_{i=-\infty}^{\infty} |u^n_{i+1} - u^n_i| \]

  - it is the sum of extrema — maxima counted positively and minima counted negatively — with the two infinite boundaries always treated as extrema and counting each once, and every other extrema counting twice

- Now, a numerical approximation inherits the TVD property if

\[ \forall n, \ TV (u^{n+1}) \leq TV(u^n) \]
Important result: *TVD implies monotonicity preservation* and therefore implies nonlinear stability

Proof: Suppose that the initial condition is monotone

- the TV of the initial condition is $u_\infty - u_{-\infty}$ if it is monotone increasing and $u_{-\infty} - u_\infty$ if it is monotone decreasing
- if the numerical solution remains monotone, $TV = cst$; otherwise, it develops new maxima and minima causing the TV to increase
- if the approximation method is TVD, this cannot happen and therefore the numerical solution remains monotone

TVD can be a stronger nonlinear stability condition than the monotonicity preserving condition
Nonlinear Stability Analysis

Total Variation Diminishing

- Drawback: Clipping phenomenon (illustrated with the linear advection of a triangle-shaped initial condition)

\[ \text{The TV should increase by } \Delta x \text{ between time-steps but a TVD scheme will not allow this } \Rightarrow \text{clipping error (here this error is } O(\Delta x) \text{ because it happens at a nonsmooth maximum, but for most smooth extrema it is } O(\Delta x^2) \text{)} \]
Summary of what should be known about TVD

- In practice, most attempts at constructing a TVD scheme end up enforcing stronger nonlinearity stability conditions such as the positivity condition discussed next.

- TVD implies monotonicity preservation: This is desirable when monotonicity preservation is too weak but less desirable when monotonicity preservation is too strong given that TVD can be stronger.

- TVD tends to cause clipping errors at extrema: In theory, clipping does not need to occur at every extrema — since, for example, the local maximum could increase provided that a local maximum decreased or a local minimum increased or a local maximum-minimum pair disappeared somewhere else — and may be only moderate when it occurs: However, in practice, most TVD schemes clip all extrema to between first- and second-order accuracy.

- In theory, TVD may allow large spurious oscillations but in practice it rarely does — in any case, it does not allow the unbounded growth type of instability.
Recall that the wave speed split form of a FT scheme is given by

\[ u_{i}^{n+1} = u_{i}^{n} + C_{i+1/2}^{+n} (u_{i+1}^{n} - u_{i}^{n}) - C_{i-1/2}^{-n} (u_{i}^{n} - u_{i-1}^{n}) \]

\[ C_{i+1/2}^{+n} \geq 0 \quad \text{and} \quad C_{i+1/2}^{-n} \geq 0 \]

Suppose that a given FT numerical scheme can be written in wave speed split form with

\[ C_{i+1/2}^{+n} \geq 0, \quad C_{i+1/2}^{-n} \geq 0 \quad \text{and} \quad C_{i+1/2}^{+n} + C_{i+1/2}^{-n} \leq 1 \quad \forall i \quad (15) \]

Condition (15) above is called the *positivity condition* (also proposed first by Harten in 1983)

What is the connection between the positivity condition and the nonlinear stability of a scheme?
Nonlinear Stability Analysis

Positivity

- The answer is: The positivity condition implies TVD
- Example: FTFS applied to the nonlinear advection equation
  \[ \frac{\partial u}{\partial t} + a(u) \frac{\partial u}{\partial x} = 0 \]
  is positive if \(-1 \leq \lambda a_{i+1/2}^n \leq 0\)
- Proof:
  - FTFS can be written in wave speed split form for the purpose of nonlinear stability analysis as follows
    \[ u_{i+1}^n = u_i^n + C_{i+1/2}^+(u_{i+1}^n - u_i^n) - C_{i-1/2}^-(u_i^n - u_{i-1}^n) \]
    where \(C_{i+1/2}^+ = -\lambda a_{i+1/2}^n\) and \(C_{i-1/2}^- = 0\)
  - \(\lambda a_{i+1/2}^n \leq 0 \Rightarrow C_{i+1/2}^+ = -\lambda a_{i+1/2}^n \geq 0\)
  - \(C_{i+1/2}^+ + C_{i+1/2}^- = -\lambda a_{i+1/2}^n\) and therefore the condition (15) becomes in this case \(-1 \leq \lambda a_{i+1/2}^n \leq 0\)
  - also, note that the positivity condition is in this case equivalent to the CFL condition
The extension to multiple dimensions of the *computational* part of the material covered in this chapter may be tedious in some cases but is straightforward (except perhaps for the characteristic theory).

The expressions of the Euler equations in 2D and 3D can be obtained from Chapter 2 (as particular cases of the expression of the Navier-Stokes equations in 3D).

For simplicity, the focus is set here on the 2D Euler equations

\[
\frac{\partial W}{\partial t} + \frac{\partial F_x}{\partial x}(W) + \frac{\partial F_y}{\partial y}(W) = 0
\]
2D structured grid
Multidimensional Extensions

\[ \frac{\partial W}{\partial t} + \frac{\partial F_x}{\partial x}(W) + \frac{\partial F_y}{\partial y}(W) = 0 \]

For the above 2D Euler equations, the equivalent of equation (2) on a 2D structured grid is

\[ \Delta t \left( \frac{\partial W}{\partial t} \right)_{i,j}^n = -\lambda_x (\hat{F}_{x_{i+1/2,j}}^n - \hat{F}_{x_{i-1/2,j}}^n) - \lambda_y (\hat{F}_{y_{i,j+1/2}}^n - \hat{F}_{y_{i,j-1/2}}^n) \]

where

\[ \lambda_x = \frac{\Delta t}{\Delta x_i}, \quad \lambda_y = \frac{\Delta t}{\Delta y_j} \]

\[ \Delta x_i = x_{i+1/2,j} - x_{i-1/2,j} \quad \forall j, \quad \Delta y_j = y_{i,j+1/2} - y_{i,j-1/2} \quad \forall i \]

and \( \hat{F}_{x_{i+1/2,j}} \) and \( \hat{F}_{y_{i,j+1/2}} \) are constructed exactly like \( \hat{f}_{i+1/2} \) in 1D.
For example, a 2D version of FTCS has the following conservative numerical fluxes

\[
\hat{F}_{n}^{x}_{i+1/2,j} = \frac{1}{2} \left( \mathcal{F}_x(W_{i+1,j}^n) + \mathcal{F}_x(W_{i,j}^n) \right)
= \frac{1}{2} \begin{pmatrix}
(\rho v_x)_{i+1,j} + (\rho v_x)_{i,j} \\
(\rho v_x^2)_{i+1,j} + (\rho v_x^2)_{i,j} + p_{i+1,j} + p_{i,j}
\end{pmatrix}
\]

\[
\hat{F}_{n}^{y}_{i,j+1/2} = \frac{1}{2} \left( \mathcal{F}_y(W_{i,j+1}^n) + \mathcal{F}_y(W_{i,j}^n) \right)
= \frac{1}{2} \begin{pmatrix}
(\rho v_y)_{i,j+1} + (\rho v_y)_{i,j} \\
(\rho v_x v_y)_{i,j+1} + (\rho v_x v_y)_{i,j}
\end{pmatrix}
\]