Simulation Astronomy

(Advection Equation)

Kazunari Iwasaki

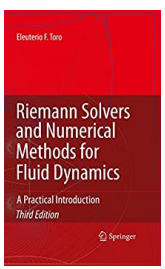
November 15, 2024

References

- 「シミュレーション天文学」富阪幸治、花輪知幸、牧野淳一郎 編

- 「流体力学の数値計算法」藤井孝蔵 著

- “Riemann Solvers and Numerical Methods for Fluid Dynamics” Eleuterio F. Toro



Basic Equation: Euler Equation

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho v_i) = 0$$

$$\frac{\partial (\rho v_i)}{\partial t} + \frac{\partial}{\partial x_j}\left(P\delta_{ij} + \rho v_i v_j\right) = 0$$

$$\frac{\partial E}{\partial t} + \frac{\partial}{\partial x_i}\left[(E+P)\, v_i\right] = 0$$

- 6 Unknown variables (ρ, P, E, v_x, v_y, v_z)
- 5 equations (1 (continuity), 3 (momentum), 1 (energy))

$\longrightarrow$ One more equation is needed

Use a relation between ρ, the internal energy $e = E - \rho v^2/2$, and P (Equation of state)

$$e = e(\rho, P)$$

For an adiabatic gas with a spacific ratio of heat γ, $e = P/(\gamma - 1)$

A Most Simplified Model of Hydrodynamic Equations

Hydrodynamic equations are too complicated.
Many variables and many equations...

Let's start from a simplified equation whose behavior is well known.

A Most Simplified Model of Hydrodynamic Equations

Hydrodynamic equations are too complicated.
Many variables and many equations...

Let's start from a simplified equation whose behavior is well known.

Roughly speaking, the hydrodynamic equations have a form of

$$\frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot (u\boldsymbol{v}) = 0$$

u can be $\rho, \rho\boldsymbol{v}, \rho v^2/2 + e$

A Most Simplified Model of Hydrodynamic Equations

Hydrodynamic equations are too complicated.
Many variables and many equations...

Let's start from a simplified equation whose behavior is well known.

Roughly speaking, the hydrodynamic equations have a form of

$$\frac{\partial u}{\partial t} + \nabla \cdot (u\boldsymbol{v}) = 0$$

u can be $\rho, \rho\boldsymbol{v}, \rho v^2/2 + e$

make further simplification

- one dimension
- velocity is constant $v = c$. $\longrightarrow$ becomes a linear equation for u.

Advection Equation (移流方程式)

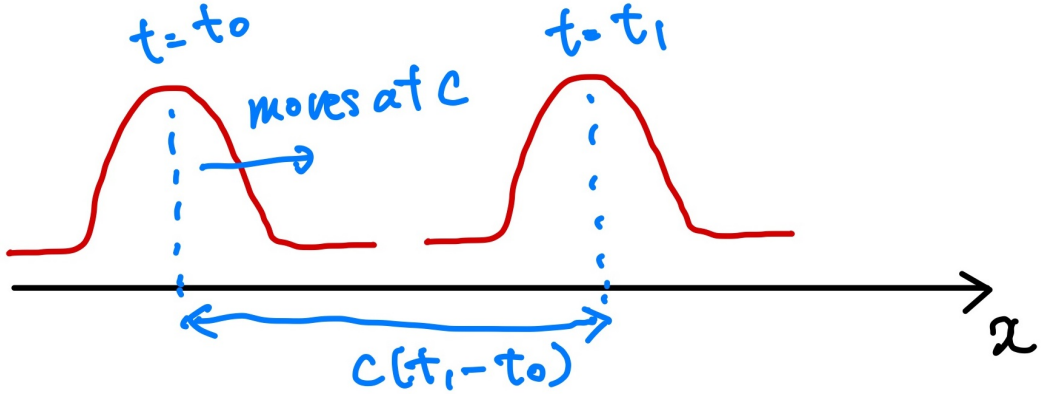
$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(cu) = 0$$

General Solution of Advection Equation

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(cu) = 0$$

The general solution is $u(t, x) = u(x - ct)$.
$\rightarrow$ The performance of numerical schemes can be easily tested.

u is propagated keeping its shape unchanged at a speed of c.



We will now consider how to solve this equation numerically!
The superiority of a numerical method can be determined by comparing the numerical results with the exact solution.

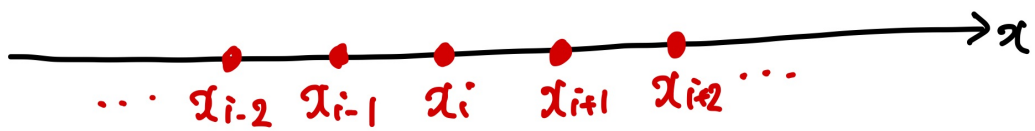
Finite-difference Methods (有限差分法)

Space and time are both descritized.

Descritization of Space

Discrete grid points are placed in the x-axis.

- The coordinate of the i-th grid is denoted as x_i.
- In this lecture, equally-spaced grid points are considered. $\Delta x = x_{i+1} - x_i \quad \forall i$



Descritization of Time

Discrete timesteps are considered.

- The time at the n-th timestep is dentoed as t^n.
- The value of u at $x = x_i$ at $t = t^n$ is denoted as u_i^n.

Finite-difference Method (有限差分法)

Definition of Derivative

$$\frac{\partial u}{\partial x} \equiv \lim_{\Delta x \to 0} \frac{u(x + \Delta x) - u(x)}{\Delta x}$$

Derivative $\partial u/\partial x$ is approximated by using u defined at discrete grids.

An Example of Finite Difference (Forward Difference)

$$\left(\frac{\partial u}{\partial x}\right)_i^n = \frac{u_{i+1}^n - u_i^n}{\Delta x}$$

To compute $\partial u/\partial x$ at x_i, u at x_{i+1} is used.

Accuracy of Finite Difference

How fast the differentiation error decreases as Δx decreases?

From Taylor expansion,

$$\begin{aligned} u_{i+1}^n &= u(x_i + \Delta x, t^n) \\ &= u(x_i) + \Delta x \left(\frac{\partial u}{\partial x}\right)_{x_i,t^n} + \frac{\Delta x^2}{2!}\left(\frac{\partial^2 u}{\partial x^2}\right)_{x_i,t^n} + O(\Delta x^3) \end{aligned}$$

Accuracy

$$\begin{aligned} \left(\frac{\partial u}{\partial x}\right)_i^n &= \frac{u_{i+1}^n - u_i^n}{\Delta x} \\ &= \left(\frac{\partial u}{\partial x}\right)_{x_i,t^n} + \frac{\Delta x}{2!}\left(\frac{\partial^2 u}{\partial x^2}\right)_{x_i,t^n} + O(\Delta x^2) \end{aligned}$$

The difference between the forward difference and exact drivative $\propto \Delta x^1$
$\Rightarrow$ 1st order spatial accuracy

Higher Order Approximation

Finite difference methods with higher orders are obtained by using Taylor expansion.

$$u_{i+1}^n = u(x_i) + \Delta x \left(\frac{\partial u}{\partial x}\right)_{x_i,t^n} + \frac{\Delta x^2}{2!}\left(\frac{\partial^2 u}{\partial x^2}\right)_{x_i,t^n} + \frac{\Delta x^3}{3!}\left(\frac{\partial^3 u}{\partial x^3}\right)_{x_i,t^n} + O(\Delta x^4)$$

$$u_{i-1}^n = u(x_i) - \Delta x \left(\frac{\partial u}{\partial x}\right)_{x_i,t^n} + \frac{\Delta x^2}{2!}\left(\frac{\partial^2 u}{\partial x^2}\right)_{x_i,t^n} - \frac{\Delta x^3}{3!}\left(\frac{\partial^3 u}{\partial x^3}\right)_{x_i,t^n} + O(\Delta x^4)$$

Central Difference

$$\frac{u_{i+1} - u_{i-1}}{2\Delta x} = \left(\frac{\partial u}{\partial x}\right)_{x_i,t^n} + \frac{\Delta x^2}{3!}\left(\frac{\partial^3 u}{\partial x^3}\right)_{x_i,t^n} + O(\Delta x^3)$$

$\Rightarrow$ 2nd order spatial accuracy

Finite differences with 3rd, 4th, ... orders are obtained.

Finite-difference Equation of Advection Equation

Both spatial and temporal derivatives $\longrightarrow$ forward difference

$$\left(\frac{\partial u}{\partial x}\right)_i \simeq \frac{u_{i+1}^n - u_i^n}{\Delta x}, \quad \left(\frac{\partial u}{\partial t}\right)_i \simeq \frac{u_i^{n+1} - u_i^n}{\Delta t}$$

Substituting them to the advection equation, one obtains the following finite difference equation

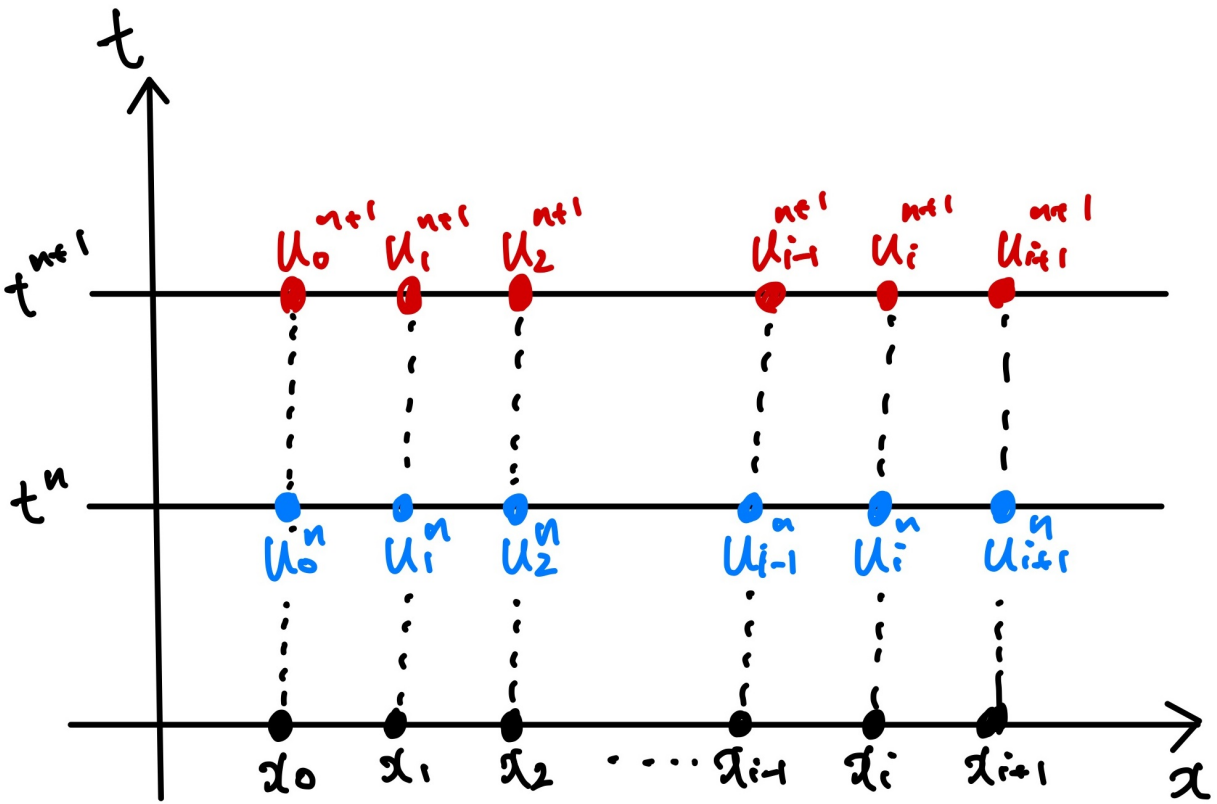
$$u_i^{n+1} = u_i^n - \frac{c\Delta t}{\Delta x}\left(u_{i+1}^n - u_i^n\right)$$

u_i is updated by only u_i at the previous timestep.

$\longrightarrow$ **Explicit solver (陽的解法)**

Explicit Solver (陽的解法)

- Given the initial condition u_i^0,
  $u_i^0 \longrightarrow u_i^1 \longrightarrow u_i^2, \ldots \longrightarrow u_i^n \longrightarrow \ldots$

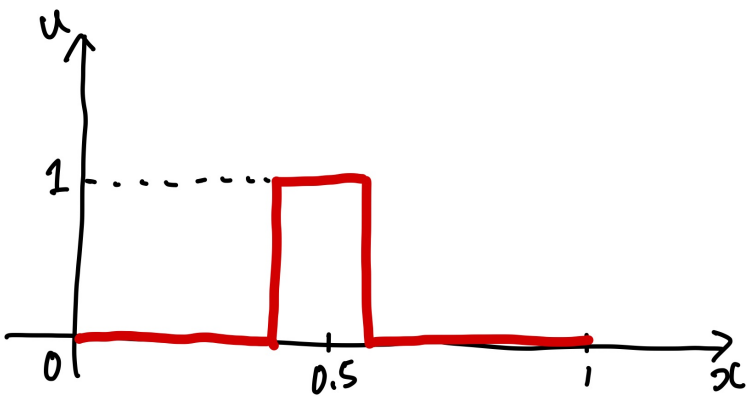


Numerical Experiments

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

- Simulation Box $0 \le x \le L$ $(L = 1)$
- Number of grids: $N = 64 \longrightarrow \Delta x = L/N = 0.015625$
- Advection speed $c = 1$ $(> 0$, propagate rightward)
- Initial profile: Square wave

$$u^0(x) = \begin{cases} 1 & \text{for } |x - 1/2| < 0.1 \\ 0 & \text{otherwise} \end{cases}$$



Google Colaboratory

What is Google Colaboratory?

- a free cloud service provided by Google.
- make and run python scripts through a browser.

The sample python scripts provided in this lecture were written in a C-like grammar for readability.

If you are interested, please modify the sample codes so that they run faster.

Calculation Procedure

Use 1D array to save "u".

- `uold[i]` to store u_i^n.
- `u[i]` to store u_i^{n+1}.

In python scripts, equal "=" is not identical to mathematical "equal".
`"x = 1"` shows that number "1" is substituted to the variable "`x`" that is allocated in a region of memory.

First, the initial profile of u_i is substituted into the array `u[i]`.
Then, the main loop starts.

1. `uold[i] = u[i]`
   - The number stored in `u[i]` is substituted in `uold[i]`.
   - After this line is executed, `uold[i]` is equivalent to u_i^n.
2. `u[i] = uold[i] - c*dt/dx*(uold[i+1] - uold[i])`
   - `u[i]` is derived from `uold[i]`.
     After this line is executed, `u[i]` is u_i^{n+1}.

   back to 1.

Results

What do you think?

Backward Difference

Another approximation of $\partial u/\partial x$,

$$\left(\frac{\partial u}{\partial x}\right)_i \simeq \frac{u_i - u_{i-1}}{\Delta x}$$

The corresponding finite-difference equation is

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

In the sample code,

```
u[i] = uold[i] - c*dt/dx*(uold[i+1] - uold[i])
```

is changed to

```
u[i] = uold[i] - c*dt/dx*(uold[i] - uold[i-1])
```

.

Then, run the simulation.

Results

What do you think?

Central Difference

Next, the central difference is considered.

$$\left(\frac{\partial f}{\partial x}\right)_i \simeq \frac{u_{i+1} - u_{i-1}}{2\Delta x}$$

The finite difference equation for the advection equation is

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

Modify the sample code, and run the simulation.

Results

What do you think?

Three Important Properties of Numerical Schemes:

Consistency (適合性), Stability (安定性), and Convergence (収束性)

Consistency

A numerical scheme has consistency if it converges to the original differential equation in the limit $\Delta x, \Delta t \to 0$.

Stability

A scheme is stable if any errors (truncation errors and/or truncation errors) do not grow.

Convergence

As Δx and Δt decreases, the numerical solutions derived by a numerical scheme approaches the exact solutions of the original differential equation.

Lax Equivalence Theorem (Laxの等価定理)

(Lax and Richtmeyer 1959)

Lax & Richtmeyer proved that

a consistent linear scheme is convergent if and only if the scheme is stable.

適合性をもつ線型スキームは安定な場合にのみ収束する

What we observed in the numerical experiments are

- **Consistency:** Forward ◯, Backward ◯, Central ◯
- **Stability:** Forward ×, Backward ◯, Central ×

Lax Equivalence Theorem (Laxの等価定理)

(Lax and Richtmeyer 1959)

Lax & Richtmeyer proved that

a consistent linear scheme is convergent if and only if the scheme is stable.

適合性をもつ線型スキームは安定な場合にのみ収束する

What we observed in the numerical experiments are

- **Consistency:** Forward ◯, Backward ◯, Central ◯
- **Stability:** Forward ×, Backward ◯, Central ×
- **Convergence (from Lax theorem):** Forward ×, Backward ◯, Central ×

Lax Equivalence Theorem (Laxの等価定理)

(Lax and Richtmeyer 1959)

Lax & Richtmeyer proved that

a consistent linear scheme is convergent if and only if the scheme is stable.

適合性をもつ線型スキームは安定な場合にのみ収束する

What we observed in the numerical experiments are

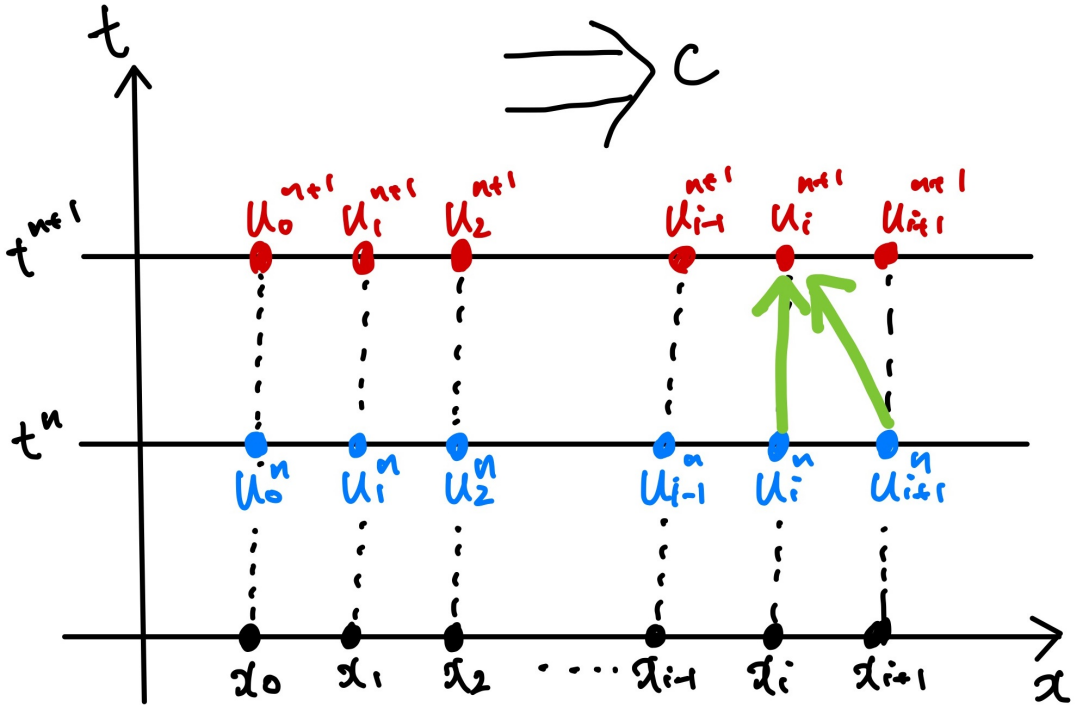
- **Consistency:** Forward ◯, Backward ◯, Central ◯
- **Stability:** Forward ×, Backward ◯, Central ×

Why are the forward-difference method unstable and backward-difference method is stable, even though both have consistency?

Direction of "Wind"

Propagation of waves is often expressed as "Wind".

For $c > 0$, "wind" blows left to right.

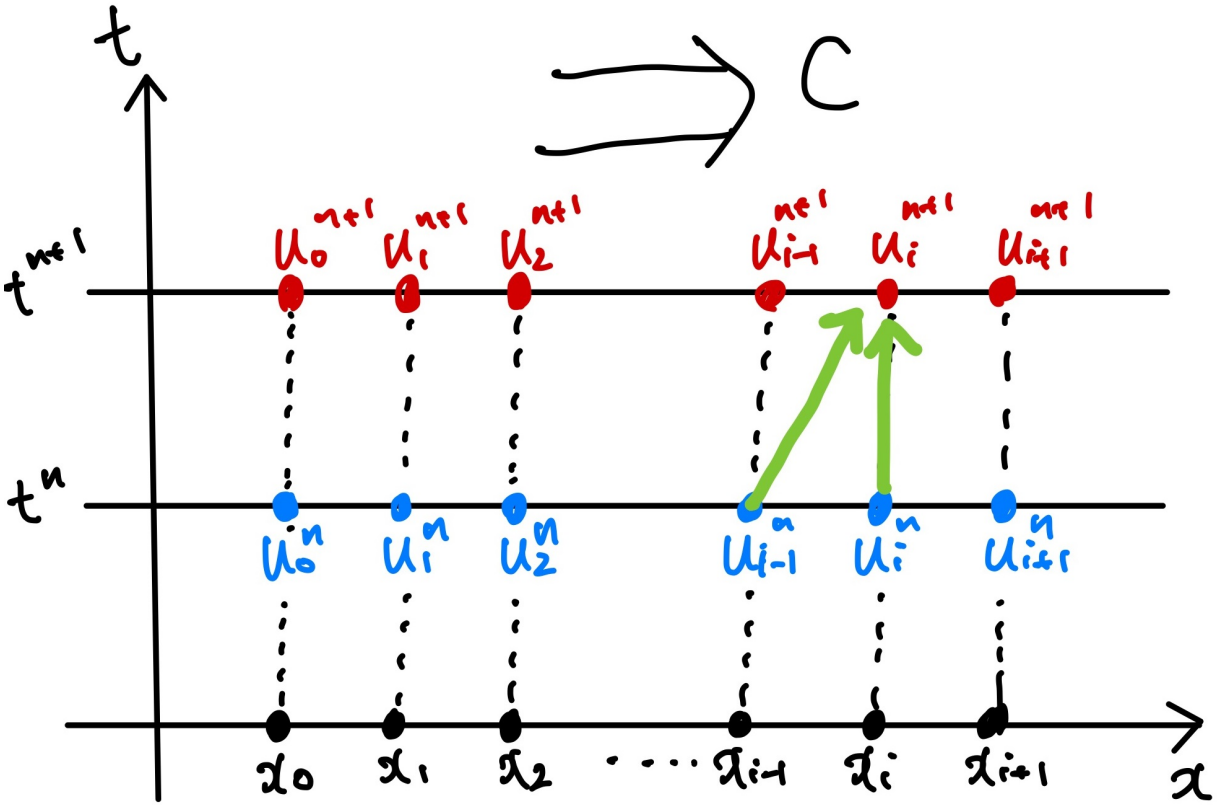


$$u_i^{n+1} = u_i^n - \frac{c\Delta t}{\Delta x}\left(u_{i+1}^n - u_i^n\right)$$

During $t^n \le t \le t^{n+1}$, $u(x_i)$ should be determined only by $u(x)$ in $x \le x_i$ because "wind" blows left to right.

$\Rightarrow$ the forward difference scheme uses u in downwind (風下, u_{i+1}) to update u_i.

Upwind (風上) Scheme



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

u_i is updated using grids on the side from which wind blows (upwind).

Accuracy of Finite Difference Methods

Truncation Error

Difference between finite difference equation and exact equation.

$$u_{i+1}^n = u_i^n + \left(\frac{\partial u}{\partial x}\right)_i^n \Delta x + \frac{1}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n \Delta x^2 \text{ and } u_i^{n+1} = u_i^n + \left(\frac{\partial u}{\partial t}\right)_i^n \Delta t + \frac{1}{2}\left(\frac{\partial^2 u}{\partial t^2}\right)_i^n \Delta t^2$$

Substituting them to the difference equation $u_i^{n+1} = u_i^n - (c\Delta t/\Delta x)(u_{i+1}^n - u_i^n)$,

$$\begin{aligned}\left(\frac{\partial u}{\partial t}\right)_i^n + c\left(\frac{\partial u}{\partial x}\right)_i^n &= -\frac{1}{2}\left(\frac{\partial^2 u}{\partial t^2}\right)_i^n \Delta t - \frac{c}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n \Delta x \\ &= -\frac{c}{2}(c\Delta t + \Delta x)\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n\end{aligned}$$

If the truncation error $\epsilon \propto O(\Delta t^p)$ and $O(\Delta x^p)$
$\Rightarrow$ p-th order accuracy in space and time.
$\Rightarrow$ **Forward difference scheme has 1st-order accuracy**

Behavior of Truncation Errors

$$\left(\frac{\partial u}{\partial t}\right)_i^n + c\left(\frac{\partial u}{\partial x}\right)_i^n = -\frac{c}{2}\left(c\Delta t + \Delta x\right)\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n$$

Behavior of Truncation Errors

$$\left(\frac{\partial u}{\partial t}\right)_i^n + c\left(\frac{\partial u}{\partial x}\right)_i^n = -\frac{c}{2}\left(c\Delta t + \Delta x\right)\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n$$

The right-hand side representing the truncation error
that works as diffusion with diffusion coefficient $-c(c\Delta t + \Delta x)/2$.

Behavior of Truncation Errors

$$\left(\frac{\partial u}{\partial t}\right)_i^n + c\left(\frac{\partial u}{\partial x}\right)_i^n = -\frac{c}{2}(c\Delta t + \Delta x)\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n$$

The right-hand side representing the truncation error
that works as diffusion with diffusion coefficient $-c(c\Delta t + \Delta x)/2$**.**

- For $c > 0$
  - Anti-diffusion $\Longrightarrow$ infinitesimally small variations of u grow
    $\Longrightarrow$ **Always unstable**
  - **Consistent with the fact that the forward difference scheme is "downwind" scheme for** $c > 0$

Behavior of Truncation Errors

$$\left(\frac{\partial u}{\partial t}\right)_i^n + c\left(\frac{\partial u}{\partial x}\right)_i^n = -\frac{c}{2}\left(c\Delta t + \Delta x\right)\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n$$

The right-hand side representing the truncation error
that works as diffusion with diffusion coefficient $-c(c\Delta t + \Delta x)/2$.

- For $c > 0$
  - Anti-diffusion $\Longrightarrow$ infinitesimally small variations of u grow
    $\Longrightarrow$ **Always unstable**
  - **Consistent with the fact that the forward difference scheme is "downwind" scheme for $c > 0$**
- For $c < 0$
  - Diffusion coefficient: $D = |c|\left(-|c|\Delta t + \Delta x\right)$
  - When $-|c|\Delta t + \Delta x > 0$, D is positive. $\Longrightarrow$ **conditionally stable**
  - **Consistent with the fact that the forward difference scheme is "upwind" scheme for $c < 0$**
    - Note that $-|c|\Delta t + \Delta x > 0$ is called CFL condition.
      see next slide

Courant-Friedrichs-Lewy (CFL) Condition

To obtain stable results from u_{i-1}^n and u_i^n, $\boxed{c\Delta t < \Delta x}$ should be satisfied.

Otherwise, $u(x_i)$ is affected by the grids $x < x_{i-1}$ that are not taken into account in the 1st order upwind scheme.

CFL Condition

$$\Delta t = C_{\mathrm{CFL}} \frac{\Delta x}{c}$$

The distance traveled in Δt shold be smaller than Δx. $C_{\mathrm{FLD}} \leq 1$ is a free parameter.

An essentially same condition is used in numerical (magneto)hydrodynamics.

Finite-volume Method (有限体積法)

Hydrodynamic equations are often expressed as differential form (微分形).
$\Longrightarrow$ finite-difference methods (derivatives are approximated by finite difference)

They can be expressed as integral form (積分形)

For instance, mass conservation

$$\frac{\partial}{\partial t}\int_V \rho d^3x = -\oint_S \rho \boldsymbol{v}\cdot d\boldsymbol{S}$$

The total mass in a volume fixed in the space varies with time by the mass flux across the boundary surface.

Finite-volume methods are based on the integral forms of equations

- Widely used in most astrophysical simulations because it has important properties.
  - Conservative quantities (such as mass, momentum, total energy) are constant during simulations within round-off errors.
    (no outflow and no inflow across the simulation box boundary)
  - It has an affinity with Godunov method (will be explained in the next lecture).

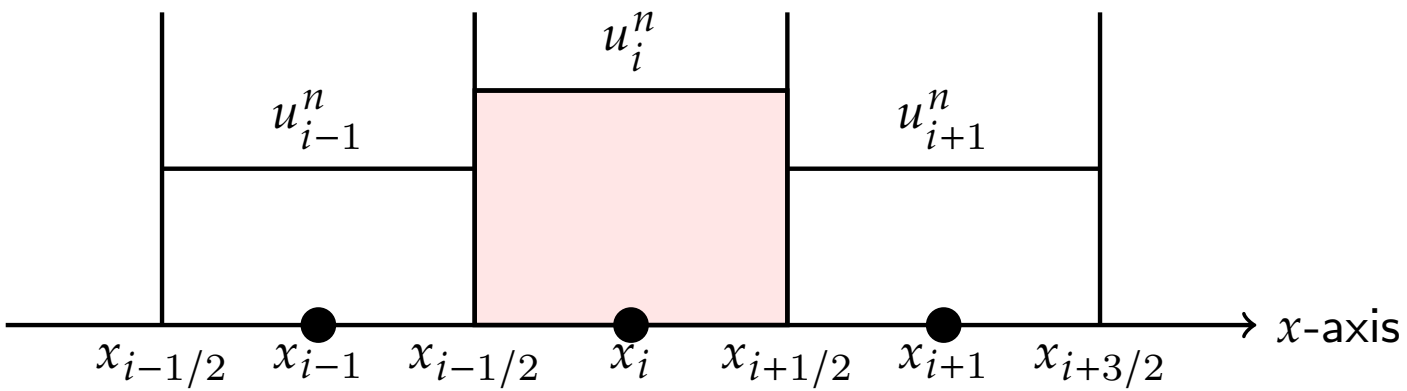
Finite-volume Method (有限体積法)

Conservation form

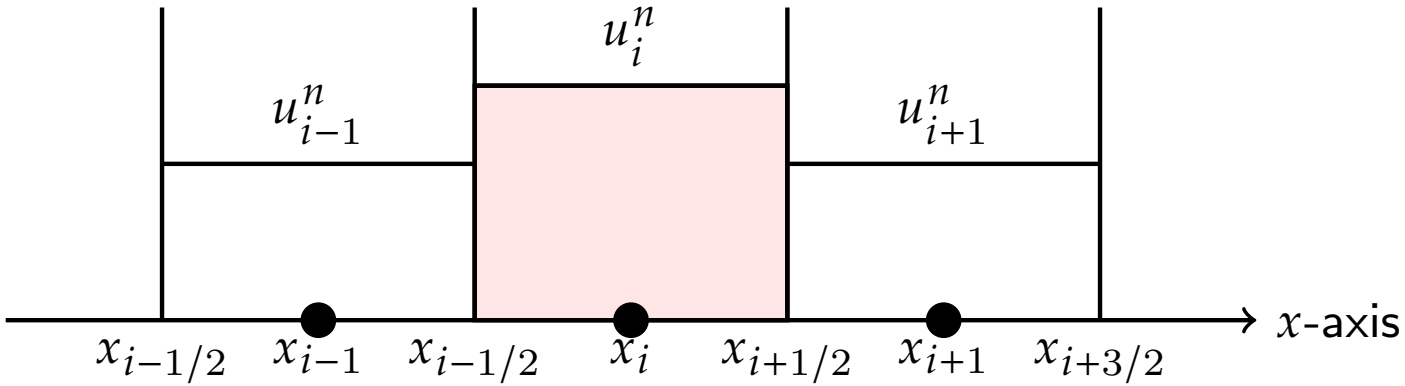
$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = 0,$$

In FV method, the space is divided not by grid points but by cells (small rooms) separated by walls.

x_i: The central coordinate of the i-th cell
$x_{i+1/2}$: the cell boundary between the i-th and $(i+1)$-th cells.



Finite-volume Method

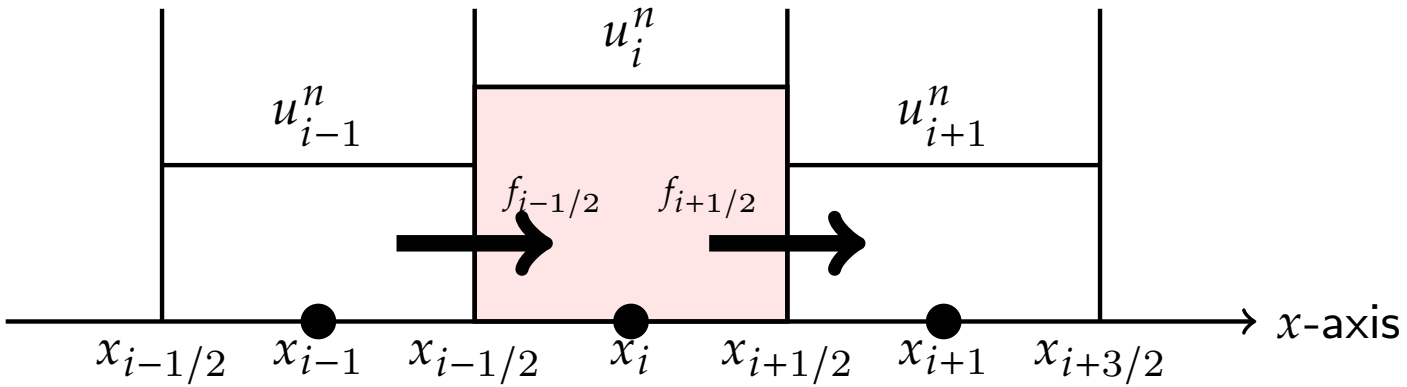


Integrating $\partial u/\partial t + \partial f/\partial x = 0$ over $x_{i-1/2} \le x \le x_{i+1/2}$, one obtains

$$\frac{\partial}{\partial t}\left(\int_{x_{i-1/2}}^{x_{i+1/2}} u dx\right) = -\left(f_{i+1/2}(t) - f_{i-1/2}(t)\right)$$

$$\frac{\partial u_i}{\partial t} = -\frac{1}{\Delta x}\left(f_{i+1/2}(t) - f_{i-1/2}(t)\right), \quad \text{where} \quad u_i(t) \equiv \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t)dx$$

Finite-volume Method (有限体積法)



Integrating the equation over $t^n \le t \le t^{n+1}$ gives

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\left(f_{i+1/2} - f_{i-1/2}\right), \quad \text{where} \quad f_{i+1/2} \equiv \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f_{i+1/2}(t)dt$$

The time evolution of u in the i-th cell is determined by
the difference between the fluxes across the surfaces of the i-th cell.
$\Longleftarrow$ conservation law

Finite-volume Method of Advection Equation

Various methods have been proposed to evaluate numerical fluxes $f_{i+1/2}$.

Of course, $f_{i+1/2} = (cu_i^n + cu_{i+1}^n)/2$ gives unstable results because the resultant difference equation is equivalent to the central difference equation.

Finite-volume Method of Advection Equation

Various methods have been proposed to evaluate numerical fluxes $f_{i+1/2}$.

Of course, $f_{i+1/2} = (cu_i^n + cu_{i+1}^n)/2$ gives unstable results because the resultant difference equation is equivalent to the central difference equation.

1st order upwind numerical flux.

$$f_{i+1/2}^{n+1/2} = \begin{cases} cu_i^n & \text{for } c > 0 \\ cu_{i+1}^n & \text{for } c < 0 \end{cases}$$

The numerical flux is evaluated by using u in the upwind side.

The following expression can be used for any c,

$$\begin{aligned} f_{i+1/2}^{n+1/2} &= \frac{c - |c|}{2} u_{i+1}^n + \frac{c + |c|}{2} u_i^n \\ &= \frac{1}{2}\left[cu_i^n + cu_{i+1}^n - |c|(u_{i+1}^n - u_i^n)\right] \end{aligned}$$

Calculation Procedure

Use 1D array to save "u".

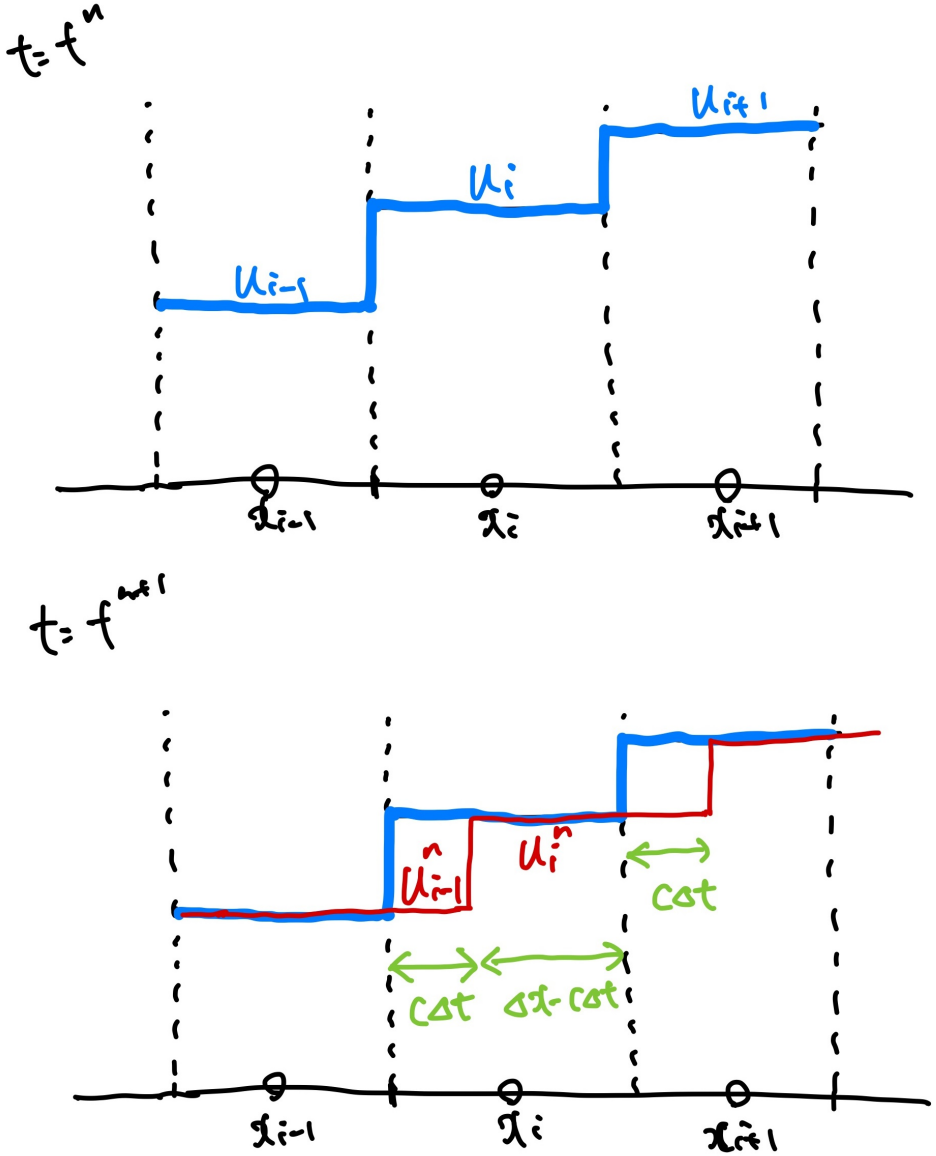
- `u[i]` to store u_i^n, u_i^{n+1}.
- `f[i]` to store $f_{i+1/2}$

First, the initial profile of u_i is substituted into the array `u[i]`.
Then, the main loop starts.

1. `f[i] = 0.5*( c*u[i] + c*u[i+1] - abs(c)*(u[i+1] - u[i]) )`
   - Calculate the numerical flux at $x_{i+1/2}$
2. `u[i] = u[i] - dt/dx*(f[i] - f[i-1])`
   - `u[i]` is derived from `f[i]`.
     After this line is executed, `u[i]` is u_i^{n+1}.

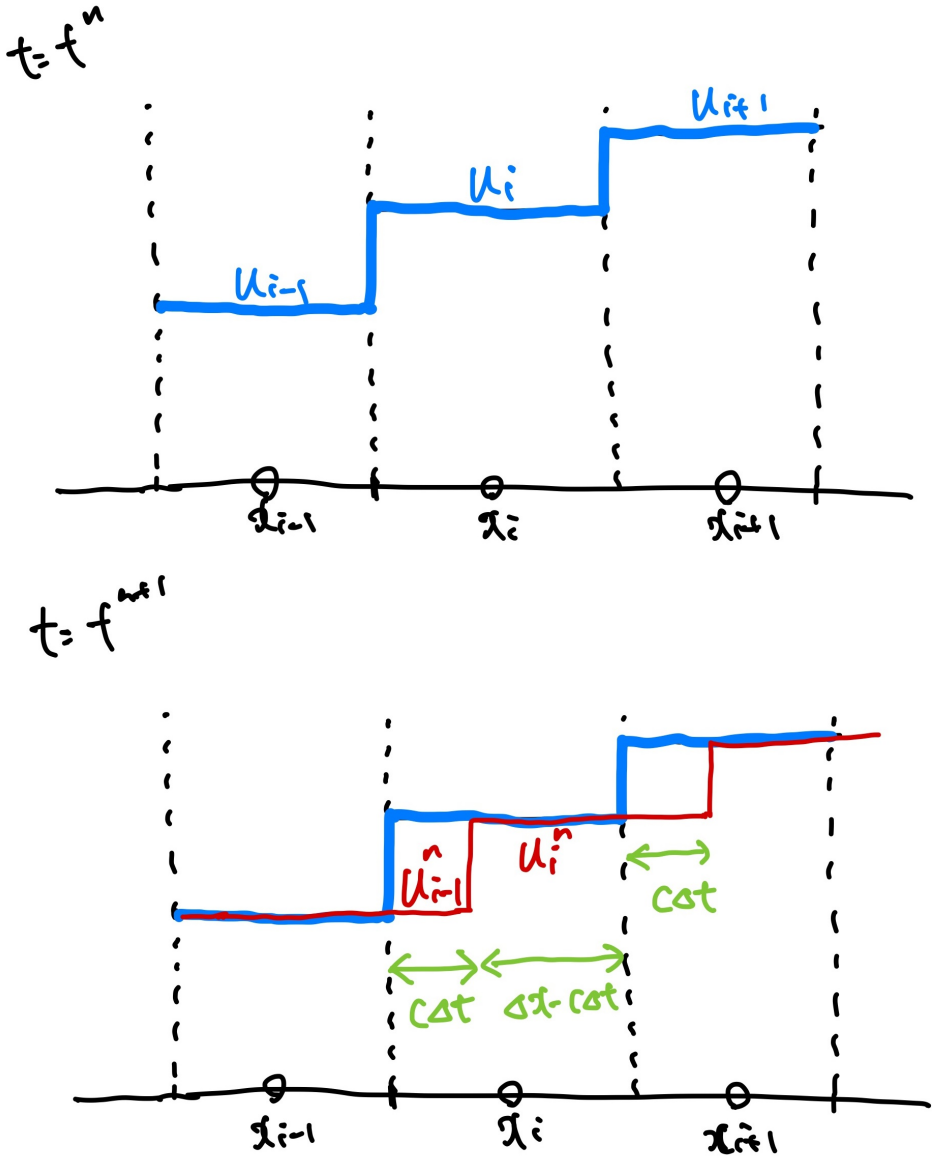
   back to 1.

Physical Interpretation of Upwind difference Scheme in Finite volume Method



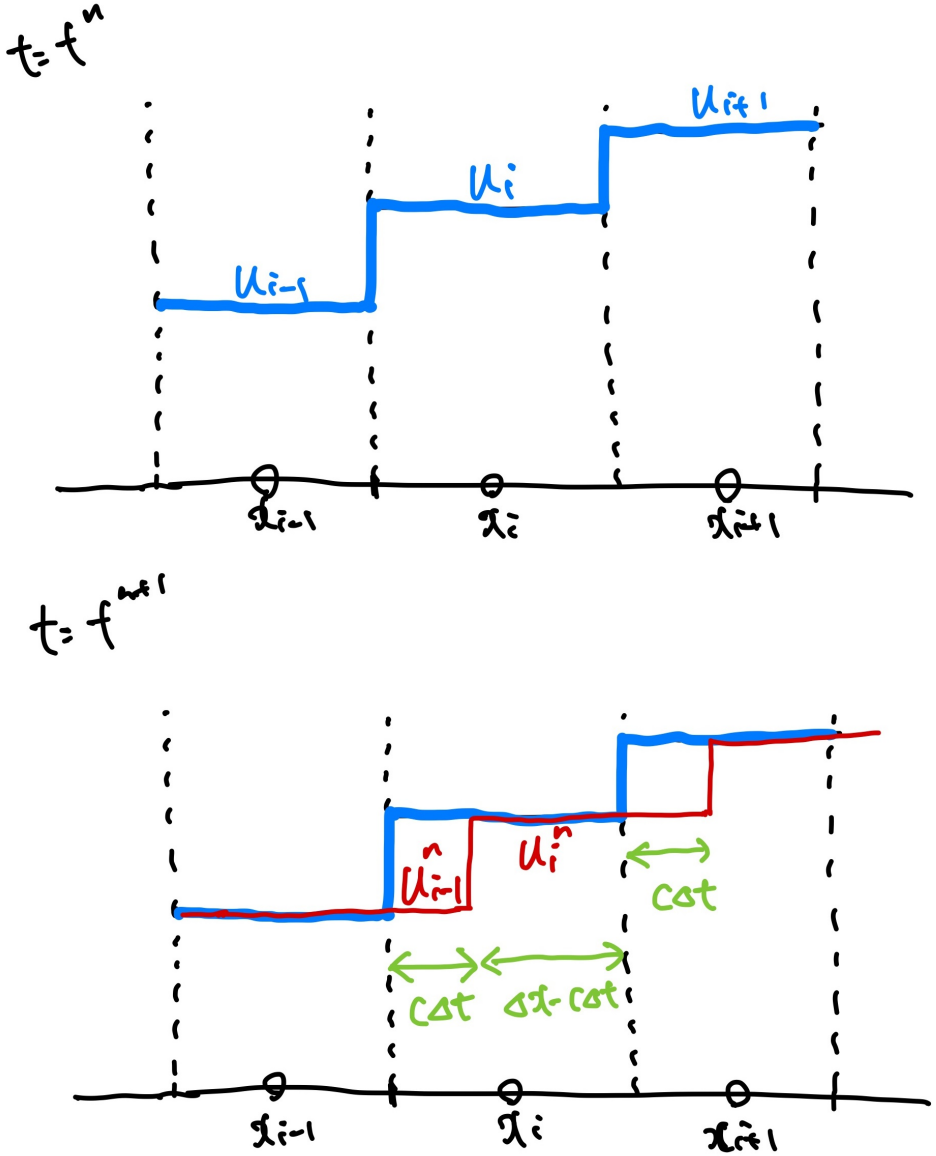
- At $t = t^n$, $u(x)^n$ is supposed to be expressed by the step functions.
  (u is constant inside each cell.)

Physical Interpretation of Upwind difference Scheme in Finite volume Method



- At $t = t^n$, $u(x)^n$ is supposed to be expressed by the step functions. (u is constant inside each cell.)
- At $t = t^{n+1}$, the exact solution is shown by the red line.

Physical Interpretation of Upwind difference Scheme in Finite volume Method



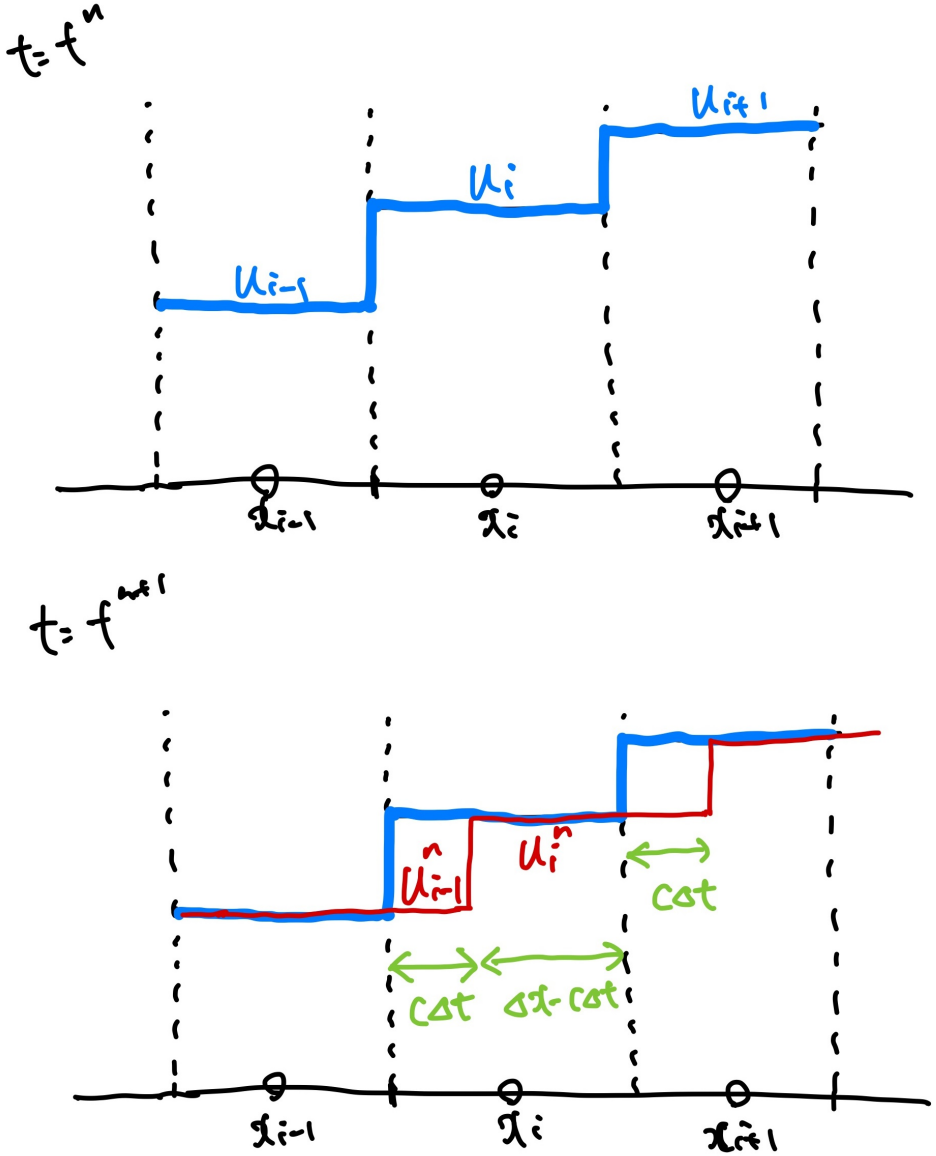
- At $t = t^n$, $u(x)^n$ is supposed to be expressed by the step functions.
  (u is constant inside each cell.)
- At $t = t^{n+1}$, the exact solution is shown by the red line.
- $u_i^{n+1}\Delta x$ should be equal to $u_{i-1}^n c\Delta t + u_i^n (\Delta x - c\Delta t)$.

Then, we obtain

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

This equation is the same as the 1st-order upwind scheme for $c > 0$.

Physical Interpretation of Upwind difference Scheme in Finite volume Method



- At $t = t^n$, $u(x)^n$ is supposed to be expressed by the step functions.
  (u is constant inside each cell.)
- At $t = t^{n+1}$, the exact solution is shown by the red line.
- $u_i^{n+1}\Delta x$ should be equal to $u_{i-1}^n c\Delta t + u_i^n (\Delta x - c\Delta t)$.

Then, we obtain

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

This equation is the same as the 1st-order upwind scheme for $c > 0$.

This argument is essentially the same as that of Godunov method.

Report (Exercise 1)

1. Copy the sample code of the finite difference method in your google drive. Rename and edit the new code.
   - Change the initial profile to

$$u(x, t = 0) = \exp\left\{-\left(\frac{x - 0.5}{0.1}\right)^2\right\}$$

   Edit `InitialProfile(x)`.
   use `np.exp(x)` to use exponential function where "`np`" stands for the `numpy` module.

Report (Exercise 1)

2. Performs convergence test for the backward-difference method with $c > 0$ by changing the grid number N.
   - Use a table and/or plot to explain how fast the error decreases as N increases.
   - Edit function `AnalysisAfterSimulation(time,u)` that is called after the main loop finishes. In this function, calculate an error that is a measure of the difference between the simulation results and the exact solution.

   Any error expression is acceptable, as long as it is reasonable.
   For instance, the following expressions are often used.

$$\epsilon_1 = \frac{1}{N}\sum_i |u_i^n - u_{\text{exact}}(x_i, t^n)|$$

$$\epsilon_2 = \sqrt{\frac{1}{N}\sum_i \left\{u_i^n - u_{\text{exact}}(x_i, t^n)\right\}^2}$$

$$\epsilon_\infty = \max_i |u_i^n - u_{\text{exact}}(x_i, t^n)|$$

   $u_{\text{exact}}(x,t)$ shows the exact solution.

Report (Exercise 2)

A numerical scheme is derived by the following procedure. Equally-spaced grid is assumed.

1. Construct the Lagrangian interpolation polynomial $u_{\mathrm{Lag}}(x)$ using the values at three grid points ($x = x_{i-1}$, $x = x_i$, and x_{i+1}) at the time step n. (u^n_{i-1}, u^n_i, and u^n_{i+1}).
2. Consider that $u_{\mathrm{Lag}}(x)$ evolves according to the exact solution. From the timestep n to $n+1$, the $u_{\mathrm{Lag}}(x)$ profile moves at the speed c.

   If the value of the advected profile u_{Lag} at $x = x_i$ and t^{n+1} is u^{n+1}_i, show u^{n+1}_i using Δt, Δx, c, u^n_{i-1}, u^n_i, and u^n_{i+1}.

Report (Exercise 2)

3. Implement the constructed method in the sample code, and solve the advection equation numerically.
   - copy the sample code of the finite-difference method in your google drive and rename it. The implementation is done in the new code.

   It is not necessary to show your sample code in this report.
4. Run the simulation with the square-wave initial condition.

   Describe how the wave profile evolves (using hand-drawn figures or attaching image files are fine).
5. Run the simulation with the Gaussian initial condition (see Exercise 1).

   Describe how the wave profile evolves (using hand-drawn figures or attaching image files are fine).
6. In a similar way as in Exercise 1, perform convergence test of the constructed method by changing the grid number. The Gaussian function shown in Exercise 1 is used.

Submission

- Write your report on papers and put it in Iwasaki's mailbox in the secretary's office of Division of Science,
  or scan it and email the data to me.

  Any format is fine. Handwritten in 日本語 or in English, LaTeX, or Microsoft Word....
- **Deadline: 28th November 2024, 5:00 pm**