Notes

1. **Weak solutions (弱解)**

   In LSWE and NSWE, shock waves (water waves with a discontinuous front) are known to be solutions. However, since these solutions are not differentiable at the jumps, they do not satisfy the governing equations in the traditional way (because some derivatives do not exist). It is therefore convenient to regard these solutions as "weak solutions" to the corresponding ODE or PDE. A weak solution must satisfy certain mathematical conditions. You can see Wikipedia or the textbook for more details.

2. **Entropy conditions (熵條件)**

   In general, weak solutions are not unique, and there can be many weak solutions to a system of equations. Additional mathematical conditions are needed to identify the physically relevant weak solution. Such conditions are called the entropy conditions, because the idea originated from the concept of "entropy" in the study of thermodynamics (熱力學). Nowadays, the term "entropy condition" is widely used in many fields, such as shallow water waves, even though they are not related to thermodynamics. You can see Wikipedia or the textbook for more details on the entropy conditions.

3. **Finite difference method (有限差分法)**

   Very simply speaking, a finite different method discretizes a system of equations using a number of grid points based on finite difference formulas. The numerical value calculated at each grid point represents the numerical solution at that location. There can be many different ways to discretize a given set of equations using finite difference methods.

4. **Difficulties with numerically solving nonlinear equations**

   In solving nonlinear equations (such as the NSWE), finite difference methods may lead to numerical results that converge to a wrong function (which may not even be a weak solution to the original equation, or a physically wrong weak solution that does not satisfy the entropy condition).

   Example

   Consider the Burger's equation:

   $$u_t + uu_x = 0. \tag{1}$$

   A simple finite difference method to solve this equation is to discretize it as

   $$U_i^{(n+1)} = U_i^{(n)} - \frac{\Delta t}{\Delta x} U_i^{(n)} (U_i^{(n)} - U_{i-1}^{(n)}). \tag{2}$$

   Reasonable-looking discontinuous numerical solutions may be obtained by using (2) to update the discretized U_i. However, it is possible for this simple numerical method to converge to a wrong function that is not a solution to (1). An example is shown in Figure 1. More restrictions need to be placed on the numerical method to ensure that the converged numerical solutions are correct solutions to the governing equation.

5. **Conservative methods (保守方法)**

   To ensure that numerical methods do not converge to non-solutions, we need to write the discretized equation in *conservation form*, which means it has the form

   $$U_i^{(n+1)} = U_i^{(n)} - \frac{\Delta t}{\Delta x} \Big[F(U_{i-p}^{(n)}, U_{i-p+1}^{(n)}, \cdots, U_{i+q}^{(n)}) - F(U_{i-p-1}^{(n)}, U_{i-p}^{(n)}, \cdots, U_{i+q-1}^{(n)}) \Big], \tag{3}$$

   where p and q are integers and F is called the *numerical flux function*.

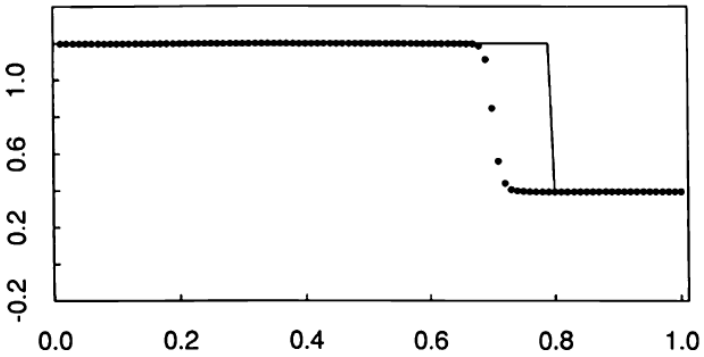

Figure 1: True (solid line) and computed (dots) solutions to Burgers' equation using a nonconservative method. From LeVeque (1992).

In the simplest case, $p = 0$ and $q = 1$, (3) reduces to

$$U_i^{(n+1)} = U_i^{(n)} - \frac{\Delta t}{\Delta x}\Big[F(U_i^{(n)}, U_{i+1}^{(n)}) - F(U_{i-1}^{(n)}, U_i^{(n)})\Big]. \tag{4}$$

Example

Here we consider the Burger's equation again. This time, we write it as

$$u_t + (\frac{1}{2}u^2)_x = 0 \tag{5}$$

instead of (1).

We can use a simple finite difference method to discretize (5) as

$$U_i^{(n+1)} = U_i^{(n)} - \frac{\Delta t}{\Delta x}\Big[\frac{1}{2}(U_i^{(n)})^2 - \frac{1}{2}(U_{i-1}^{(n)})^2\Big]. \tag{6}$$

Since (6) is now written in conservation form, i.e., it has the same form as (4), with

$$F(U_i^{(n)}, U_{i+1}^{(n)}) = \frac{1}{2}(U_i^{(n)})^2, \tag{7}$$

this numerical method will compute numerical results that converge to the correct solution.

6. **Lax-Wendroff theorem (拉文定理)**

   The Lax-Wendroff theorem states that if a conservative numerical method converges to some function as the grid is refined, then this function is a weak solution to the governing equation.

   However, the theorem does not guarantee that a numerical method will converge, nor does it indicate which weak solution the numerical method converges to if there are multiple weak solutions.

7. **Finite volume method (有限體積法)**

   When we write equations in conservation form, it becomes more convenient to interpret and solve these equations using finite volume methods (as opposed to the more general finite difference methods).

The weak solution $u(x,t)$ satisfies the integral form of the conservation law,

$$\begin{aligned}&\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t_{n+1})dx \\ &= \int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t_n)dx - \Big[\int_{t_n}^{t_{n+1}} f\big(u(x_{i+1/2},t_n)\big)dt - \int_{t_n}^{t_{n+1}} f\big(u(x_{i-1/2},t_n)\big)dt\Big]\end{aligned}, \tag{8}$$

where $f(u)$ is the flux of $u(x,t)$. Dividing both sides by Δx gives

$$\begin{aligned}&\frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t_{n+1})dx \\ &= \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t_n)dx - \frac{1}{\Delta x}\Big[\int_{t_n}^{t_{n+1}} f\Big(u(x_{i+1/2},t_n)\Big)dt - \int_{t_n}^{t_{n+1}} f\Big(u(x_{i-1/2},t_n)\Big)dt\Big]\end{aligned}. \tag{9}$$

If we define cell-averaged $u(x,t)$ as

$$\overline{u}_i^{(n)} = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t_n)dx, \tag{10}$$

and the numerical flux function $F(u)$ as

$$F(u_{i+1/2}^{(n)}) = \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} f\Big(u(x_{i+1/2},t_n)\Big)dt, \tag{11}$$

we can write (9) as

$$\overline{u}_i^{(n+1)} = \overline{u}_i^{(n)} - \frac{\Delta t}{\Delta x}\Big[F(u_{i+1/2}^{(n)}) - F(u_{i-1/2}^{(n)})\Big]. \tag{12}$$

We thus see that the numerical flux function $F(u)$ is used to represent the time-averaged fluxes through the cell interfaces $x_{i-1/2}$ and $x_{i+1/2}$.

Since using the concept of "cells" makes writing and interpreting the equations easier, like how we simplify (8) to (12), it can be easier to solve the cell-averaged equations to find the cell-averaged solutions. Numerical methods that follow this train of thought are generally classified as *finite volume methods*. They are called "finite volume methods" because each cell is seen as a control volume.

A very simple explanation of the difference between finite difference methods and finite volume methods is sketched in Figure 2.

8. **Godunov-type numerical scheme**

In (12), how do we calculate the intercell fluxes in (12), $F(u_{i-1/2}^{(n)})$ and $F(u_{i+1/2}^{(n)})$, using the cell-averaged solutions $\overline{u}_i^{(n)}$? This is problematic because we only know the cell-averaged values – we do not know the actual distribution within each cell.

S. K. Godunov (郭都諾夫) proposed a simple way to address this issue in 1959. In its original form, Godunov's method is only first-order accurate in time and space. Many higher-order numerical methods have since been derived based on Godunov's method, and these schemes are commonly classified as Godunov-type schemes.

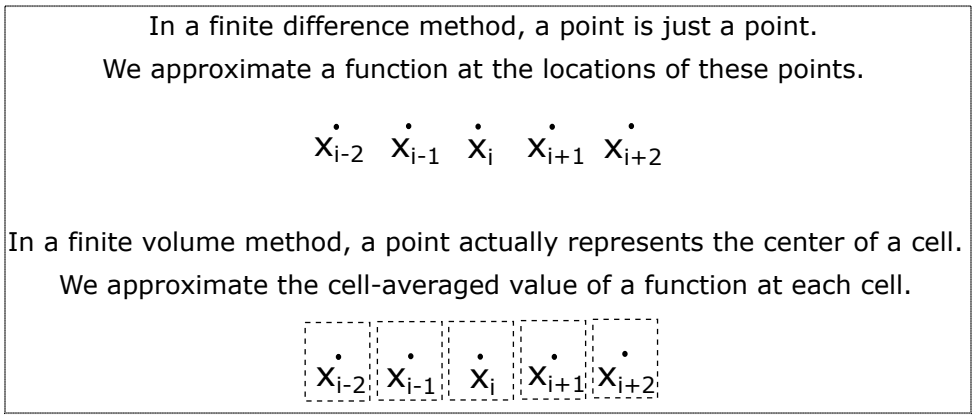

Figure 2: A simple explanation of the difference between finite difference methods and finite volume methods.

Godunov assumed that the distribution within each cell is constant. Namely,

$$u(x,t) = \begin{cases} \cdots \\ \overline{u}_{i-1}^{(n)}, \quad x_{i-1-\frac{1}{2}} \le x \le x_{i-1+\frac{1}{2}} \\ \overline{u}_{i}^{(n)}, \quad x_{i-\frac{1}{2}} \le x \le x_{i+\frac{1}{2}} \\ \overline{u}_{i+1}^{(n)}, \quad x_{i+1-\frac{1}{2}} \le x \le x_{i+1+\frac{1}{2}} \\ \cdots \end{cases} . \tag{13}$$

An illustration is shown in Figure 3.

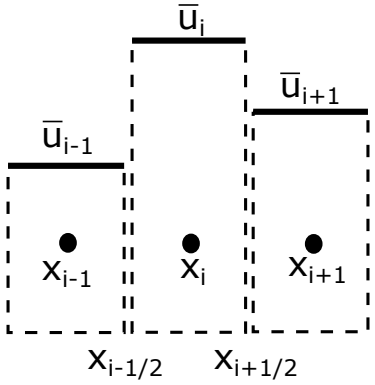

Figure 3: An illustration of cell data distribution based on Godunov's method.

To determine the flux at a cell interface, take $x_{i-1/2}$ for example, one then have to solve a local "dam-break" problem, where the quantity $u(x,t)$ jumps from $\overline{u}_{i-1}$ to $\overline{u}_i$. Such an intercell "dam-break" problem is formally referred to as a *Riemann problem (*黎曼問題*)*. For many commonly used PDEs, such as the NSWE, many Riemann solvers (which can be exact or approximate) have been developed.

9. **A shock-capturing numerical scheme for solving NSWE**

In this course, we will use a second-order accurate in space (MUSCL) and third-order accurate in time (SSP-RK) numerical scheme to solve the NSWE. This numerical scheme

is conservative and highly stable (it satisfies the "TVD" condition) near discontinuities. Therefore, it is considered a shock-capturing numerical scheme.

Instead of assuming a piece-wise constant data distribution like in Godunov's original proposal, we can interpolate between cells to get a higher-order data distribution within each cell. This interpolation process is often referred to as "data reconstruction". A stable and efficient method is the *MUSCL data reconstruction method*, which will be discussed below.

A possible distribution of $u(x, t_n)$ as an outcome of the MUSCL reconstruction process is illustrated in Figure 4. In general, there will be jumps at each cell interface because there are two possible values of u. For example, at the cell interface $x_{i-1/2}$, there are $u^-_{i-1/2}$ evaluated from the left (cell $i-1$), and $u^+_{i-1/2}$ evaluated from the right (cell i). Thus, local Riemann problems must be solved to determine the fluxes at cell interfaces. We will use the *HLLC approximate Riemann solver*, which will be discussed below.

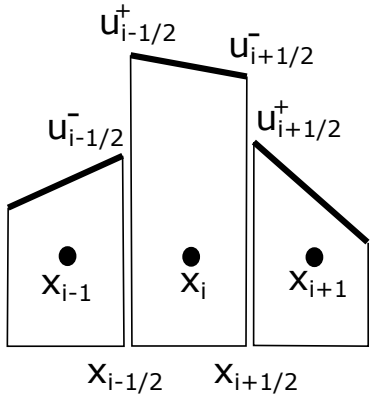

Figure 4: Sketch of a possible distribution of $u(x, t_n)$, as an outcome of the MUSCL reconstruction process.

Once the cell interface fluxes are determined, we will use the SSP-RK method to advance in time. The full details of the numerical scheme is provided at the end of the handout, after the introduction of the MUSCL reconstruction method and the HLLC approximate Riemann solver.

10. **MUSCL data reconstruction method**

Consider the cells sketched in Figure 4. The value of a cell-averaged quantity $\overline{u}(x, t_n)$ is known at each cell at a certain instant t_n: $\overline{u}_i$. If the intercell values of u are desired, e.g., $u_{i-1/2}$ or $u_{i+1/2}$, a reconstruction method is needed to obtain the intercell values from the known cell-averaged values.

There are many different ways to reconstruct the data. The superscript "+" is used in $u^+_{i-1/2}$ to indicate that the interpolation is based on cell i, which is to the right of this intercell location; the superscript "−" is used in $u^-_{i-1/2}$ to indicate that the interpolation is based on cell $i-1$, which is to the left of this intercell location.

Here we will discuss the *second-order accurate MUSCL-TVD (total variation diminishing, 全差變遞減格式)* method presented in Toro (2001). TVD is a mathematical condition that makes a numerical scheme stable at resolving shock waves (discontinuous solutions) and minimizes numerical oscillations. The intercell value between cell $i-1$ and cell i, $u^+_{i-1/2}$ calculated from the right (i.e., based on cell i) is determined as

$$u^+_{i-1/2} = \overline{u}_i - \frac{1}{2}\overline{\Delta}_i, \tag{14}$$

and the intercell value between cell i and cell $i+1$, $u^-_{i+1/2}$ calculated from the left (i.e., based on cell i) is determined as

$$u^-_{i+1/2} = \overline{u}_i + \frac{1}{2}\overline{\Delta}_i, \tag{15}$$

where $\overline{\Delta}_i$ is a limited slope (or a limited difference to be exact, since its technically not defined as a slope here), which is designed to satisfy the TVD condition.

We will use a version of the *van Leer slope limiter* to calculate the limited slope as

$$\begin{cases} \Delta^+_i = \overline{u}_{i+1} - \overline{u}_i \\ \\ \Delta^-_i = \overline{u}_i - \overline{u}_{i-1} \\ \\ \overline{\Delta}_i = \dfrac{\Delta^+_i|\Delta^-_i| + |\Delta^+_i|\Delta^-_i}{|\Delta^+_i| + |\Delta^-_i|}, \text{ for } |\Delta^+_i| + |\Delta^-_i| \neq 0 \\ \\ \overline{\Delta}_i = 0, \qquad\qquad\qquad\quad \text{for } |\Delta^+_i| + |\Delta^-_i| = 0 \end{cases}. \tag{16}$$

Note that the above expressions hold only for a constant step size Δx.

11. **HLLC approximate Riemann solver (近似黎曼解算子)**

In general, an intercell value of a function f approximated from the left is different from that approximated from the right, e.g., $f^-_{i-1/2} \neq f^+_{i-1/2}$. Therefore, additional treatment is needed to determine the intercell value, $f_{i-1/2}$. One popular approach is to solve a "dam-break" problem (also known as a Riemann problem) at each intercell boundary, as sketched in Figure 5. Using H to denote the local water depth, U to denote the depth-averaged velocity in the x-direction, we shall approximate the local Riemann problem in the x-direction as

$$\vec{\phi}_t + \vec{f}_x(\vec{\phi}) = 0, \;\; \vec{\phi} = \begin{bmatrix} H \\ HU \end{bmatrix}, \;\; \vec{f} = \begin{bmatrix} H \cdot U \\ HU \cdot U \end{bmatrix}. \tag{17}$$

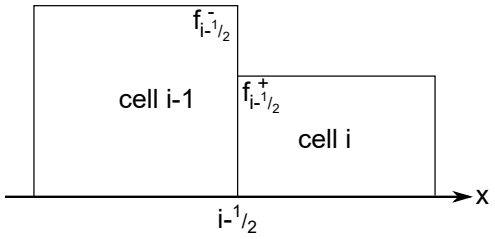

Figure 5: Sketch of the intercell Riemann problem to be solved.

In our numerical scheme, we implement the HLLC (Harten-Lax-van Leer contact) approximate Riemann solver discussed in Toro (1992), Toro (2001), and Shiach and Mingham (2009). The HLLC approximate Riemann solver determines the intercell value $f_{i-1/2}$ (for brevity, the subscript "$i-1/2$" is dropped in the below expressions) as

$$\vec{f} = \begin{cases} \vec{f}^- & \text{for } 0 \leq S^- \\ \vec{f}^- + S^-(\vec{\phi}^-_* - \vec{\phi}^-) & \text{for } S^- < 0 \leq u_* \\ \vec{f}^+ + S^+(\vec{\phi}^+_* - \vec{\phi}^+) & \text{for } u_* < 0 \leq S^+ \\ \vec{f}^+ & \text{for } S^+ < 0 \end{cases}, \tag{18}$$

where

$$\begin{cases} u_* = \dfrac{U^- + U^+}{2} + \sqrt{gH^-} - \sqrt{gH^+} \\ c_* = \dfrac{\sqrt{gH^-} + \sqrt{gH^+}}{2} + \dfrac{U^- - U^+}{4} \end{cases}, \tag{19}$$

$$S^- = \begin{cases} \min(U^- - \sqrt{gH^-}, u_* - c_*) & \text{for a wet cell } i-1 \\ U^+ - 2\sqrt{gH^+} & \text{for a dry cell } i-1 \end{cases}, \tag{20}$$

$$S^+ = \begin{cases} \max(U^+ + \sqrt{gH^+}, u_* + c_*) & \text{for a wet cell } i \\ U^- + 2\sqrt{gH^-} & \text{for a dry cell } i \end{cases}, \tag{21}$$

and

$$\vec{\phi}_*^- = H^-(\frac{S^- - U^-}{S^- - u_*}) \begin{bmatrix} 1 \\ u_* \end{bmatrix}, \ \vec{\phi}_*^+ = H^+(\frac{S^+ - U^+}{S^+ - u_*}) \begin{bmatrix} 1 \\ u_* \end{bmatrix}. \tag{22}$$

As can be inferred from the expressions for S^- and S^+, wave runup, i.e., a moving wet-dry boundary, is automatically captured by the HLLC approximate solver.

12. **Details of the numerical scheme**

Governing equations

The governing equations are the 1DH NSWE. We meticulously write the equations in the following form because it is more ideal for numerical computation:

$$\begin{cases} \dfrac{\partial H}{\partial t} + \dfrac{\partial HU}{\partial x} = 0 \\ \dfrac{\partial HU}{\partial t} + \dfrac{\partial \Big(HU^2 + \frac{1}{2} g(\eta^2 + 2\eta h) \Big)}{\partial x} = g\eta \dfrac{\partial h}{\partial x} \end{cases}, \tag{23}$$

where η is the free surface elevation, h is the still water depth, and $H = \eta + h$ is the total water depth.

It may be more convenient to write (23) in vector form:

$$\vec{\phi}_t + \vec{f}_x(\vec{\phi}) = \vec{s}, \tag{24}$$

where

$$\vec{\phi} = \begin{bmatrix} H \\ HU \end{bmatrix}, \ \vec{f} = \begin{bmatrix} HU \\ HU^2 + \dfrac{1}{2} g(\eta^2 + 2\eta h) \end{bmatrix} = \begin{bmatrix} F \\ G \end{bmatrix}, \ \vec{s} = \begin{bmatrix} 0 \\ g\eta \dfrac{\partial h}{\partial x} \end{bmatrix}. \tag{25}$$

Solution procedure

To march in time, we will use the SSP-RK method to update H and $[HU]$ (note that U can be recovered by dividing $[HU]$ by H). In each of the three time-iterations in SSP-RK, we need to perform the MUSCL reconstruction on η and U in order to calculate F^+, F^-, G^+, and G^- at each cell interface. Then, we use the HLLC approximatye Riemann solver to determine the flux values F and G at each cell interface from the reconstructed F^+, F^-, G^+, and G^-.

The derivative(s) in the source term $\vec{s}$ on the right hand side of (24) will be discretized using the second-order central difference method.

Time-updating scheme

Written out fully, the numerical scheme is

$$\begin{cases} \text{First round:} \\ H_i^{(*)} = H_i^{(n)} - \dfrac{\Delta t}{\Delta x}\Big(F_{i+1/2}^{(n)} - F_{i-1/2}^{(n)}\Big) \\ [HU]_i^{(*)} = [HU]_i^{(n)} - \dfrac{\Delta t}{\Delta x}\Big(G_{i+1/2}^{(n)} - G_{i-1/2}^{(n)}\Big) - \dfrac{\Delta t}{2\Delta x} g\eta_i^{(n)}\Big(h_{i+1} - h_{i-1}\Big) \\ \text{Second round:} \\ H_i^{(**)} = \dfrac{3}{4}H_i^{(n)} + \dfrac{1}{4}H_i^{(*)} - \dfrac{1}{4}\dfrac{\Delta t}{\Delta x}\Big(F_{i+1/2}^{(*)} - F_{i-1/2}^{(*)}\Big) \\ [HU]_i^{(**)} = \dfrac{3}{4}[HU]_i^{(n)} + \dfrac{1}{4}[HU]_i^{(*)} - \dfrac{1}{4}\dfrac{\Delta t}{\Delta x}\Big(G_{i+1/2}^{(*)} - G_{i-1/2}^{(*)}\Big) - \dfrac{1}{4}\dfrac{\Delta t}{2\Delta x} g\eta_i^{(*)}\Big(h_{i+1} - h_{i-1}\Big) \\ \text{Third round:} \\ H_i^{(n+1)} = \dfrac{1}{3}H_i^{(n)} + \dfrac{2}{3}H_i^{(**)} - \dfrac{2}{3}\dfrac{\Delta t}{\Delta x}\Big(F_{i+1/2}^{(**)} - F_{i-1/2}^{(**)}\Big) \\ [HU]_i^{(n+1)} = \dfrac{1}{3}[HU]_i^{(n)} + \dfrac{2}{3}[HU]_i^{(**)} - \dfrac{2}{3}\dfrac{\Delta t}{\Delta x}\Big(G_{i+1/2}^{(**)} - G_{i-1/2}^{(**)}\Big) - \dfrac{2}{3}\dfrac{\Delta t}{2\Delta x} g\eta_i^{(**)}\Big(h_{i+1} - h_{i-1}\Big) \end{cases} . \tag{26}$$

Reconstruction and flux calculation

In practice, it may be more beneficial to perform the MUSCL reconstruction on η and U. Recall that η is related to H by $H = \eta + h$, and U can be obtained as $[HU]/H$.

Using the cell interface at $x_{i-1/2}$ for example, before each round of the SSP-RK iteration, we perform the MUSCL reconstruction to obtain $\eta_{i-1/2}^+$, $\eta_{i-1/2}^-$, $U_{i-1/2}^+$, and $U_{i-1/2}^-$. The total water depth can be calculated as

$$H_{i-1/2}^+ = \eta_{i-1/2}^+ + h_{i-1/2}, \;\; H_{i-1/2}^- = \eta_{i-1/2}^- + h_{i-1/2}. \tag{27}$$

Since the still water depth $h(x)$ is a given and doesn't change in time, we can simply use linear interpolation to find its intercell values.

Then, we feed

$$\vec{\phi}_{i-1/2}^+ = \begin{bmatrix} H_{i-1/2}^+ \\ H_{i-1/2}^+ U_{i-1/2}^+ \end{bmatrix}, \;\; \vec{f}_{i-1/2}^+ = \begin{bmatrix} H_{i-1/2}^+ U_{i-1/2}^+ \\ H_{i-1/2}^+ (U_{i-1/2}^+)^2 + \dfrac{1}{2}g\Big((\eta_{i-1/2}^+)^2 + 2\eta_{i-1/2}^+ h_{i-1/2}\Big) \end{bmatrix}. \tag{28}$$

and

$$\vec{\phi}_{i-1/2}^- = \begin{bmatrix} H_{i-1/2}^- \\ H_{i-1/2}^- U_{i-1/2}^- \end{bmatrix}, \;\; \vec{f}_{i-1/2}^- = \begin{bmatrix} H_{i-1/2}^- U_{i-1/2}^- \\ H_{i-1/2}^- (U_{i-1/2}^-)^2 + \dfrac{1}{2}g\Big((\eta_{i-1/2}^-)^2 + 2\eta_{i-1/2}^- h_{i-1/2}\Big) \end{bmatrix}. \tag{29}$$

into the HLLC solver to obtain the numerical fluxes, i.e., $F_{i-1/2}$, $F_{i+1/2}$, $G_{i-1/2}$, and $G_{i+1/2}$ in (26).

Stability condition

The CFL condition needs to be satisfied:

$$\Delta t \le C_{\text{CFL}} \frac{\Delta x}{u_{\text{CFL}}}, \tag{30}$$

where u_{CFL} is a characteristic speed. In nonlinear equations, the characteristic speed is not easily known. For NSWE, it is generally estimated as

$$u_{\mathrm{CFL}} \simeq \max(|U| + \sqrt{gH}). \tag{31}$$

Since U, H, and u_{CFL} change in time, Δt also changes in time. Thus, a *dynamic time step* (動態時間步) is used – Δt needs to be calculated at each time step using the latest numerical results.

While the theoretical maximum Courant number C_{CFL} of this numerical scheme can be derived to be one for linearized equations, there is no theoretical proof for nonlinear equations. In my experience (and studies by other researchers), $C_{\mathrm{CFL}} = 0.9$ is stable for 1DH problems.

References

Shiach, J. B. and C. G. Mingham (2009), "A temporally second-order accurate Godunov-type scheme for solving the extended Boussinesq equations." *Coastal Eng.*, 56, 32–45.

Toro, E. F. (1992), "Riemann problems and the WAF method for solving two-dimensional shallow water equations." *Philos. Trans. R. Soc. Lond., A.*, 338, 43–68.

Toro, E. F. (2001), *Shock-capturing methods for free-surface shallow flows*. Wiley.