# Computation of Shallow Water Equation

Student ID: r07525117 Student Name: Chang-Syuan Syu

## 1. Governing Equations

In this study, I will use basic numerical methods to solve the linear shallow water equation (LSWE) in 1DH:

$$\begin{cases} \dfrac{\partial \eta}{\partial t} + \dfrac{\partial hU}{\partial x} = 0 \\ \dfrac{\partial U}{\partial t} = -g\dfrac{\partial \eta}{\partial x} \end{cases} \tag{1}$$

Where $\eta$ is the free surface elevation, $U$ is the horizontal flow velocity, $h$ is the still water depth, and $g$ is the gravitational acceleration.

## 2. Discretization

We can employ the Lax-Friedrichs method to discretize (1) as

$$\begin{cases} \eta_i^{(n+1)} = \dfrac{\eta_{i+1}^{(n)} + \eta_{i-1}^{(n)}}{2} - \dfrac{\Delta t}{2\Delta x}\left[U_{i+1}^{(n)} h_{i+1} - U_{i-1}^{(n)} h_{i-1}\right] \\ U_i^{(n+1)} = \dfrac{U_{i+1}^{(n)} + U_{i-1}^{(n)}}{2} - \dfrac{\Delta t}{2\Delta x} g\left[\eta_{i+1}^{n} - \eta_{i-1}^{n}\right] \end{cases} \tag{2}$$

where (n) denotes variables at the current time step, (n + 1) denotes variables at the new time step, *i* denotes the *i*-th discretized node in space, $\Delta x$ is the step size in space, and $\Delta t$ is the step size in time

The relation between $\Delta x$ and $\Delta t$ is required by the CFL (Courant, Friedrichs, and Lewy) condition:

$$\Delta t = C_{\mathrm{CFL}} \frac{\Delta x}{\sqrt{g h_0}}, \tag{3}$$

where the constant $C_{\mathrm{CFL}}$ is often referred to as the *Courant number* or the CFL number, and $h_0$ is the *maximum water depth* in the problem.

The Lax-Friedrichs method can be proved to be stable for $0 \leqq C_{\mathrm{CFL}} \leqq 1$ (see for example the textbook by LeVeque, 1992). In practice, $C_{\mathrm{CFL}} = 0.9$ is found to be an optimal choice, which I will use in this assignment.

## 3. Boundary Conditions

Let us consider a numerical wave flume spanning from $x$ = -12 m to $x$ = 24 m. The water depth is constant, $h = h_0 = 0.3$ m. The two ends of the wave flume are solid walls. At the left end, $x = -12$ m for example, the wall boundary condition means

$$\begin{cases} U_{i=1} = 0 \\ \dfrac{\partial U}{\partial t} = 0 = -g\dfrac{\partial \eta}{\partial x} \rightarrow \left(\dfrac{\partial \eta}{\partial x}\right)_{i=1} = 0 \end{cases} \quad (4)$$

This also means the ghost cells to the left of x = -12 have the values

$$\begin{cases} \eta_{i=0} = \eta_{i=2} \\ U_{i=0} = 0 \end{cases} \quad (5)$$

# 4. Initial Conditions

I specify the initial conditions - $\eta$ (x, 0) and $U$(x, 0). In constant water depth, we know that any wave of translation (平移波) moving at the speed $\sqrt{gh}$ is a solution to the 1DH LSWE (1). This can be verified by checking that a function of the form f(x - $\sqrt{gh} \cdot$ t) is a solution to (1) in constant water depth.

In an addition, based on the linear wave theory the horizontal flow velocity for linear shallow water waves can be determined as

$$U(x,t) = \frac{\eta(x,t)}{h}\sqrt{gh} \quad (6)$$

As the initial conditions for this assignment, I will use a wave of translation of the form

$$\eta(x,t) = Hsech^2\big(K(x - Ct)\big), \qquad K = \frac{1}{h}\sqrt{\frac{3H}{4h}}, \quad (7)$$

where $H$ is the wave height, $C$ denotes the wave speed ($C = \sqrt{gh}$ for LSWE), and sech($x$) is the hyperbolic secant function. $K$ can be seen as the effective wave number for this wave, and an effective wavelength $L$ can be defined as

$$L = \frac{2\pi}{K} \quad (8)$$

An effective wave period $T$ can also be defined:

$$T = \frac{L}{C} \quad (9)$$

A wave of the form (7), whose flow velocity can be calculated from (6), is called the *solitary wave* (孤立波). It is often used as a benchmark wave in many long-wave studies. I will use H = 0.04 m. In a water depth of h = 0.3 m, this means that the effective wavelength is *L* = 5.961 m, and the effective wave period is *T* = 3.475 s.

The initial conditions, i.e., $\eta(x; 0)$ and $U(x; 0)$, to be used in the simulations shown in the Figure 1.

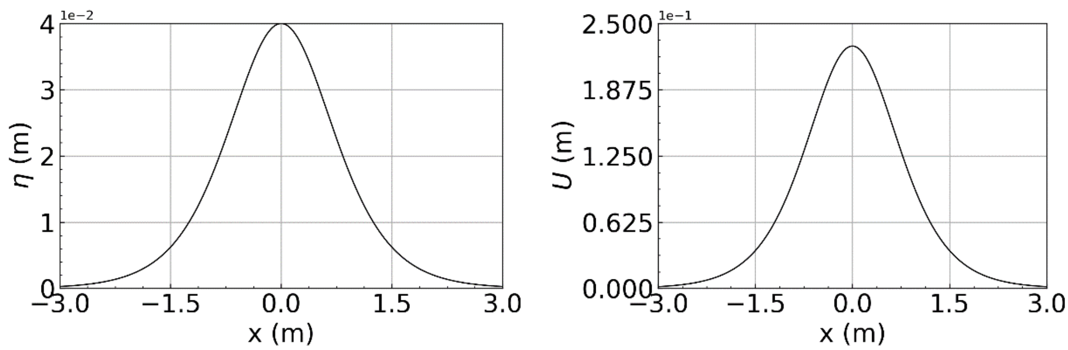


Figure 1: Initial value plot

# 5. Solve the 1DH LSWE

In this study, I written a program to solve the 1DH LSWE using these initial conditions, for $-12 < x < 24$ (m) and $0 < t < 6.95$ (s). I try using the step size $\Delta x$ = 0.06 m as a start. The codes are shown in the appendix.

# 6. Validation

In order to make sure my code runs correctly. I compared my numerical results for $\eta$ against the analytical solution at t = 6.95 s, i.e., (7). The results are shown in Figure 2

# 7. Behavior on Different Step Sizes

I run several simulations with different step sizes $\Delta x$ at physical time equal 6.95. The results are shown in Figure 2.

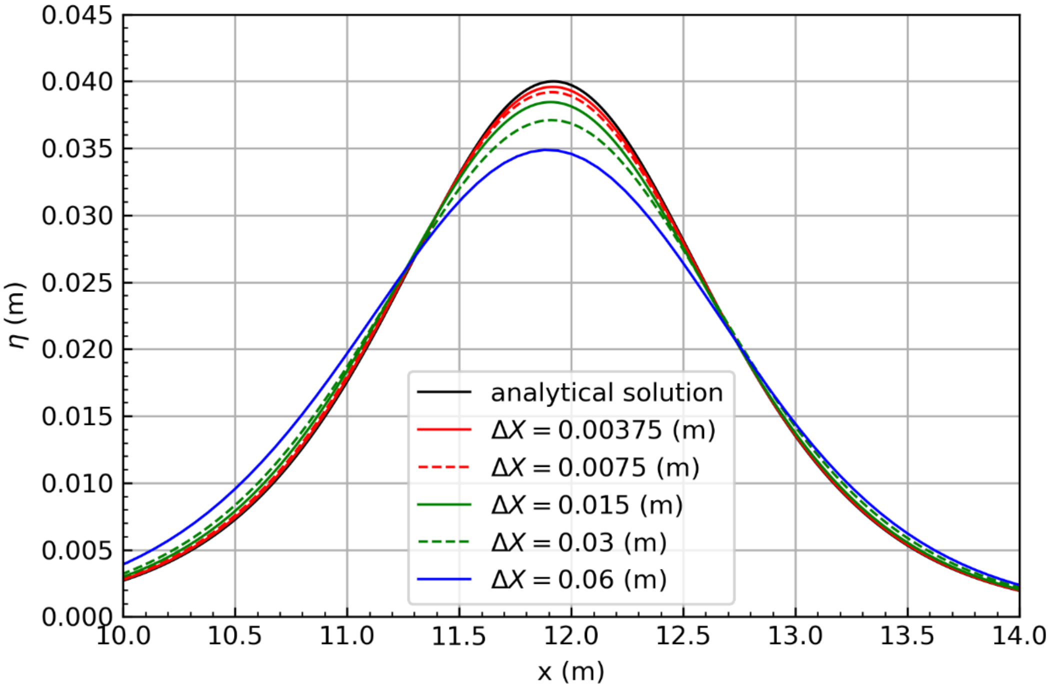


Figure 2: Reference plot for topic 6 and 7.

# 8. Grid Dependency Test

The results are shown in Figure 3 which shown that the convergence rate of the Lax-Friedrichs numerical scheme is first-order.

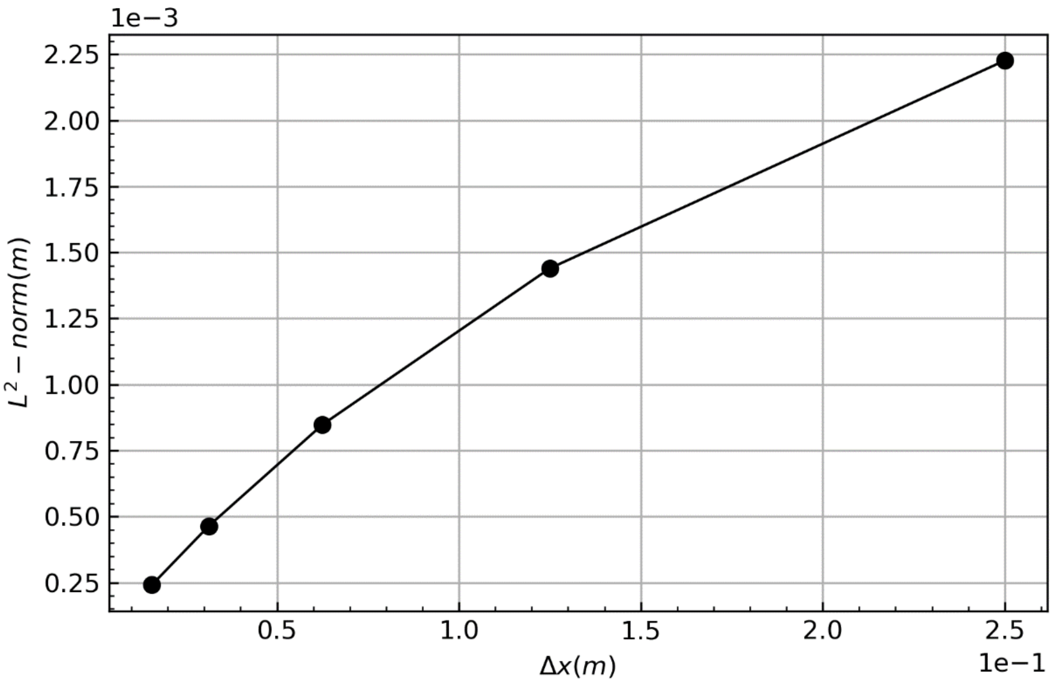


Figure 3: Reference plot for topic 8.

# Appendix

Python Code:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 17 12:33:45 2020

@author: 88693
"""

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import ScalarFormatter
import os
os.chdir('D:\ShallowWaterComputation\HW2b')
from numba import jit

g = 9.806
h = 0.3

def sech(x):
    return 1.0/np.cosh(x)

def eta(x, t=0, H=0.04):
    K = (1.0/h)*np.sqrt( (3*H) / (4*h) )
    C = np.sqrt(g*h)
    return H*np.power( sech( K*( x-C*t ) ) , 2 )

def U(x, t=0):
    TEMP = eta(x, t)
    return TEMP *np.sqrt(g*h) /float(h)

def
```

```
plot(x,y,xlim=None,ylim=None,xn=5,yn=5,xlabel='',ylabel='',fil
ename=None):
    fig, ax1 = plt.subplots()
    ax1.plot( x, y, '-', color = 'black', linewidth= 1)
    ax1.tick_params(which='both',direction='in')
    ax1.minorticks_on()
    ax1.grid()
    if ( xlim!=None ):
        ax1.set_xlim( xlim )
    if ( ylim!=None ):
        ax1.set_ylim( ylim )
    if ( xlabel!='' ):
        ax1.set_xlabel(xlabel, fontsize=20)
    if ( xlabel!='' ):
        ax1.set_ylabel(ylabel, fontsize=20)
    plt.xticks( np.linspace(xlim[0],xlim[1],xn), fontsize=20 )
    plt.yticks( np.linspace(ylim[0],ylim[1],yn), fontsize=20 )
    yfmt = ScalarFormatter()
    yfmt.set_powerlimits((0,0))
    ax1.yaxis.set_major_formatter(yfmt)
    #plt.ticklabel_format(style='sci', axis='y',
scilimits=(0,0))
    fig.tight_layout()
    fig.savefig( filename , dpi=300)




@jit
def Numeri(deltaX,End_Time=6.95,Ccfl=1):
    deltaT = float(deltaX)/np.sqrt(g*h)*Ccfl

    x = np.arange(-12, 24, deltaX)
    t = np.arange(0, End_Time, deltaT)

    tlen = len(t)
    xlen = len(x)
```

```
    ETA = np.empty( ( tlen, xlen+2 ) )
    Vel_U = np.empty( ( tlen, xlen+2 ) )

    #####Initialize

    ##Cell
    x = np.append(x, x[-1]+deltaX )
    x = np.append(x[0]-deltaX, x )

    xlen = len(x)

    for i in range(1,xlen-1):
        ETA[0][i] = eta(x[i])

    for i in range(1,xlen-1):
        Vel_U[0][i] = U(x[i])

    ##BC
    ETA[0][0] = ETA[0][2]
    ETA[0][-1] = ETA[0][-3]
    Vel_U[0][0] = 0
    Vel_U[0][-1] = 0

    n = 0
    #######Numerical Solution
    while( n < (tlen-1) ):
        #cell
        for i in range(1,xlen-1):
            ETA[n+1][i] = ( ETA[n][i+1] + ETA[n][i-1] )/2.0 -
(deltaT*h)/(2*deltaX)*( Vel_U[n][i+1] - Vel_U[n][i-1] )
            Vel_U[n+1][i] = ( Vel_U[n][i+1] + Vel_U[n][i-1] )
/2.0 - (deltaT*g)/(2*deltaX)*( ETA[n][i+1] - ETA[n][i-1] )
        #BC
        ETA[n+1][0] = ETA[n+1][2]
        ETA[n+1][-1] = ETA[n+1][-3]
        Vel_U[n+1][0] = 0
        Vel_U[n+1][-1] = 0
        n = n+1
```

```
    return x, ETA

####Initial Plot
deltaX = 0.00375
x = np.arange(-12, 24, deltaX)
U0 = U(x)
ETA0 = eta(x, t=0, H=0.04)

plot(x,ETA0, xlim = [-3,3], ylim = [0,0.04],xlabel='x
(m)' ,ylabel='${\eta}$ (m)',filename='ETA0.png')
plot(x,U0, xlim = [-3,3], ylim = [0,0.25],xlabel='x
(m)' ,ylabel='${U}$ (m)',filename='U0.png')

####Q1
Cfl = 0.9
x1,ETA1= Numeri(0.00375, Ccfl=Cfl)
x2,ETA2= Numeri(0.0075, Ccfl=Cfl)
x3,ETA3= Numeri(0.015, Ccfl=Cfl)
x4,ETA4= Numeri(0.03, Ccfl=Cfl)
x5,ETA5= Numeri(0.06, Ccfl=Cfl)


#Analytical Solution
x = np.arange(-12, 24, 0.015)
AnalyticalSol = eta(x,t=6.95)

fig, ax1 = plt.subplots()
l1 = ax1.plot( x, AnalyticalSol, '-', color = 'black',
linewidth= 1)
l2 = ax1.plot( x1, ETA1[-1], '-', color = 'r', linewidth= 1)
l3 = ax1.plot( x2, ETA2[-1], '--', color = 'r', linewidth= 1)
l4 = ax1.plot( x3, ETA3[-1], '-', color = 'g', linewidth= 1)
l5 = ax1.plot( x4, ETA4[-1], '--', color = 'g', linewidth= 1)
l6 = ax1.plot( x5, ETA5[-1], '-', color = 'b', linewidth= 1)


ax1.set_xlim( [10,14] )
ax1.set_ylim( [0,0.045] )
```

```
ax1.tick_params(which='both',direction='in')
ax1.minorticks_on()
ax1.grid()
ax1.set_xlabel( 'x (m)' )
ax1.set_ylabel( '${\eta}$ (m)' )

lns =  l1 + l2 + l3 + l4 + l5 +l6
labels = [  'analytical solution', '${\Delta}X = 0.00375$
(m)' ,'${\Delta}X = 0.0075$ (m)' ,'${\Delta}X = 0.015$
(m)' ,'${\Delta}X = 0.03$ (m)','${\Delta}X = 0.06$ (m)' ]
ax1.legend(lns ,labels , loc = 'lower center' )

fig.tight_layout()
fig.savefig( 'Q1.png', dpi=300)


###L-norm
@jit
def norm(EtaNum, EtaTheory):
    temp = 0
    for i , j in zip(EtaNum, EtaTheory):
        temp = temp + np.power(i-j, 2)
    N = len(EtaNum)
    temp = np.sqrt( float(temp) / float(N) )
    return temp




Cfl = 0.9
x1,ETA1= Numeri(0.015625, Ccfl=Cfl)
x2,ETA2= Numeri(0.03125, Ccfl=Cfl)
x3,ETA3= Numeri(0.0625, Ccfl=Cfl)
x4,ETA4= Numeri(0.125, Ccfl=Cfl)
x5,ETA5= Numeri(0.25, Ccfl=Cfl)

AnalySol1 = eta(x1,t=6.95)
AnalySol2 = eta(x2,t=6.95)
AnalySol3 = eta(x3,t=6.95)
```

```
AnalySol4 = eta(x4,t=6.95)
AnalySol5 = eta(x5,t=6.95)


e1 = norm(ETA1[-1],AnalySol1 )
e2 = norm(ETA2[-1],AnalySol2 )
e3 = norm(ETA3[-1],AnalySol3 )
e4 = norm(ETA4[-1],AnalySol4 )
e5 = norm(ETA5[-1],AnalySol5 )

x = np.array([0.015625, 0.03125, 0.0625, 0.125, 0.25])
Error = np.array([e1,e2,e3,e4,e5])

fig, ax1 = plt.subplots()
l1 = ax1.plot( x, Error, '-o', color = 'black', linewidth= 1)

#ax1.set_xlim( [10,14] )
#ax1.set_ylim( [0,0.045] )
ax1.tick_params(which='both',direction='in')
ax1.minorticks_on()
ax1.grid()
ax1.yaxis.get_major_formatter().set_powerlimits((0,2))
ax1.xaxis.get_major_formatter().set_powerlimits((0,1))
ax1.set_xlabel( '${\Delta}x (m)$ ' )
ax1.set_ylabel( '${L^2}-norm (m)$' )


fig.tight_layout()
fig.savefig( 'Q2.png', dpi=300)
```