Practical Numerical Methods with Python for Shallow Water Equations

Baijun Xie
Mechanical Engineering Department
George Washington University

December 14, 2017

1. Introduction

The shallow water equations(SWEs) are a set of hyperbolic partial differential equations that describe the flow below a pressure surface in a fluid[1]. The equations are derived from depth-integrating the NavierStokes equations, in the case where the horizontal length scale is much greater than the vertical length scale. Under this condition, conservation of mass implies that the vertical velocity of the fluid is small, which means the vertical velocity is negligible. Under this condition, we will more focus on the velocity at the horizontal directions.

2. Shallow Water Equation

2.1 2D Shallow Water Equations with Conservative form

The equation of the Conservative form of SWEs is:

$$\frac{\partial \rho h}{\partial t} + \frac{\partial(\rho h u)}{\partial x} + \frac{\partial(\rho h v)}{\partial y} = 0 \tag{1}$$

$$\frac{\partial(\rho h u)}{\partial t} + \frac{\partial(\rho h u^2 + \frac{\rho g h^2}{2})}{\partial x} + \frac{\partial(\rho h u v)}{\partial y} = 0 \tag{2}$$

$$\frac{\partial(\rho hv)}{\partial t} + \frac{\partial(\rho huv)}{\partial x} + \frac{\partial(\rho hv^2 + \frac{\rho g h^2}{2})}{\partial y} = 0 \tag{3}$$

which is a coupled system of PDEs.

The SWEs are derived from the Navier-Stokes equations. The equations obey the conservation law. The first equation is governing the conservation of mass, and both second and third equations are governing the conservation of momentum.

where,

u: is the velocity in the x direction, or zonal velocity

v: is the velocity in the y direction, or meridional velocity

h: is the height deviation of the horizontal pressure surface from its mean height

g: is the acceleration due to gravity

ρ: is the fluid density, we simulate the water at here, so $\rho = 1$ and will not consider this term in the next few steps

2.2 Vectorization

We can vectorized these three equations:

$$\frac{\partial}{\partial t}\begin{bmatrix} h \\ hu \\ hv \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} hu \\ hu^2 + \frac{gh^2}{2} \\ huv \end{bmatrix} + \frac{\partial}{\partial y}\begin{bmatrix} hv \\ huv \\ hv^2 + \frac{gh^2}{2} \end{bmatrix} = 0 \tag{4}$$

Set the variables $\underline{\mathbf{h}}$ for the equations are:

$$\underline{\mathbf{H}} = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix} = \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix} \tag{5}$$

Similarily,

$$\underline{\mathbf{U}} = \begin{bmatrix} hu \\ hu^2 + \frac{gh^2}{2} \\ huv \end{bmatrix} = \begin{bmatrix} H_2 \\ \frac{H_2^2}{H_1} + \frac{gH_1^2}{2} \\ \frac{H_2 H_3}{H_1} \end{bmatrix} \tag{6}$$

$$\underline{\mathbf{V}} = \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{gh^2}{2} \end{bmatrix} = \begin{bmatrix} H_3 \\ \frac{H_2 H_3}{H_1} \\ \frac{H_3^2}{H_1} + \frac{gH_1^2}{2} \end{bmatrix} \tag{7}$$

Now, the system of equations can be written as:

$$\frac{\partial}{\partial t}\underline{\mathbf{H}} + \frac{\partial}{\partial x}\underline{\mathbf{U}} + \frac{\partial}{\partial y}\underline{\mathbf{V}} = 0 \tag{8}$$

2.3 Discretization with FTCS scheme

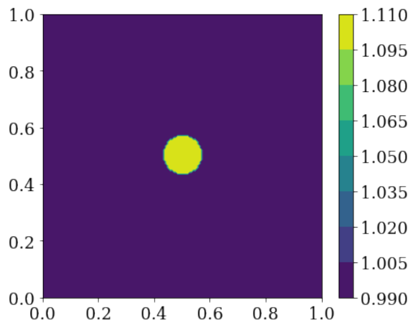
Then write this out discretized using forward difference in time, and central difference in space, using an explicit scheme.

$$\frac{\underline{\mathbf{H}}_{i,j}^{n+1} - \underline{\mathbf{H}}_{i,j}^{n}}{\Delta t} = -\left(\frac{\underline{\mathbf{U}}_{i+1,j}^{n} - \underline{\mathbf{U}}_{i-1,j}^{n}}{2\Delta x} + \frac{\underline{\mathbf{V}}_{i,j+1}^{n} - \underline{\mathbf{V}}_{i,j-1}^{n}}{2\Delta y}\right) \tag{9}$$

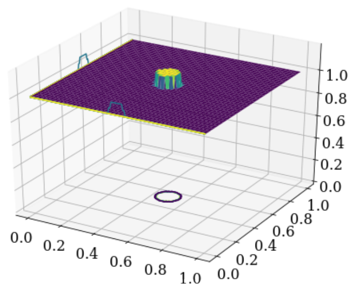
Rearranging the equation to solve for the value at the next time step, $\underline{\mathbf{H}}_{i,j}^{n+1}$, yields

$$\underline{\mathbf{H}}_{i,j}^{n+1} = \underline{\mathbf{H}}_{i,j}^{n} - \Delta t\left(\frac{\underline{\mathbf{U}}_{i+1,j}^{n} - \underline{\mathbf{U}}_{i-1,j}^{n}}{2\Delta x} + \frac{\underline{\mathbf{V}}_{i,j+1}^{n} - \underline{\mathbf{V}}_{i,j-1}^{n}}{2\Delta y}\right) \tag{10}$$

2.4 Initial Condition



(a) Initial condition on imshow



(b) Initial condition on 3D plot

Figure 1: The initial condition on two different views.

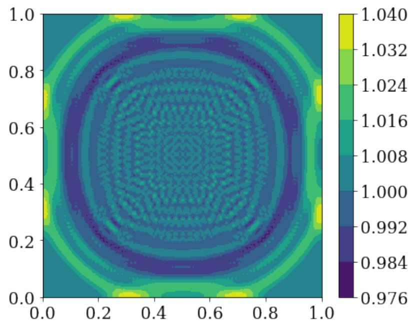
For the simulation, I simply imagine this is a "bathtub" model. The simulation environment is a square box with 4 reflective boundaries and the size of the box is 1-by-1. For the initial condition, the uniform height of the still water is 1 and I add a perturbation in the center of the surface whose height is 1.1, which simulating an impulsive disturbance like a water drop hitting the surface. Then, due to the gravity, the perturbation will drop down and spread out from the center.

3. Results

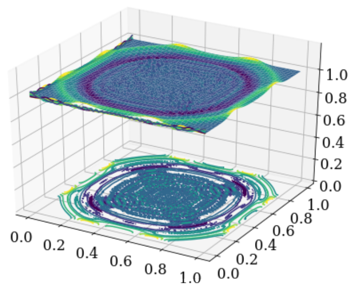
3.1 Animations

Python can use two ways to simulate the process of the SWEs:

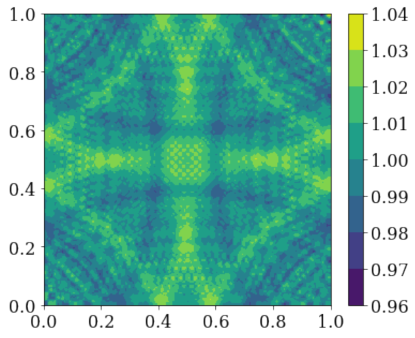
1. Using imshow, the perspective from up-to-down.
2. Using 3D plot, which get a 3D animation.



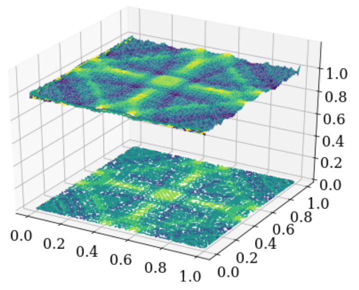
(a) Intermediate state on imshow



(b) Intermediate state on 3D plot



(c) Final state on imshow



(d) Final state on 3D plot

Figure 2: The final state on two different views.

Basically, because the Neumann boundary conditions are enforced at four boundaries, the resulting waves will propagate back and forth over the region. A few snapshots of the intermediate and final state of the process are shown in Figure 2.

3.2 Grid-convergence

Using the grid-convergence method introduced in the module 1 of numerical-mooc[2], we can examine the FTCS scheme in different dt values.

From Figure 3. we can see, as Δt shrinks, the error get smaller compared to the fine-grid solution.

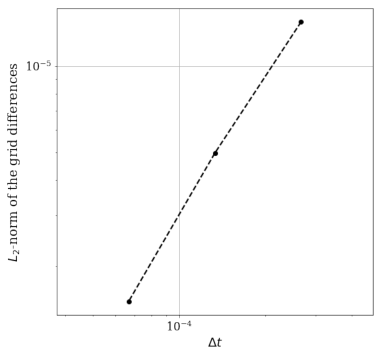


Figure 3: L2-norm of the grid differences

4. Discussion

4.1 Discretization with Lax-Wendroff scheme

The numerical method we used, the FTCS scheme giving the first-order convergence in time and second-order convergence in space. However, FTCS is unstable for advection equations, as well as the SWEs. At this part, we can consider using the Lax-Wendroff scheme to achieve second-order accuracy in both space and time[3]. At this part, the numerical results between FTCS and Lax-Wendroff scheme are compared. Finally, results show that the Lax-Wendroff scheme will be more stable than FTCS scheme, and we can implement the Lax-Wendroff scheme for SWEs in the future.

The Lax-Wendroff method for SWEs involves two stages in each time step. The first stage is a half step which represents the solution at the midpoints of the edges in the finite difference grid.

The equations for the first stage:

$$\underline{\mathbf{H}}x^{n+1/2}_{i+1/2,j} = \frac{1}{2}(\underline{\mathbf{H}}^n_{i+1,j} + \underline{\mathbf{H}}^n_{i,j}) + \frac{\Delta t}{2\Delta x}(\underline{\mathbf{U}}^n_{i+1,j} - \underline{\mathbf{U}}^n_{i,j}) \tag{11}$$

$$\underline{\mathbf{H}}y^{n+1/2}_{i,j+1/2} = \frac{1}{2}(\underline{\mathbf{H}}^{n}_{i,j+1} + \underline{\mathbf{H}}^{n}_{i,j}) + \frac{\Delta t}{2\Delta y}(\underline{\mathbf{V}}^{n}_{i,j+1} - \underline{\mathbf{V}}^{n}_{i,j}) \tag{12}$$

Then in the second stage, we compute the next time step by using the values computed in the first stage, and the equation of the second stage can be rewritten as:

$$\underline{\mathbf{H}}^{n+1}_{i,j} = \underline{\mathbf{H}}^{n}_{i,j} - \Delta t \left(\frac{\underline{\mathbf{U}}^{n+1/2}_{i+1/2,j} - \underline{\mathbf{U}}^{n+1/2}_{i-1/2,j}}{\Delta x} + \frac{\underline{\mathbf{V}}^{n+1/2}_{i,j+1/2} - \underline{\mathbf{V}}^{n+1/2}_{i,j-1/2}}{\Delta y} \right) \tag{13}$$

4.2 CFL condition

We have learned the CFL condition for the 1-D case of advection equation in the module 2 of numerical-mooc[2], which is:

$$\sigma = \frac{c\Delta t}{\Delta x} \leq 1 \tag{14}$$

For the 2-D case, the CFL condition becomes[4]

$$\frac{u_x \Delta t}{\Delta x} + \frac{v_y \Delta t}{\Delta y} \leq 1 \tag{15}$$

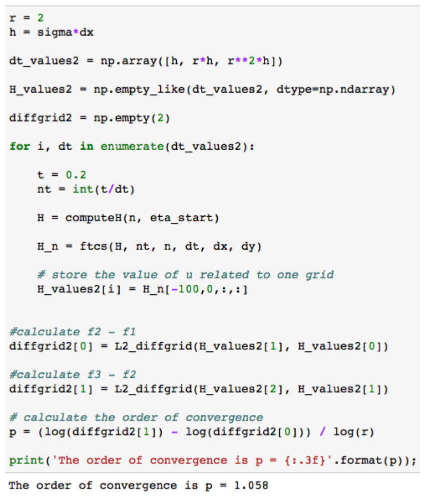
where u_x and v_y is the magnitude of the velocity on x and y direction, and both simply equal to 1 at here.

4.3 Order of convergence

The order of convergence is the rate at which the numerical solution approaches the exact one as the mesh is refined. Considering that we're not comparing with an exact solution, we use 3 grid resolutions that are refined at a constant ratio r to find the observed order of convergence (p), which is given by:

$$p = \frac{\log\left(\frac{f_3 - f_2}{f_2 - f_1}\right)}{\log(r)} \tag{16}$$

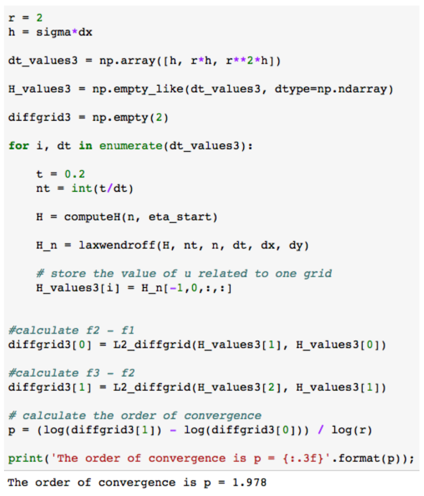
where f_1 is the finest mesh solution, and f_3 the coarsest.



```
r = 2
h = sigma*dx

dt_values2 = np.array([h, r*h, r**2*h])

H_values2 = np.empty_like(dt_values2, dtype=np.ndarray)

diffgrid2 = np.empty(2)

for i, dt in enumerate(dt_values2):

    t = 0.2
    nt = int(t/dt)

    H = computeH(n, eta_start)

    H_n = ftcs(H, nt, n, dt, dx, dy)

    # store the value of u related to one grid
    H_values2[i] = H_n[-100,0,:,:]


#calculate f2 - f1
diffgrid2[0] = L2_diffgrid(H_values2[1], H_values2[0])

#calculate f3 - f2
diffgrid2[1] = L2_diffgrid(H_values2[2], H_values2[1])

# calculate the order of convergence
p = (log(diffgrid2[1]) - log(diffgrid2[0])) / log(r)

print('The order of convergence is p = {:.3f}'.format(p));
```

```
The order of convergence is p = 1.058
```

(a) Order of convergence of FTCS



```
r = 2
h = sigma*dx

dt_values3 = np.array([h, r*h, r**2*h])

H_values3 = np.empty_like(dt_values3, dtype=np.ndarray)

diffgrid3 = np.empty(2)

for i, dt in enumerate(dt_values3):

    t = 0.2
    nt = int(t/dt)

    H = computeH(n, eta_start)

    H_n = laxwendroff(H, nt, n, dt, dx, dy)

    # store the value of u related to one grid
    H_values3[i] = H_n[-1,0,:,:]


#calculate f2 - f1
diffgrid3[0] = L2_diffgrid(H_values3[1], H_values3[0])

#calculate f3 - f2
diffgrid3[1] = L2_diffgrid(H_values3[2], H_values3[1])

# calculate the order of convergence
p = (log(diffgrid3[1]) - log(diffgrid3[0])) / log(r)

print('The order of convergence is p = {:.3f}'.format(p));
```

```
The order of convergence is p = 1.978
```

(b) Order of convergence of Lax-Wendroff

Figure 4: The final state on two different views.

From Figure 4. we can see, the order of convergence in time for FTCS scheme is about to 1, which is reasonable because the FTCS scheme is first-order convergence in time. For the Lax-Wendroff scheme, the order of convergence in time is about to 2. We can get a second-order accuracy in time by using the Lax-Wendroff scheme.

5. Summary

The project was mostly achieved. The simulation of the process of the SWEs was achieved finally, and the running time for the 3D animation took about 8 hours to run out. We may consider using the Colonial One of GWU to simulate the process of PDEs with large numbers of time steps. Regrettably, the boundary conditions setup is tricky for the Lax-Wendroff scheme, because the numerical values will overflow after the wave hit the boundary. The next step of the analysis of the SWEs could focus on the boundary condition, and we could consider constructing a Gaussian model for the perturbation of initial conditions which is more close to reality.

6. Acknowledgements

The author would like to thank Professor Aron Ahmadia for his suggestions and discussion for the final project. The author took advantage of the Jupyter Notebook "Practical Numerical Methods with Python" written by the Professor Lorena A. Barba and her group throughout this semester.

References

[1] https://en.wikipedia.org/wiki/Shallow_water_equations

[2] https://github.com/numerical-mooc/numerical-mooc

[3] https://en.wikipedia.org/wiki/Lax-Wendroff_method

[4] https://en.wikipedia.org/wiki/Courant-Friedrichs-Lewy_condition